Введение в хемометрику

v. 1.0.0, 18 августа 2023 г.

Сергей Кучерявский, Виталий Панчук, Юлия Монахова, Дмитрий Кирсанов

Содержание

Введение

Дорогой читатель, эта книга представляет собой учебник по хемометрике — сравнительно молодой дисциплине среди химических наук. Несмотря на юный по меркам естествознания возраст (первые научные работы с использованием хемометрики появились в середине 70-х годов XX века), без изучения этой дисциплины современное химическое образование сейчас уже не будет полным.

Это обусловлено несколькими причинами. Прежде всего это связано с тем, что современное аналитическое оборудование, которое мы используем, позволяет генерировать очень большие объемы данных, которые часто невозможно интерпретировать без соответствующей математической обработки. В то же время быстрое развитие компьютерной техники привело к появлению целого кластера новых научных направлений, связанных именно с обработкой больших массивов многомерных данных. Такие термины как “Data Science”, “Artificial intelligence”, “Big Data”, “Machine learning” (программа переводчик, основанная на методах машинного обучения перевела их как «Наука о данных», «Искусственный интеллект», «Большие данные», «Машинное обучение») у всех на слуху и большинство из нас пользуются их достижениями каждый день, например при поиске информации в интернете. Методы машинного обучения все больше проникают в разные сферы человеческой жизни и химия давно уже не исключение. В контексте химических исследований эти методы часто и называют методами хемометрики.

По определению Золотой книги ИЮПАК, https://goldbook.iupac.org, хемометрикой называется применение статистических методов для анализа химических данных, полученных в органической, аналитической или медицинской химии, а также планирование химического эксперимента и численное моделирование. Это определение, на наш взгляд, немного устарело и требует некоторых уточнений. Во-первых, статистические методы — слишком размытое понятие. Методы классической статистики (распределение Стьюдента, сравнение средних и стандартных отклонений, статистические тесты) давно и широко использовались в химии до появления хемометрики для решения метрологических задач (расчета погрешностей, воспроизводимости, сходимости, пределов обнаружения и т.д.). Однако лишь с развитием современных методов анализа многомерных данных хемометрика завоевала свою популярность. Во-вторых, область применения хемометрики не ограничивается перечисленными в определении ИЮПАК разделами химической науки. В настоящее время хемометрика применяется в экологической, физической, неорганической химии, химической кинетики и во многих других областях.

Наиболее широкое применение методы хемометрики нашли в области аналитической химии. Дело в том, что при анализе реальных объектов сложного состава, когда в пробе присутствует много компонентов,

их взаимное влияние может приводить к перекрывающимся неселективным сигналам, которые являются причиной снижения точности анализа. Традиционно эта проблема решается усложнением экспериментальных процедур: предварительным разделением компонентов пробы, или использованием более дорогостоящего оборудования с более высоким разрешением. Однако “математика дешевле физики” (как справедливо заметил Stuart A. Borman) и зачастую использование хемометрических методов на таких аналитических сигналах низкого качества позволяет достигать схожей точности анализа, что и при использовании дорогостоящего оборудования и сложных экспериментальных процедур. Этот подход успешно применяется, например, в хроматографии, молекулярной или атомной спектроскопии.

Развитие хемометрики привело даже к появлению новых аналитических инструментов и методов, которые ранее не могли быть использованы для анализа реальных объектов, поскольку в этих методах отсутствуют привычные для классической аналитической химии селективные аналитические сигналы от отдельных компонентов. Так, например, классической историей успеха хемометрики является ближняя инфракрасная спектроскопия. Спектр в этой области представляет собой наложение большого количества обертонов колебаний из средней ИК области. Использование таких сигналов в аналитических целях с помощью традиционных одномерных методов градуировки не представляется возможным. Хемометрические же методы позволяют легко выделить вклад отдельных химических компонент в форму БИК спектра, что стало причиной буквально лавинного роста приложений БИК в различных отраслях науки и промышленности в последние пару десятков лет.

Другим примером такого успеха является развитие мультисенсорных систем (“электронный нос”, “электронный язык”), где используются наборы слабоселективных сенсоров, каждый из которых дает информацию сразу от нескольких компонентов образца. Применение методов обработки многомерных данных (хемометрики) позволяет извлекать из таких неселективных перекрывающихся аналитических сигналов важную качественную и количественную информацию.

Сами хемометрические методы уходят корнями в линейную алгебру и известны достаточно давно. Так, например, один из наиболее популярных хемометрических алгоритмов — метод главных компонент, был предложен Пирсоном в 1901 году. Однако, практическое применение этих методов в химии было существенно ограничено, пока не появились доступные для широкого круга пользователей инструменты, позволяющие применять этот математический аппарат, не вдаваясь в тонкости линейной алгебры. Наш опыт преподавания хемометрики показывает, что даже без специализированной математической подготовки студенты и аспиранты химических дисциплин успешно осваивают и применяют на практике современные хемометрические алгоритмы.

Хорошей иллюстрацией роста популярности хемометрики является число публикаций по этой теме. Так, согласно данным библиографической базы Scopus™ в 1990 году было опубликовано всего 18 научных работ, авторы которых указали среди ключевых слов термин “chemometrics”. В 2000 году таких работ насчитывалось уже около полутора сотен, в 2010 – около четырехсот, а в 2020 уже почти тысяча.

Разумеется, такие критерии поиска не отражают общее количество научных исследований, связанных с тематикой, но четко обозначают тенденцию. Поэтому для современного химика знание основ этой дисциплины представляется нам очень важным. Объем учебной литературы по хемометрике в настоящее время поистине огромен, однако количество русскоязычных изданий по этой теме можно пересчитать по пальцам одной руки. Настоящая книга ставит своей задачей восполнить этот пробел. Она написана, в первую очередь, опираясь на опыт преподавания хемометрики ее авторами. Мы старались использовать простой и доступный язык изложения с множеством примеров и без погружения в слишком подробные математические детали. Совсем без математики в таком учебнике обойтись не получится, поэтому мы добавили отдельную главу, которая позволит освежить необходимые знания. Для того, чтобы читатель мог сразу после прочтения книги самостоятельно пользоваться рассматриваемыми алгоритмами, все главы сопровождаются примерами реализации алгоритмов в среде R — бесплатном и популярном инструменте для анализа и визуализации данных.

Тем, кто заинтересуется историей возникновения и развития хемометрики, мы с удовольствием рекомендуем несколько отличных статей, описывающих становление этой дисциплины. Две первые статьи (10.1002/cem.1180040503, 10.1002/cem.1180040604) написаны замечательными учеными Полем Гелади и Кимом Эсбенсеном и представляют собой сборник вполне неформальных интервью, взятых у нескольких исследователей, стоявших у истоков создания хемометрики в начале 70-х годов двадцатого века — на тот момент совершенно нового направления в химии.

Следующая работа: 10.1007/s00216-017-0517-1 написана «звездным» коллективом практикующих хемометриков и представляет собой краткий очерк по истории предмета, а продолжение этой статьи: 10.1007/s00216-018-1283-4 посвящено различным методическим аспектам применения хемометрики в аналитической химии.

Книга организована следующим образом:

В главе 1 рассматриваются важные базовые понятия, такие как структура, модальность и размерность данных, операции с матрицами и векторами, а также дается короткий обзор основных статистических методов, необходимых для понимания изложенного в учебнике материала.

Глава 2 подробно описывает теорию, алгоритм и примеры применения «рабочей лошадки» хемометрики — метода главных компонент. Этот метод очень часто используется для первого шага — разведочного анализа ваших данных. Кроме этого, он служит основой для ряда более продвинутых методов, например для многомерной регрессии и классификации.

Глава 3 посвящена предварительной обработке данных — методам, которые позволяют улучшить качество данных и подготовить их к последующему анализу. Особый акцент делается на улучшении качества спектральных данных.

В главе 4 рассматривается регрессионный анализ: от простых одномерных моделей до методов многомерной регрессии.

Глава 5 посвящена методам многомерной классификации. В главе рассмотрены основные теоретические представления о методах классификации, способы оценки качества классификационных моделей, и подробно описаны несколько популярных алгоритмов: метод ближайших соседей, линейный и квадратичный дискриминантный анализ, логистическая регрессия, SIMCA и PLS-DA.

Глава 6 рассказывает о задаче разделения кривых на примере метода независимых компонент.

Мы планируем работать над учебником и дальше, вносить исправления, улучшения и добавлять новые главы. Отслеживать изменения удобнее всего через GitHub репозиторий https://github.com/chemometrics-ru/book, где, помимо последней версии самого учебника, также собраны файлы с данными, используемыми в практических примерах и, позднее, будет доступна информация о новых версиях со списком изменений.

Желаем вам приятного чтения.

R — язык для анализа и визуализации данных

Работа с данными, даже самыми простыми, почти всегда требует математических расчетов. Например, если у нас есть несколько образцов воды из одного и того же водоема, и мы хотим знать концентрацию железа в нем, то придется, помимо измерений в лаборатории, провести еще и статистический анализ. Самым простым действием в этом случае будет расчет среднего значения, поскольку в каждом образце концентрация железа будет разной, а нам нужно одно число, которое будет характеризовать водоем целиком.

Такие расчеты можно проводить в различных программах, самая простая из них, которую вы, наверняка, хорошо знаете, — табличный процессор *Microsoft Excel* или его аналоги, например, *Google Sheets* или *LibreOffice Calc*. Табличные процессоры особенно удобны для сбора небольшого объема данных и простого статистического анализа, как в примере выше.

Однако, если данных очень много (например, таблицы с тысячами строк и столбцов), то анализ таких данных в табличных процессорах — довольно трудоемкое дело. Особенно, если речь идет не о простых вычислениях, как, например, расчет среднего значения, а об использовании более сложных методов, которые мы собираемся обсудить в этой книге. Для этого существует множество специализированных программ с графическим интерфейсом, где вы можете провести нужные расчеты одним щелчком мыши. Такие программы хороши в первую очередь для продвинутых пользователей, которые понимают, что последует за этим щелчком, какие математические модели будут использованы для анализа и как интерпретировать полученный результат.

Наконец, есть и третий выбор — работать с данными с помощью языков программирования, специально разработанных для этой цели. С одной стороны, такие языки достаточно просты для освоения, и для работы с ними не требуется глубоких знаний о программировании. С другой стороны, эти языки дают возможность использовать большое количество методов, включая самые новые, которые были недавно опубликованы в научных журналах, но еще не появились в специализированном программном обеспечении.

Работу с такими языками можно сравнить с конструктором Лего — вместо того, чтобы купить готовую модель машины или самолета вы собираете их из небольших блоков. Основной недостаток таких языков в том, что требуется дополнительное время на изучение основ программирования в целом, и на освоение синтаксиса и правил выбранного языка, в частности. Но результат стоит того — ваши возможности больше не будут ограничены тем, что заложили в программу разработчики, вы без труда сможете расширять набор методов из вашего арсенала.

Наиболее популярными из таких языков, предназначенных для описания, визуализации и анализа данных, являются R и Python. Для решения задач хемометрики R представляется более привлекательным, поскольку он существенно проще в освоении и для этого языка свободно доступны многочисленные пакеты, реализующие различные хемометрические алгоритмы. Поэтому мы выбрали R в качестве инструмента для всех практических примеров в этом учебнике.

Для того, чтобы начать использовать R, откройте в браузере официальный сайт R https://www.r-project.org, скачайте и установите последнюю версию языка на свой компьютер. Мы также рекомендуем скачать и установить программу RStudio Desktop. RStudio является самой популярной и дружественной к пользователю средой разработки на R, и сильно упростит его использование. Программа, как и сам язык, распространяются бесплатно и имеют версию для всех современных операционных систем. Полезно будет освоить руководство, или простой видео-курс для работы с R для начинающих. Таких ресурсов сейчас очень много в сети и на русском, и на английском языке, думаем вы без труда найдете несколько подходящих для вас. Например, вы можете воспользоваться руководствами доступными на официальном сайте.

Happy coding!

1 Представление, визуализация и статистическое описание данных

1.1 Измерения, наблюдения и переменные

Термин *данные* имеет очень широкую трактовку, однако, почти всегда он означает некоторый набор информации об интересующем нас объекте, или явлении. Такая информация может быть довольно разнородной и не обязательно числовой, к примеру, если кто-то говорит вам, что ваши новые соседи — это молодая семья без детей, которые ездят на красной Ладе Приора — это тоже данные.

В этой книге мы будем иметь дело с данными, во-первых, структурированными, а во-вторых, полученными в результате измерения, или наблюдения какого-то определенного свойства изучаемого объекта, или явления. Например, если вы измерите кислотность пяти образцов воды, вы получите пять различных чисел, которые можно организовать в виде простой таблицы с одним столбцом/колонкой, как это показано ниже:

pH
6.8
7.0
6.6
7.9
8.1

Если для этих же образцов измерить еще и содержание ионов железа, то таких столбцов с числами в таблице будет уже два. Это и есть простой пример структурированных данных, столбцы в этой таблице представляют собой измеряемые свойства, а строки соответствуют отдельным образцам.

pH	Fe^{3+}, мг/л
6.8	0.32
7.0	0.28
6.6	0.21

pH	Fe^{3+}, мг/л
7.9	0.35
8.1	0.33

Структуры данных будут подробно рассмотрены в следующем разделе, а пока поговорим об отдельных свойствах. Подразумевается, что изучаемое свойство не является постоянным, поэтому часто его называют *переменная* (англ. *variable*). Если вернуться к примеру с образцами воды, очень маловероятно что содержание железа во всех образцах воды будет одинаково, даже если все образцы взяты из одного источника.

1.1.1 Типы переменных

В зависимости от того, каким образом измерения считывают или записывают, различают *качественные* и *количественные* переменные. Количественные (англ. *quantitative*) переменные позволяют представить измеренное значение в виде некоторой величины, которую можно выразить в виде обычного числа. Например, концентрация, измеренная в граммах на единицу объема, или число электронов в отдельном атоме.

Количественные переменные могут быть *непрерывными* и *дискретными*. Дискретные переменные, как правило, имеют ограниченный, часто целочисленный набор значений. К примеру, число машин, которое в среднестатистической семье варьируется обычно от 0 до 2. Непрерывные переменные теоретически могут принимать бесконечное число значений, однако, в реальности оно всегда ограничено точностью измерений. Например, если измерять рост человека в сантиметрах, то, строго говоря, полученная переменная будет дискретной, так как все измеренные значения будут целочисленными. Но, так как в этом случае число возможных значений будет достаточно велико (если мы говорим о взрослых людях с ростом от 141 до 210 см, то число возможных значений равно 70), то часто такие переменные считают непрерывными.

Качественные (англ. *qualitative*) переменные описывают свойство некоторым фиксированным набором нечисловых характеристик, называемых *категориями*. Например, все химические элементы можно разделить на несколько категорий, которые соответствуют их химическим свойствам: “щелочные металлы”, “щелочноземельные металлы”, “металлоиды” и так далее. Кислотность раствора можно тоже задать с помощью категорий: “кислый”, “нейтральный” и “щелочной”. Таким образом, кислотность можно измерить с помощью количественной переменной, получив числовое значение водородного показателя, pH, а можно это сделать с помощью качественной переменной, присвоив раствору одну из трех категорий. В последнем случае в столбце таблицы с данными будет записано название соответствующей категории, а не число.

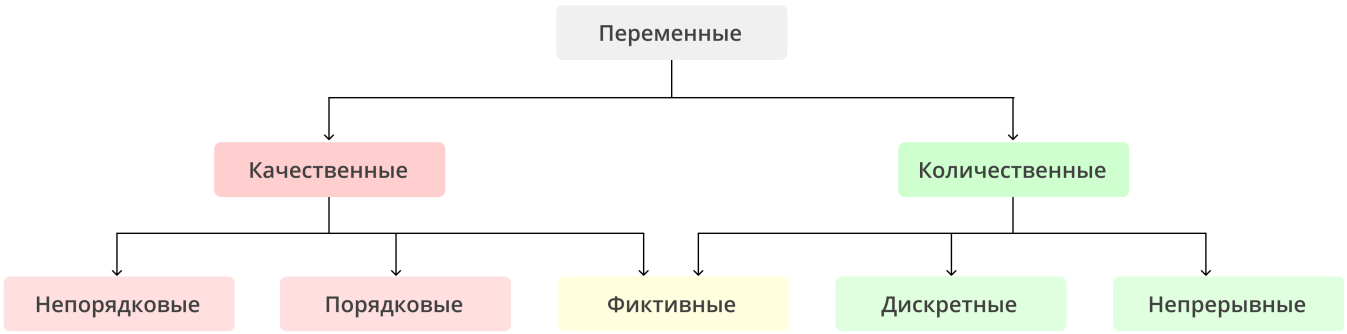


Рис. 1.1. Типы переменных.

В некоторых случаях категории можно упорядочить определенным образом. Например, кислотность имеет порядок от “кислый” до “щелочной” — от наиболее кислого до наименее. Или, скажем, температуру можно описать с помощью категорий “холодный”, “теплый” и “горячий”, которые также имеют определенный порядок. Качественные переменные, для которых порядок категорий имеет смысл, также называются *порядковые* (англ. *ordinal*).

Часто бывает удобно (а иногда и необходимо) представить качественные значения в числовом виде. Существует распространённый способ сделать это с помощью так называемых *фиктивных* (англ. *dummy*) переменных. В этом случае для каждой категории создаётся отдельная фиктивная переменная, которая равна 1, в случае если объект попадает в эту категорию, и 0 — если нет (иногда для этой цели используют значения 1 и -1).

Если вернуться к примеру с кислотностью растворов, то фиктивных переменных будет три — по одной на каждую категорию. Таблица ниже показывает, как описать кислотность пяти растворов используя качественную переменную (первый столбец) либо три фиктивных переменных (три последних столбца).

Категория	Кислый	Нейтральный	Щелочной
“Кислый”	1	0	0
“Нейтральный”	0	1	0
“Кислый”	1	0	0
“Щелочной”	0	0	1
“Нейтральный	0	1	0

Такой способ позволяет использовать качественные переменные для математических расчетов, о чем мы будем рассказывать позже в этой книге.

Рисунок 1.1 показывает иерархию типов переменных схематически.

1.1.2 Независимые и зависимые переменные

Часто, особенно если речь идёт об экспериментальных данных, переменные можно также разделить на две группы — *независимые* и *зависимые*. Независимыми переменными обычно называют переменные, описывающие состояние эксперимента, которые не измеряются, а задаются экспериментатором напрямую. Они не зависят от значений других переменных (отсюда и название). К примеру, если можно напрямую задать требуемую температуру жидкости, или концентрацию катализатора в химической реакции, то обе переменные (температура и концентрация катализатора) будут независимыми.

Зависимые переменные, это те, которые могут быть только измерены, напрямую их изменить невозможно, или нецелесообразно. Считается, что такие переменные можно поменять косвенно, за счёт изменения независимых переменных. Т.е. предполагается, что их значения зависят от других переменных. Если продолжить пример с химической реакцией, то зависимыми переменными могут считаться скорость реакции, или выход ее конечного продукта. Очевидно, что если изменить температуру реакции (независимая переменная), то это повлияет на ее скорость (зависимая переменная).

> *Если вы собираетесь сегодня вечером приготовить попкорн в микроволновке, то вы, сами того не осознавая, столкнётесь и с разными типами переменных, и с упомянутыми выше группами. Понятно, что невозможно напрямую изменить ни вкус готового продукта, ни число «раскрывшихся» зёрен — это зависимые переменные. Но вы можете варьировать время, мощность излучения, а также выбрать производителя, или бренд пакетов с зернами. Очевидно, время можно считать непрерывной количественной переменной, мощность — дискретной, а бренд — это качественная переменная.*

1.1.3 Временные ряды

Временные ряды представляют собой особый вид переменных, так как они отражают изменение свойства одного объекта со временем, тогда как обычные переменные содержат значения для множества объектов или явлений.

Приведем простой пример. Предположим, что вы хотите знать содержание железа в воде 15 небольших озер, расположенных в радиусе нескольких километров от промышленного производства. В этом случае вам необходимо взять пробы из каждого озера и после необходимых измерений и статистического анализа вы получите 15 значений — среднюю концентрацию железа, найденного в пробах взятых из каждого озера. Это пример обычного набора данных с одной переменной и 15 различными объектами исследования.

Допустим, оказалось, что концентрация в одном из озер, расположенных наиболее близко к производству, намного выше допустимой. Чтобы это исправить, на производстве полностью поменяли систему очистки. Однако насколько сильным будет эффект от замены? Для этого необходимо контролировать состояние озера в течение некоторого времени, например, брать пробы из этого озера раз в месяц в течение 15 месяцев.

В этом случае мы тоже получим 15 чисел в результате, но они будут соответствовать одному свойству (концентрации железа), измеренного для одного объекта (данного озера) в разные временные интервалы. Такие данные и будут представлять собой временной ряд.

Основная разница между обычными переменными и временными рядами заключается в том, что значения временного ряда обычно зависят друг от друга, причём эта зависимость различна для разных временных масштабов и участков. Например, если измерять концентрацию какого-то компонента химической реакции в течение некоторого времени, то можно по этим данным вычислить скорость этой реакции, так как концентрация, которую мы наблюдаем сейчас, зависит от того, какая концентрация была минутой ранее.

1.1.4 Ввод и простейшие операции с данными в R

Как мы уже написали в начале книги, все примеры в ней будут сопровождаться кодом, написанным на R — специальном языке программирования, созданным для анализа и визуализации данных. В конце первой главы мы написали о том, какие программы понадобятся, чтобы начать работать с R на вашем компьютере, и как их установить. Так что, начиная с этого момента, мы предполагаем, что вы установили все на свой компьютер и прочитали как минимум одно руководство о том, как с этим программным обеспечением работать.

Код, который мы приводим в этой книге, можно набирать непосредственно в консоли, но лучше всего создать отдельный скрипт, чтобы проще было писать и выполнять код, а также вставлять туда свои комментарии. Например, вы можете создавать отдельные скрипты для каждого раздела книги.

Итак, поговорим о том, как представить и визуализировать простейшие данные в R. Основной способ представления данных в R — *атомарные* (англ. *atomic*) векторы. Атомарный вектор, по сути, это обычная последовательность значений — числовых, текстовых, либо логических. Каждое значение имеет своё место в этой последовательности, называемое индексом. Следующий код показывает, как создать такие векторы:

```
weight <- c(6.94, 9.01, 22.99, 24.31)
name <- c("Li", "Be", "Na", "Mg")
earth.metal <- c(FALSE, TRUE, FALSE, TRUE)
```

Как можно видеть, векторы создаются с помощью функции `c()` которая принимает отдельные значения в виде аргументов и объединяет их в единой целое. Оператор `<-`, это основной оператор присваивания, он позволяет создавать любые объекты в R, от простых векторов до таблиц и функций. Такие объекты в R называются переменными, так как пользователь в любое время может поменять значение этого объекта. Любой объект/переменная должен иметь уникальное имя, состоящее из латинских букв, цифр, точки и символа подчеркивания. Имя используется впоследствии для обращения к объекту, или для изменения его значения.

Текстовые значения должны быть заключены в одинарные или двойные кавычки. Если написать текст без кавычек, R будет пытаться найти объект с соответствующим именем. Например `weight` означает объект с таким именем (например вектор значений, который мы создали в блоке кода выше), а `"weight"` — это слово, которое можно использовать в качестве текстового значения.

Значения в векторе могут также иметь имена. По сути, это означает, что вектор будет иметь специальный атрибут с последовательностью текстовых значений для каждого элемента. Имена значениям можно задать сразу при создании вектора, либо после, с помощью функции `names()`. Несколько примеров показаны в блоке ниже.

```
# создать вектор с именованными значениями
weight <- c("Li" = 6.94, "Be" = 9.01, "Na" = 22.99, "Mg" = 24.31)
show(weight)
```

```
   Li    Be    Na    Mg
 6.94  9.01 22.99 24.31
```

```
# создать вектор и присвоить имена отдельной командой
earth.metal <- c(FALSE, TRUE, FALSE, TRUE)
names(earth.metal) <- c("Li", "Be", "Na", "Mg")
show(earth.metal)
```

```
   Li    Be    Na    Mg
FALSE  TRUE FALSE  TRUE
```

Как можно заметить, для вывода содержимого переменной в примерах выше используется функция `show()`. На самом деле, если вы работаете в консоли, то можно просто написать имя переменной и нажать на клавишу Enter чтобы показать ее содержимое. Функция `show()`, как и ее аналог `print()`, обычно используются, если нужно вывести результат на экран внутри отдельного скрипта, или функции.

Извлечение данных

Для того, чтобы извлечь часть значений из вектора используются квадратные скобки, как это показано в блоке с кодом ниже. Одинарные скобки позволяют извлечь любое количество элементов вектора, тогда как двойные скобки возвращают только один элемент и при попытке извлечь несколько элементов R покажет сообщение об ошибке.

Внутри скобок нужно указать индекс элемента, который следует извлечь, либо его имя (если имена заданы). Если требуется извлечь несколько элементов, то нужно собрать их индексы, или имена в вектор с помощью уже известной вам функции `c()`.

```
# извлечь одно значение из вектора
li.weight <- weight[[1]]
li.weight <- weight[["Li"]]

# извлечь несколько значений из вектора
em.weight <- weight[c(2, 4)]
em.weight <- weight[c("Be", "Mg")]
```

Существует также возможность извлекать значения из вектора, пользуясь логическими значениями `TRUE` и `FALSE`. Пример ниже показывает, как извлечь значения массы только для щелочноземельных металлов. Для этого используется вектор `earth.metal` который имеет значение `TRUE` для щелочноземельных металлов (в данном примере Be, Mg) и `FALSE` для остальных.

```
em.weight <- weight[earth.metal]
show(em.weight)
```

```
   Be    Mg
 9.01 24.31
```

Математические операции

Числовые векторы можно использовать для математических операций, в этом случае все операции будут применены поэлементно, т.е. результатом всегда будет вектор с таким же количеством элементов, что и исходный. В этом случае важно, чтобы все векторы, используемые в операции, имели одинаковую длину. Ну и конечно, операции могут производится между вектором и одним значением (скаляром).

Пример ниже показывает, как вычислить давление внутри закрытого сосуда объемом 2 литра с двумя молями идеального газа для заданного набора температур. В результате `P` — это вектор с четырьмя значениями, соответствующими четырём значениям температуры из вектора `Temp`. Такое имя для переменной используется, так как строчная `T` в R зарезервирована для короткого значения `TRUE`.

```
n <- 2
Rc <- 8.31
V <- 2
```

```r
Temp <- c(0, 10, 30, 50)
P <- n * Rc * (Temp + 273) / (V / 1000)
show(P)
```

```
[1] 2268630 2351730 2517930 2684130
```

R так же позволяет использовать все основные математические функции, которые также применяются поэлементно, например `log()`, `exp()`, `sqrt()` и т.п.

Представление количественных данных

В R существует специальный тип переменной, позволяющий удобно работать с количественными данными — *фактор*. По сути фактор — это обычный вектор с целочисленными значениями, но, во-первых, со специальным атрибутом `labels`, а во-вторых, с дополнительным набором методов недоступных обычным числовым векторам.

Атрибут `labels` — это вектор с именами (метками) категорий. Каждая метка имеет свой индекс, а индексы как раз хранятся в самом векторе. Код в блоке ниже показывает, как создать фактор, а также манипулировать с его основными частями - индексами и метками категорий.

```r
# создать фактор с 5 значениями из трех категорий
x <- factor(c(1, 1, 2, 1, 3), labels = c("щелочной", "нейтральный", "кислый"))

# показать значения фактора
show(x)
```

```
[1] щелочной    щелочной    нейтральный щелочной    кислый
Levels: щелочной нейтральный кислый
```

```r
# показать индексы
show(as.numeric(x))
```

```
[1] 1 1 2 1 3
```

```r
# показать вектор с метками категорий
show(levels(x))
```

```
[1] "щелочной"    "нейтральный" "кислый"
```

Фактор можно также создать из вектора с текстовыми значениями, как показано ниже:

```
# создать вектор с текстовыми значениями
x <- c("щелочной", "нейтральный", "кислый", "нейтральный", "щелочной")
show(x)
```

```
[1] "щелочной"    "нейтральный" "кислый"      "нейтральный" "щелочной"
```

```
# превратить его в фактор
y <- as.factor(x)
show(y)
```

```
[1] щелочной    нейтральный кислый      нейтральный щелочной
Levels: кислый нейтральный щелочной
```

1.1.5 Графическое представление числовых значений

Для того, чтобы отобразить значения из одного атомарного вектора графически, можно воспользоваться диаграммной лентой (англ. *strip chart*). Это простой график с одной горизонтальной осью, на котором ставятся полоски, или точки, соответствующие каждому значению. Код ниже показывает пример создания такого графика для значений из вектора `weight`, созданного нами ранее:

```
stripchart(weight)
```

У любого графического метода в R существует большое количество параметров, позволяющих изменять практически все составляющие графика. В примере ниже мы меняем цвет (за это отвечает параметр с именем `col`) и тип (`pch`) маркеров, а также добавляем название координатной оси (с помощью параметра `xlab`) и самого графика (за это отвечает параметр `main`).

```
stripchart(
   weight,
   col = "blue",
   pch = 16,
   main = "Атомная масса элементов",
   xlab = "Стандартная атомная масса"
)
```

Полный список параметров и описание того, какие значения конкретный параметр может принимать, можно найти в справке для конкретной функции (например, выполнив команду `?stripchart` в консоли). Забегая немного вперед, нужно сказать, что вы можете использовать эти же параметры и для других графических методов, что облегчает их изучение.

Функция `text()` позволяет добавлять любой текст на уже созданный график, указав координаты точки (x, y) и, собственно, сам текст. Так как все три аргумента могут быть в виде векторов, ее удобно использовать для добавления меток к точкам на графиках, как показано ниже.

```
text(weight, 1, names(weight), pos = 3)
```

Здесь мы используем названия значений вектора `weights` в качестве меток. Второй аргумент, 1, это положение меток по оси *y*. Формально этой оси нет на графике, но, так как график двумерный, она на самом деле существует, просто все точки имеют одну и туже координату по этой оси — 1.

Рисунок 1.2 показывает результат выполнения всех трех блоков с кодом, если запустить их последовательно.

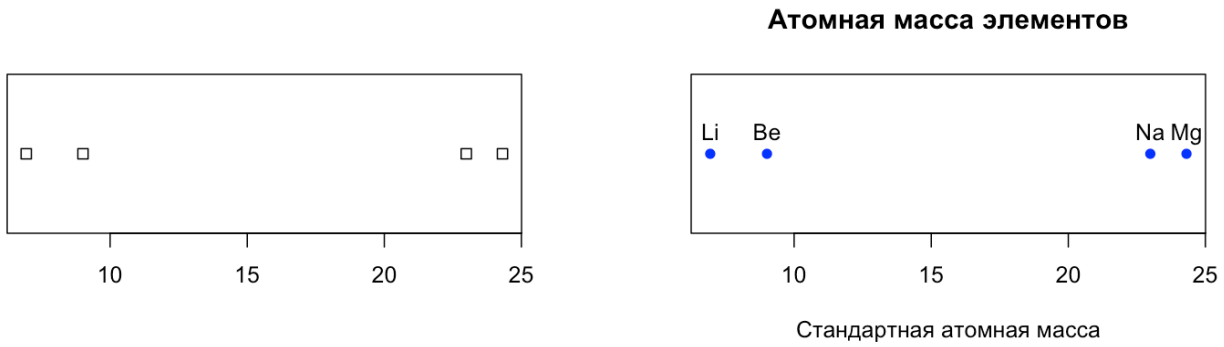


Рис. 1.2. Результат выполнения блоков с кодом

1.2 Модальность и размерность данных

В предыдущем разделе мы обсуждали, как представить измерения в виде последовательности значений, подразумевая, что эти значения относятся:

1. к одной переменной (изучаемому свойству);
2. к нескольким разным объектам или явлениям (выборке);
3. к одному объекту, но измерениям сделанным в разные отрезки времени.

Однако, в реальности часто необходимо измерять и анализировать большое число переменных одновременно. Например, при спектроскопическом анализе одно измерение содержит значения для сотни, а иногда и тысячи переменных, — результат поглощения или отражения объектом электромагнитного

излучения на разных длинах волн. Данные могут иметь и более сложную структуру, как будет показано ниже.

В английском языке для описания структуры данных используются два термина — *dimension* и *way*. В этой книге мы будем использовать для них русскоязычные термины *размерность* и *модальность* соответственно. Рассмотрим сначала, что такое модальность.

1.2.1 Модальность данных

Под *модальностью* здесь мы будем понимать число независимых источников (причин, направлений) вариации данных — мод. Например, значения одной переменной могут изменяться от одного объекта к другому, так как не бывает двух объектов с абсолютно идентичными свойствами. Соответственно, вектор значений такой переменной будет иметь модальность равную единице (англ. *one-way data*), так как в этом случае в данных есть единственный источник, или причина для вариации значений — наличие разных объектов.

Если для одной и той же выборки объектов измерять несколько различных свойств, то в этом случае мы говорим о двухмодальных (англ. *two-way*) данных. Помимо наличия разных объектов, у нас появился еще один источник вариации — разные переменные (свойства). Как мы уже говорили, такие данные обычно представляются в виде таблицы, в которой строки соответствуют отдельным измерениям, а столбцы — свойствам.

Данные могут иметь модальность и выше двух. Скажем, если двухмодальные данные измерить несколько раз, в различные промежутки времени, то такие измерения можно представить в виде трехмерного массива (куба) и модальность такого набора данных будет равна трём (англ. *three-way data*), так как к уже имеющимся двум модам добавляется еще одна — время.

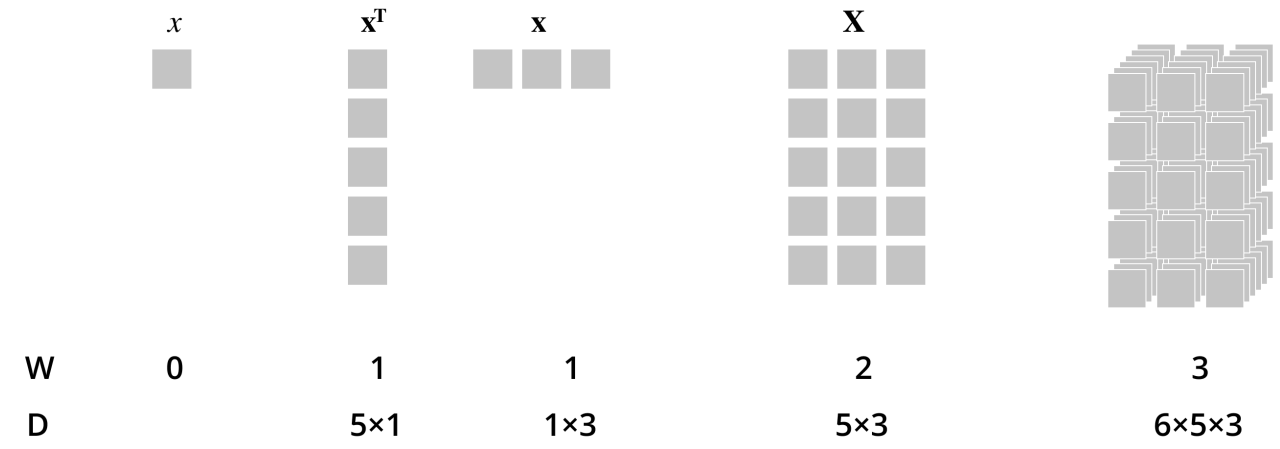


Рис. 1.3. Схематическое представление данных разной модальности (W), и размерности (D).

Рисунок 1.3 схематически показывает разницу между данными разной модальности (для ее обозначения используется буква W) и то, как они обозначаются в математике. Каждый квадрат на этом рисунке представляет собой одно значение, например, какое-то число. Само по себе единственное значение всегда постоянно, поэтому модальность таких данных равна нулю. В математике отдельное число, имеющее нулевую модальность, называют *скаляром.*

В аналитической химии можно часто встретить данные с высоким числом модальности. Так, например, инфракрасные спектры, измеренные для нескольких объектов, будут являться двухмодальными данными. В этом случае мы говорим о поглощении инфракрасного излучения, которое меняется и в зависимости от объектов и в зависимости от длины волны (т.е. две моды). Пример такого представления показан на рисунке 1.4 — в таблице данных каждая строка соответствует отдельному спектру, а каждый столбец — спектральным значениям (поглощению излучения), измеренным для конкретной длины волны.

Результаты измерений, сделанных с помощью газовой хромато-масс-спектрометрии, являются трехмодальными данными. Каждому объекту в этом случае соответствует вектор компонент, получаемый с помощью хроматографа, а каждой компоненте из этого вектора соответствует масс-спектр. Другими словами, каждое число в таких данных характеризует комбинацию трех мод — объект, на котором проводились измерения, время регистрации в хроматографической колонке и отношение массы к заряду. Использование двумерного хроматографа (GCxGC-MS) позволяет получить данные с модальностью четыре.

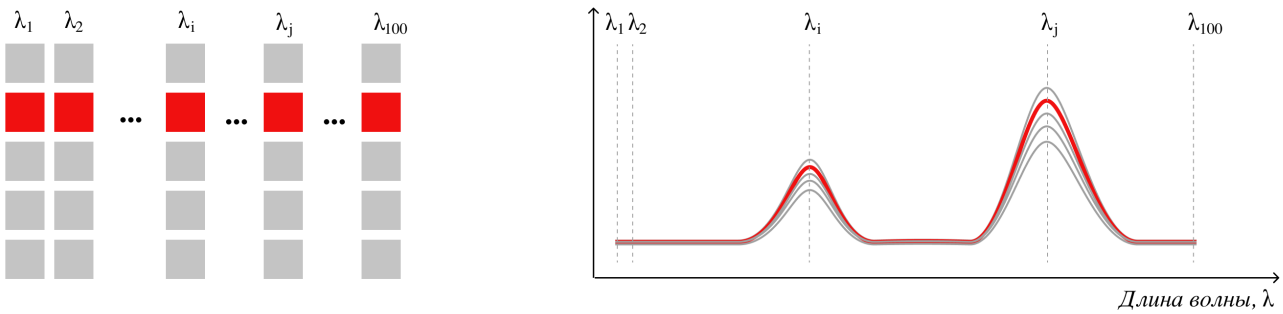


Рис. 1.4. Пример представления спектральных данных в виде таблицы: каждый спектр — отдельная строка, а спектральные значения измеренные для конкретной длины волны — отдельный столбец.

В этой книге большую часть времени мы будем иметь дело с одномодальными и двухмодальными данными. Методы для анализа трехмодальных данных также существуют и являются важной частью хемометрики, но выходят за рамки этого учебника.

1.2.2 Размерность и геометрическое представление данных

Под *размерностью* данных будем понимать число элементов для каждой моды. Скажем, для двухмодальных данных размерность будет означать число объектов, или индивидуальных измерений, и число переменных. Часто, когда речь идёт именно о двухмодальных данных, термин “размерность” используется только для обозначения числа переменных. Так, *трехмерные данные* в первую очередь означает данные для некоторой выборки объектов с тремя измеренными свойствами (переменными).

Такая терминология имеет прямое отношение к геометрическому представлению данных. Измеренные значения можно представить в виде облака точек в пространстве переменных. Другими словами, каждая переменная (свойство) представляется в виде координатной оси, а измерение — в виде точки на пересечении измеренных значений всех переменных. Таким образом, если построить подобный график для двух переменных, он будет двумерным, а если для трёх — то трехмерным.

Рассмотрим следующую таблицу, в которой собраны результаты химического анализа 10 проб воды, собранных с разных участков озера. Каждый столбец содержит концентрацию ионов в образцах (хлора, натрия и калия):

Cl^-, мг/л	Na^+, мг/л	K^+, мг/л
50.0	97.4	14.8
41.0	87.0	13.8
47.6	92.0	13.9
59.9	120.4	14.2
53.4	108.5	13.6
54.4	110.1	15.5
44.0	78.9	15.1
49.4	91.3	13.5
46.4	86.2	11.1
58.5	107.5	16.9

Здесь мы имеем классический образец двухмодальных данных с тремя переменными и десятью измерениями. Эти данные можно представить в виде одного трехмерного, или трёх двумерных графиков рассеяния (англ. *scatter plots*). Примеры двумерных графиков для двух различных пар переменных показаны на рисунке 1.5.

Графики рассеяния позволяют делать выводы об отношениях между отдельными измерениями. Например& две точки, расположенные близко друг к другу на таком графике, будут соответствовать двум измерениями с очень близкими значениями переменных, используемых в качестве осей для графика. Т.е. можно сказать, что соответствующие образцы имеют схожих химический состав, как, например,

точки 5 и 6 на левом графике рисунка 1.5. Если свериться с таблицей, то можно заметить, что, на самом деле, концентрации ионов хлора и натрия для этих двух измерений достаточно близки (см. значения соответствующих столбцов и строк 5 и 6 таблицы).

Однако, тут мы говорим, конечно, не о полном составе, а лишь о двух химических компонентах, которые мы использовали в качестве осей. Например, если посмотреть на график справа, то можно увидеть что эти точки расположены достаточно далеко друг от друга, так как концентрация ионов калия, которую мы использовали в качестве оси ординат на этом графике, довольно сильно отличается. Если имеется несколько измерений со схожими значениями, то на таком графике их можно заметить в виде плотного облака точек, или кластера.

Подобные графики позволяют так же судить о взаимоотношениях между переменными. Например, на левом графике можно заметить, что все точки расположены близко к некоторой воображаемой прямой, направленной из левого нижнего угла в правый верхний. Это связано с тем, что в наших измерениях имеется зависимость между концентрацией ионов хлора и натрия — образцы с высоким содержанием хлора имеют также высокое содержание натрия, и наоборот. Такой визуальный эффект на графике часто называют *трендом*.

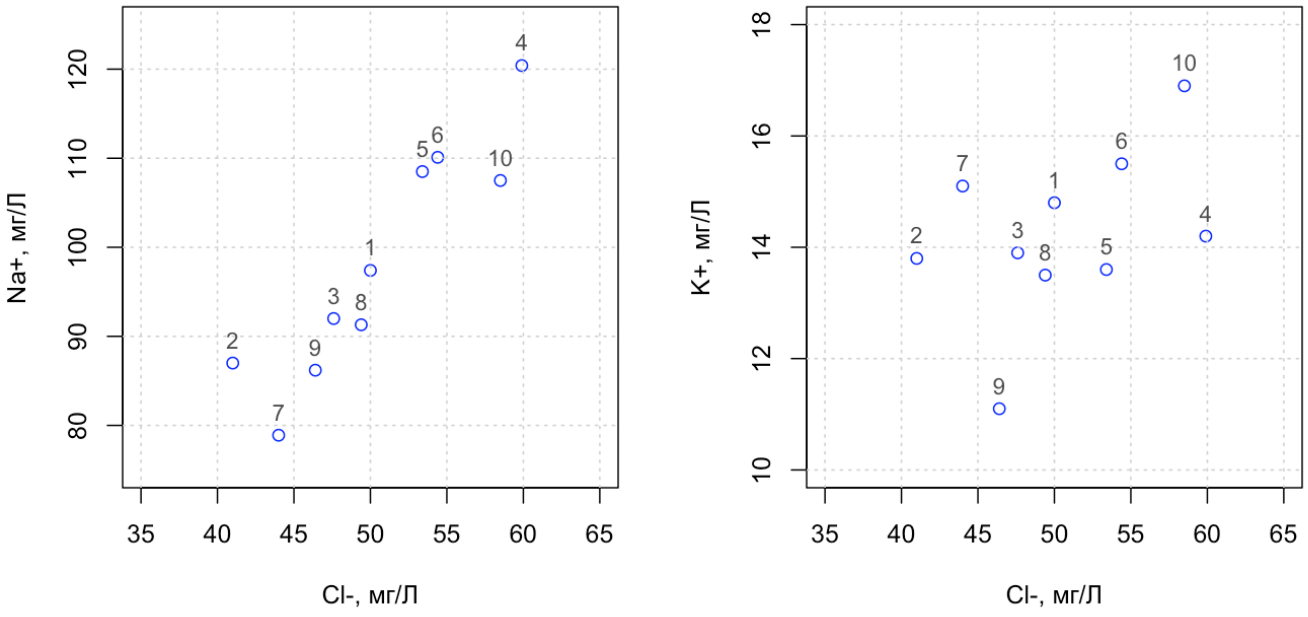


Рис. 1.5. Представление данных в пространстве переменных.

Однако, как быть, если число переменных не две, или три, а, скажем, несколько десятков или сотен? Так как плоский график рассеяния можно построить только для двух переменных, то в этом случае придется

перебрать все возможные пары. Для трех переменных таких пар (а значит и графиков) будет три, для пяти — десять, а для 10 переменных число возможных пар равно 45. Изучить даже 45 графиков дело непростое, а что делать, если переменных сотни, или даже тысячи? Чтобы разрешить эту дилемму и нужны методы анализа многомерных данных, которым посвящена эта книга. Однако, не будем забегать вперёд и поговорим еще немного о простых примерах с небольшим количеством переменных.

1.2.3 Представление и визуализация двухмодальных данных в R

Для представления двухмодальных данных в R существует достаточно большой выбор способов — матрицы, фреймы, таблицы, и так называемые тиблы (англ. *tibbles*). Мы ограничимся фреймами и матрицами, остальные способы представления данных, а так же их плюсы и минусы оставим читателю для самостоятельного изучения.

Фрейм (англ. *data frame*) — это набор (англ. *list*) отдельных атомарных векторов. Т.е. по сути фрейм — это некоторая обертка, позволяющая хранить и манипулировать отдельными векторами, как единым набором данных. Для того, чтобы создать фрейм, нужно воспользоваться функцией `data.frame()` и указать список векторов в качестве аргументов. Очень важно, чтобы все векторы имели одинаковую длину (число значений) — это одно из ограничений фреймов. Блок кода ниже показывает как создать фрейм для таблицы с содержанием ионов в воде, которую мы использовали ранее в этом разделе.

```
d <- data.frame(
   Cl = c(50.0, 41.0, 47.6, 59.9, 53.4, 54.4, 44.0, 49.4, 46.4, 58.5),
   Na = c(97.4, 87.0, 92.0, 120.4, 108.5, 110.1, 78.9, 91.3, 86.2, 107.5),
   K = c(14.8, 13.8, 13.9, 14.2, 13.6, 15.5, 15.1, 13.5, 11.1, 16.9)
)
```

Если необходимо сделать обратную операцию — извлечь значения, принадлежащие конкретному столбцу, необходимо указать имя переменной с фреймом и имя столбца разделив их знаком доллара, как показано ниже.

```
# посчитать среднюю концентрацию хлорид-ионов
mCl = mean(d$Cl)
show(mCl)
```

```
[1] 50.46
```

Если же нужно извлечь одно значение из конкретного столбца и строки, то можно воспользоваться квадратными скобками, как мы это делали для атомарных векторов, но нужно указать два индекса — первый для строки и второй для столбца:

```
show(d[1, 1])
```

```
[1] 50
```

Чтение данных из файлов

Вводить вручную данные в R довольно утомительное занятие. К счастью, этого можно избежать и читать значения напрямую из файлов. R предоставляет такую возможность для любых текстовых файлов, в которых значения как-то разделены — пробелами, запятыми, точками с запятой и так далее.

Для этого есть ряд методов, основным из которых является `read.delim()`. На его основе существуют также и более узко специализированные методы, например `read.csv()` который позволяет читать данные из файла, где значения разделены запятыми (CSV означает *comma separated values*).

Если же нужно читать данные из бинарных файлов, то придется воспользоваться различными библиотеками, которые в R называются пакетами (англ. *packages*). Скажем для чтения данных из таблиц Microsoft Excel есть замечательный пакет `readxl`.

Графическое представление данных

В предыдущем разделе мы уже обсуждали, как представить данные графически с помощью метода `stripchart()`, однако, он применим только к отдельным векторам. Чтобы построить полноценный двумерный график нужно воспользоваться другим методом — `plot()`. По синтаксису он очень похож на `stripchart()` и "понимает" все аргументы последнего. Основная разница в том, что в `plot()` можно задавать значения для обеих координат, x и y. Пример ниже показывает как построить первый график изображенный на рисунке 1.5, используя фрейм со значениями `d`, определенный нами ранее (важно убедиться, что вы на самом деле создали этот фрейм перед запуском этого кода).

```
plot(d$Cl, d$Na, pch = 1, col = "blue", xlab = "Cl-, mg/L", ylab = "Na+, mg/L")
```

Существует множество параметров, которые дают дополнительные возможности для построения графиков. Так, параметры `xlim` и `ylim` позволяют задавать масштабы соответствующих осей, для этого нужно указать наименьшее и наибольшее число для каждой оси в виде вектора. Используйте справку (например для функции `plot()` ее можно получить с помощью команды `?plot`), чтобы получить полные сведения о возможностях того или иного метода, либо ищите описание и примеры с помощью поисковых систем в Интернете.

Кроме изменения параметров графика, на уже созданный график можно добавлять большое количество разных элементов. Мы уже показали, как добавить текстовые элементы, а вот список методов, позволяющих

добавлять различные графические элементы: `points()`, `lines()`, `rect()`, `segments()`, `polygon()`, `arrows()`, `abline()`, `grid()`. Подробное описание и список аргументов вы также сможете найти в справке.

Построчные и поколоночные операции

В R есть несколько методов позволяющих эффективно работать с двухмодальными данными, например `apply()`, `lapply()`, `sapply()`, `aggregate()` и другие. Мы рассмотрим подробно только работу первого метода из списка и рекомендуем изучить остальные методы самостоятельно.

Представим, что у нас есть фрейм с данными, состоящими из нескольких переменных, или нескольких десятков переменных (столбцов), и требуется посчитать различные статистики для каждого из них. Или же сделать тоже самое, но для строк. В этом случае и выручит функция `apply()`. В блоке кода ниже показано как посчитать среднюю концентрацию для каждого иона, а также суммарную концентрацию всех трех ионов для каждого образца.

```
# посчитать среднее для каждого столбца фрейма d
m <- apply(d, 2, mean)
show(m)
```

```
   Cl    Na     K 
50.46 97.93 14.24 
```

```
# посчитать сумму значений в каждой строке фрейма d
s <- apply(d, 1, sum)
show(s)
```

```
 [1] 162.2 141.8 153.5 194.5 175.5 180.0 138.0 154.2 143.7 182.9
```

Как вы скорее всего уже догадались, первый аргумент функции `apply()` — это собственно фрейм с данными, второй аргумент показывает с какой модой нужно работать (1 — строки, 2 — столбцы), а третий, это функция, которая для каждого набора значений будет возвращать определенную статистику, например среднее значение или сумму. Т.е. команду `apply(d, 1, sum)` R воспримет как: “посчитать сумму каждой строки фрейма с данными `d`”.

Функция `apply()` может работать и с пользовательскими функциями, при этом необязательно, чтобы она возвращала только одно значение. Важно, чтобы функция возвращала значения одного типа и размера, так как результат будет объединен в матрицу (подробнее про матрицы мы поговорим в следующем разделе). Код ниже показывает, как кроме среднего вычислить еще и наименьшее, и наибольшее значения.

```r
mystat <- function(x) {
   c(min(x), mean(x), max(x))
}

s <- apply(d, 2, mystat)
rownames(s) <- c("Min", "Mean", "Max")
show(s)
```

```
        Cl     Na     K
Min  41.00  78.90 11.10
Mean 50.46  97.93 14.24
Max  59.90 120.40 16.90
```

Обратите внимание на то, что вначале мы определяем пользовательскую функцию, которая принимает вектор значений `x` в качестве аргумента, а возвращает вектор с тремя статистиками, посчитанными для `x`. Мы также пользуемся методом `rownames()`, который позволяет задавать метки для строк матрицы или фрейма. В результате получается быстрый способ посчитать статистики для всего набора данных и вывести результат в понятной форме.

1.3 Представление данных в виде векторов и матриц

Если все измеренные значения имеют числовое представление, то вместо таблицы их также можно представить в виде *матрицы*. Основное различие между таблицей и матрицей в том, что таблица по сути представляет набор отдельных столбцов, каждый из которых может имеет собственный тип значений. Например, если говорить о людях, то столбцы могут содержать и числовые данные (рост, возраст, вес), и текстовые (имя), и качественные данные (пол).

В матрице все данные представлены в виде чисел, что позволяет использовать ее, как единый объект, и применять различные математические операции ко всем числам. Отдельные строки и столбцы матрицы называются *векторами*. Т.е. значения различных переменных для одного объекта представляются в виде вектора-строки, а значения одной переменной, измеренной для нескольких объектов — в виде вектора-столбца.

Для того, чтобы лучше понять смысл представления данных в виде векторов и матриц, вернемся еще раз к их геометрическому представлению в виде графиков рассеяния. Итак, данные могут быть представлены в виде облака точек в N-мерном пространстве переменных (N — число переменных). Каким образом можно представить другие переменные в этом пространстве? В виде векторов.

Вектор почти ничем не отличается от точки, он также может быть задан в виде последовательности координат, но вектор задаёт некоторое направление в пространстве. Чтобы отличить координаты вектора

от координат точки (измерения) будем представлять векторы в виде столбцов, а переменные — в виде строк, как мы уже упомянули в предыдущем параграфе.

> *Для того, чтобы легко понять в тексте о каком именно векторе идет речь, введем несколько обозначений. Во-первых, любой вектор будем обозначать строчной латинской буквой в полужирном начертании, например,* $\mathbf{x}$. *Во-вторых, координаты вектора-строки будем обозначать в квадратных скобках:* $[10, 20]$, *а вектора-столбца — таким же образом но с индексом T:* $[10, 20]^T$. *Этот индекс обозначает операцию транспонирования, которая делает столбец строкой и наоборот.*

Посмотрим еще раз на первый график на рисунке 1.5 из предыдущего раздела, где представлена зависимость между содержанием ионов хлора и натрия. Как мы уже отмечали, наблюдается некоторая взаимосвязь между этими двумя переменными — у образцов с высоким содержанием ионов натрия, содержание ионов хлора тоже высоко и наоборот. Эта связь имеет линейный характер, можно представить себе некую прямую, которая пересекает это облако точек точно посередине.

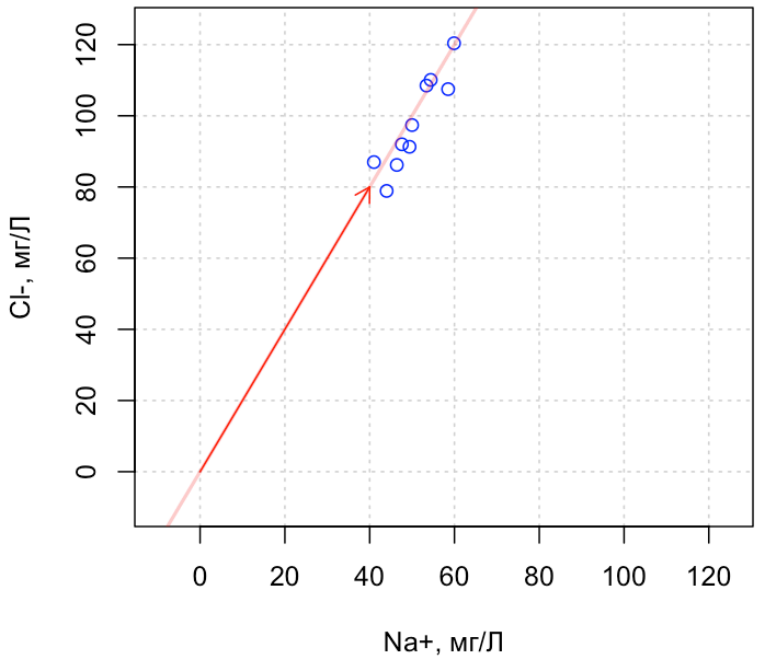


Рис. 1.6. График рассеяния для набора данных с концентрацией ионов в воде, показанный в полномасштабном виде: оси имеют одинаковый диапазон и содержат начало координат.

Такую прямую можно задать с помощью вектора. Однако, в этом случае, так как речь идет о направлении

в пространстве переменных, это направление всегда задается относительно начала координат, которое мы не видим на исходном графике. График на рисунке 1.6 дает такую возможность, он также показывает, что вектор с координатами $[40, 80]^T$ (показан на графике в виде ярко-красной стрелки) мог бы стать неплохим выбором для задания направления нашей прямой.

Обратите внимание, что начало стрелки, которая соответствует нашему вектору, находится в начале координат, а ее конец — в точке с координатами вектора ($x = 40$, $y = 80$). Очевидно, что векторы с координатами $[20, 40]^T$ или $[1, 2]^T$ будут задавать точно такое же направление. Для того, чтобы уйти от этой многозначности, направление принято задавать *единичными векторами* (англ. *unit vectors*), длина которых всегда равна единице. Так как вектор и его координаты в двумерном пространстве представляют собой прямоугольный треугольник, то длину вектора можно легко вычислить по теореме Пифагора:

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2}$$

Длина нашего исходного вектора равна $\sqrt{40^2 + 80^2} = 89.4$, таким образом, разделив обе координаты на это число, мы получим единичный вектор $[0.4472, 0.8944]^T$. Длина вектора, которую, как вы уже заметили, мы будем обозначать в виде $\|\mathbf{x}\|$, так же называется его *нормой*, а операция приведения вектора к единичному, когда каждая координата делится на длину вектора, называется *нормировкой*.

Использование единичных векторов для задания направления в пространстве имеет еще одно преимущество, позволяя делать проекции точек на прямую, ориентированную вдоль этого направления. Для того, чтобы понять, как сделать такую проекцию, нужно сначала вспомнить несколько простых операций из линейной алгебры.

1.3.1 Скалярное произведение

Скалярное произведение является основной операцией в хемометрике и будет использоваться в этой книге довольно часто. Для начала вспомним, что скалярное произведение двух векторов-столбцов, $\mathbf{a} = [a_1, a_2, ..., a_n]^T$ и $\mathbf{b} = [b_1, b_2, ..., b_n]^T$ определяется, как линейная комбинация их координат:

$$\mathbf{a}^T\mathbf{b} = a_1b_1 + a_2b_2 + ... + a_nb_n$$

Обратите внимание на знак транспонирования у первого вектора — по сути, скалярное произведение определено для вектора-строки и вектора-столбца (строка умножается на столбец). Понятно, что в этом случае длина векторов должна быть одинаковой. Такое произведение еще иногда называют *внутренним*. Существует еще и внешнее произведение, в результате которого получится не скаляр, а матрица, о нем мы расскажем позже. Рисунок 1.7 показывает пример скалярного умножения двух векторов из четырехмерного пространства.

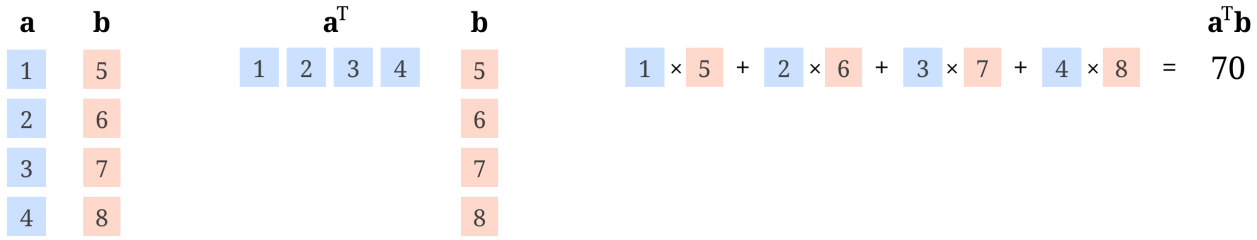


Рис. 1.7. Скалярное произведение двух векторов.

Скалярное произведение позволяет очень компактно записывать операции с векторами и матрицами. Например вычисление нормы вектора $\mathbf{x}$ можно записать в форме:

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^{\mathrm{T}}\mathbf{x}},$$

независимо от его длины.

Еще одно важное свойство скалярного произведения двух векторов, заключается в том, что оно позволяет определить векторы, расположенные ортогонально друг другу (т.е. угол между ними в пространстве равен 90°). Скалярное произведение таких векторов всегда равно нулю. Попробуйте в качестве тренировки нарисовать два вектора с координатами $[4, 1]^{\mathrm{T}}$ и $[-1, 4]^{\mathrm{T}}$ на листке бумаги, измерить угол между ними, а затем вычислить скалярное произведение для этих векторов.

> *Чтобы нарисовать вектор в двухмерном пространстве, используйте листок бумаги, размеченный в виде клеток. Для начала нарисуйте оси X и Y. После этого найдите точку с координатами $X = 4$ и $Y = 1$ и нарисуйте линию, соединяющую эту точку с началом координат. Потом повторите процедуру для второго набора координат, $X = -1$, $Y = 4$. Используйте транспортир, чтобы измерить угол между этими двумя линиями.*

Скалярное произведение можно также задать и для двух матриц, **A** и **B**, для этого нужно каждую строку первой матрицы скалярно умножить на каждый столбец второй матрицы. Так как в результате скалярного произведения двух векторов всегда получается число (скаляр, отсюда, собственно, и название), то, в результатом такой операции для двух матриц станет тоже матрица, в которой число строк равно числу строк в **A**, а число столбцов — числу столбцов в матрице **B**. Рисунок 1.8 показывает схему для скалярного произведения матриц.

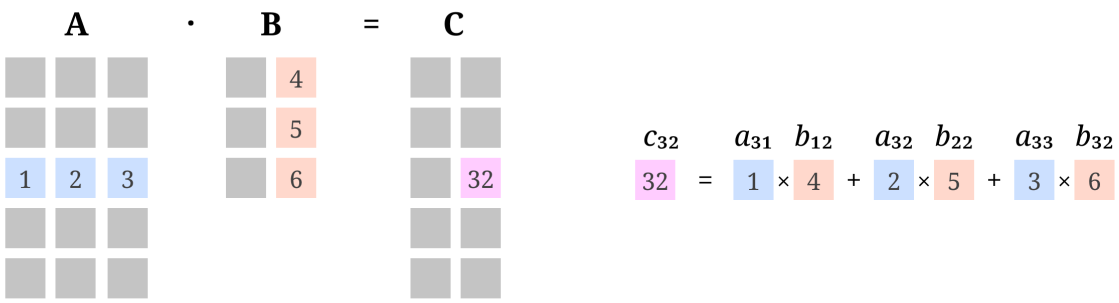


Рис. 1.8. Скалярное произведение двух матриц. Каждый элемент матрицы **C** является результатом скалярного произведения строки матрицы **A** и столбца матрицы **B**.

Мы можем использовать эти операции даже для описания повседневных рутинных действий. Предположим, что вы решили приготовить простое блюдо: смешать вареный рис и яйца, после чего запечь полученную смесь в духовке. Мы знаем, что число жиров, белков и углеводов в каждом продукте следующее (грамм на 100 грамм продукта):

	Яйцо	Рис
Белки	12	4
Жиры	13	1
Углеводы	1	25

Вопрос — каково будет содержание белков, жиров и углеводов в полученном блюде? Очевидно, что он зависит от пропорций и представим, что вы рассматриваете следующие варианты:

	Яйцо	Рис
Вариант 1	1	1
Вариант 2	2	1
Вариант 3	1	2

Здесь 1 означает 100 г. продукта, 2 — 200 г. и так далее. Очевидно, что содержание белков в варианте 3 можно посчитать, умножив массу каждого ингредиента на содержание белков в нем:

$$1 \times 12 + 2 \times 4 = 20$$

Теперь посмотрите внимательно — это не что иное, как скалярное произведение третьей строки из второй таблицы на первую строку из первой (которую нужно сначала транспонировать). Если мы представим все значения из второй таблицы в виде матрицы C с тремя сроками и двумя столбцами, все значения из первой таблицы в виде матрицы S с тремя строками и двумя столбцами, то скалярное произведение этих двух матриц:

$$\mathbf{D} = \mathbf{C}\mathbf{S}^{\mathrm{T}},$$

будет представлять собой матрицу 3×3, каждая строка которой будет представлять содержание жиров, белков и углеводов для каждого варианта блюда из второй таблицы.

Более того, мы можем представить эти данные в виде трехмерного Декартового пространства, ось X которого соответствует содержанию белков, ось Y — содержанию жиров, а ось Z — содержанию углеводов в блюдах и ингредиентах. В этом случае, каждый отдельный ингредиент можно представить в виде вектора в этом пространстве, а готовое блюдо — в виде точки.

Попробуйте вычислить все значения матрицы **D**, а потом построить такое пространство на листке бумаги в качестве упражнения. Это может здорово помочь в понимании многих нюансов представления многомерных данных и различных операций над ними, которые мы собираемся затронуть в этой книге.

1.3.2 Проецирование точек на прямую с помощью скалярного произведения

Другая важная операция с данными, которую мы будем активно эксплуатировать в следующей главе — это проекция. Рассмотрим, как скалярное произведение позволяет упростить получение проекции точек на прямую, заданную единичным вектором. Чтобы получить такую проекцию, сначала нужно умножить скалярно координаты каждой точки на координаты этого вектора, как показано ниже для точки из седьмой строки нашей таблицы с содержанием ионов в воде:

$$\begin{bmatrix} 44.0 & 78.9 \end{bmatrix} \begin{bmatrix} 0.4472 \\ 0.8944 \end{bmatrix} = 90.2$$

Однако, в результате получится одно число, которое показывает, как далеко эта точка находится от начала координат вдоль этой прямой (см. график на рисунке 1.9). Чтобы получить координаты проекции в исходном пространстве переменных нужно умножить это число на транспонированный вектор:

$$\begin{bmatrix} 90.2 \end{bmatrix} \begin{bmatrix} 0.4472 & 0.8944 \end{bmatrix} = \begin{bmatrix} 80.7 & 40.4 \end{bmatrix}$$

Это можно записать короче в виде:

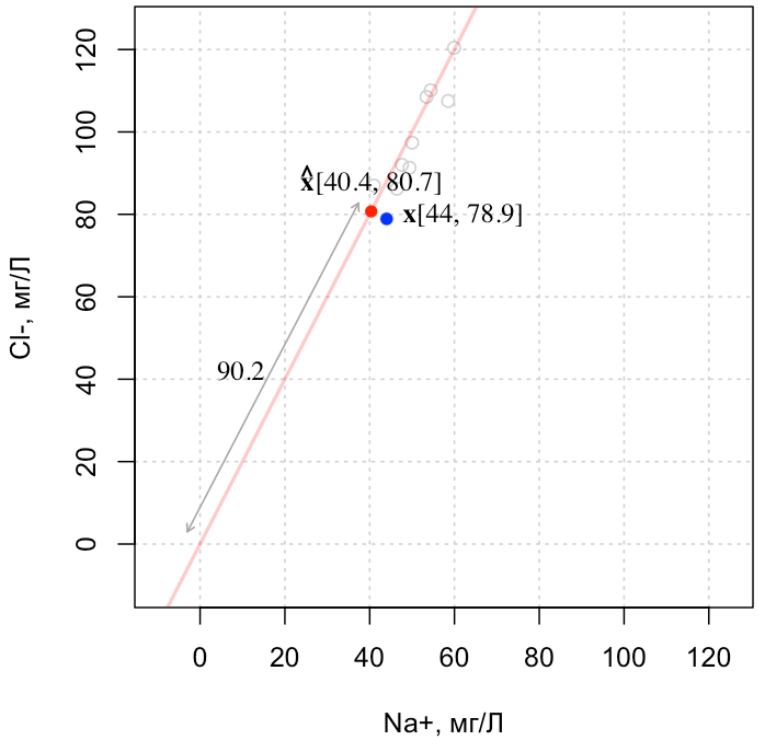


Рис. 1.9. Проекция отдельно выбранной точки на прямую, заданную единичным вектором.

$$\hat{\mathbf{x}} = \mathbf{x}\mathbf{p}\mathbf{p}^{\mathrm{T}} = \mathbf{x}\mathbf{\Pi}$$

Здесь $\mathbf{x}$ — это вектор-строка с координатами исходной точки (в нашем случае — значения из седьмой строки таблицы с данными или синяя точка на графике), $\hat{\mathbf{x}}$ — это вектор-строка с координатами ее проекции (красная точка на графике), $\mathbf{p}$ — единичный вектор, который мы определили ранее. Матрица $\mathbf{\Pi} = \mathbf{p}\mathbf{p}^{\mathrm{T}}$ называется *проекционной матрицей*.

Теперь, независимо от размерности нашего пространства и числа измерений, мы можем вычислить проекции всех точек в исходной матрице $\mathbf{X}$ на прямую, заданную единичным вектором $\mathbf{p}$, используя простую операцию:

$$\hat{\mathbf{X}} = \mathbf{X}\mathbf{p}\mathbf{p}^{\mathrm{T}} = \mathbf{X}\mathbf{\Pi}$$

Как будет показано ниже, в R (как и во многих других подобных языках) — это ровно одна строка кода. Другими словами, знание простых операций линейной алгебры существенно упрощает многие вычисления.

> *Операция проекции имеет непосредственное практическое значение в химии. Представьте, что координатные оси — это две длины волны, на которых вы измеряете поглощение инфракрасного излучения: λ_1 и λ_2. Предположим, что чистый этиловый спирт поглощает 60% на λ_1 и 80% излучения на λ_2. Тогда спектр чистого спирта можно задать в этом пространстве координат с помощью единичного вектора $\mathbf{p} = [0.6, 0.8]^T$. Далее вы делаете измерение для образца, содержание спирта в котором неизвестно и получаете другие значения поглощения на этих длинах волн: $\mathbf{x} = [x_1, x_2]$. Это измерение геометрически можно представить в виде точки в данной системе координат. Так вот, проекция этой точки на вектор $\mathbf{p}$ даст вам концентрацию этилового спирта в вашем образце, $c = \mathbf{x}\mathbf{p}$.*

1.3.3 Операции с векторами и матрицами в R

Матрицы являются альтернативой фреймам с данными, если все значения числовые. Использование матриц имеет как плюсы (например, возможность применять операции линейной алгебры), так и минусы (не так просто работать с отдельными столбцами). Код ниже показывает, как создать матрицу из простой последовательности чисел от 1 до 12 с помощью функции `matrix()`:

```
m <- matrix(1:12, nrow = 3, ncol = 4)
show(m)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

Как можно заметить, матрица “заполняется” числами из предоставленной последовательности по столбцам. Это можно поменять указав дополнительный параметр `byrow = TRUE` как показано в следующем блоке кода:

```
m <- matrix(1:12, nrow = 3, ncol = 4, byrow = TRUE)
show(m)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
```

Наконец, матрицы можно делать из фреймов, и наоборот. Для этого есть специальные методы `as.matrix()` и `as.data.frame()`. Первым нужно пользоваться осторожно, так как если фрейм содержит хотя бы один столбец с текстовыми данными, то вся матрица будет также содержать текстовые значения. В блоке кода ниже показано, как из фрейма, созданного ранее для хранения значений концентраций ионов, сделать матрицу:

```
d <- data.frame(
   Cl = c(50.0, 41.0, 47.6, 59.9, 53.4, 54.4, 44.0, 49.4, 46.4, 58.5),
   Na = c(97.4, 87.0, 92.0, 120.4, 108.5, 110.1, 78.9, 91.3, 86.2, 107.5),
   K = c(14.8, 13.8, 13.9, 14.2, 13.6, 15.5, 15.1, 13.5, 11.1, 16.9)
)

m <- as.matrix(d)
show(class(m))
```

```
[1] "matrix" "array"
```

```
show(class(d))
```

```
[1] "data.frame"
```

Метод `class()` позволяет увидеть тип, или класс переменных/объектов в R. Как можно видеть, R считает, что объект `m` — это матрица, или массив. Тогда как объект `d` относится к классу фреймов.

Матричные операции

Все арифметические операции в R можно применять и к матрицам, при этом они будут выполняться поэлементно, т.е. подразумевается, что матрицы имеют один и тот же размер. Скажем следующий код вычисляет квадрат концентраций для наших данных.

```
# превращаем фрейм в матрицу
conc <- as.matrix(d)

# умножаем матрицу с концентрациями на саму себя поэлементно
conc2 <- conc * conc

# показываем первые три строки новой матрицы
head(conc2, n = 3)
```

```
          Cl      Na      K
[1,] 2500.00 9486.76 219.04
[2,] 1681.00 7569.00 190.44
[3,] 2265.76 8464.00 193.21
```

Для того, чтобы выполнить скалярное умножение, необходимо использовать оператор %*% вместо *. Блок с кодом ниже вычисляет матрицу **D** с содержанием белков, жиров и углеводов из примера с блюдами из яиц и риса, который мы рассмотрели ранее.

```
# задаем матрицу с БЖУ ингредиентов
S <- matrix(c(12, 13, 1, 4, 1, 25), nrow = 3, ncol = 2)
rownames(S) <- c("Белки", "Жиры", "Углеводы")
colnames(S) <- c("Яйцо", "Рис")
show(S)
```

```
         Яйцо Рис
Белки       12   4
Жиры        13   1
Углеводы     1  25
```

```
# задаем матрицу с содержимым ингредиентов в разных вариантах блюда
C <- matrix(c(1, 2, 1, 1, 1, 2), nrow = 3, ncol = 2)
rownames(C) <- c("Вариант 1", "Вариант 2", "Вариант 3")
colnames(C) <- c("Яйцо", "Рис")
show(C)
```

```
          Яйцо Рис
Вариант 1    1   1
Вариант 2    2   1
Вариант 3    1   2
```

```
# вычисляем БЖУ для каждого варианта
D <- C %*% t(S)
show(D)
```

```
          Белки Жиры Углеводы
Вариант 1    16   14       26
Вариант 2    28   27       27
Вариант 3    20   15       51
```

Здесь функция `t()` выполняет операцию транспонирования. Заметим, что при выполнении операции скалярного умножения матрица, полученная в результате, "унаследовала" правильные названия строк и столбцов, что делает вывод результата более наглядным.

Если нужно получить скалярное произведение двух матриц или векторов в форме $\mathbf{A}^{\mathrm{T}}\mathbf{B}$, или $\mathbf{A}\mathbf{B}^{\mathrm{T}}$, то лучше (и быстрее) использовать специальные методы, `crossprod(A, B)` и `tcrossprod(A, B)` соответственно.

Блок кода ниже показывает решение задачи проекции точек на направление, заданное вектором $[40, 80]^{\mathrm{T}}$, которое мы рассмотрели ранее.

```
# превращаем фрейм в матрицу и берем только два первых столбца
conc <- as.matrix(d[, 1:2])

# задаем координаты вектора
p <- c(40, 80)

# вычисляем длину вектора (норму) и делаем нормировку
p_length <- sqrt(crossprod(p))
```

```
p <- p / as.numeric(p_length)

# вычисляем проекционную матрицу и проецируем точки
Pi <- tcrossprod(p)
conc_proj <- conc %*% Pi

# показываем график с оригинальными значениями
plot(conc[, 1], conc[, 2], col = "blue", xlim = c(30, 120), ylim = c(30, 120),
     xlab = "Cl-, mg/L", ylab = "Na+, mg/L")

# добавляем направление проекции и спроецированные точки
abline(a = 0, b = p[2]/p[1], col = "pink")
points(conc_proj[, 1], conc_proj[, 2], col = "red")
```

Результат выполнения кода показан на рисунке 1.10. Оси приведены к одному масштабу чтобы направление проекций отображалось правильно (под прямым углом к прямой).

Попробуйте поменять координаты вектора **p**, и запустить код еще раз, все будет посчитано правильно. Еще раз отметим, что без использования скалярного произведения решение этой задачи было бы гораздо более громоздким и потребовало бы использования нескольких циклов.

Блок с кодом ниже показывает, как сделать первый график, показанный на рисунке 1.6, где к точкам добавляется решетка, стрелка для вектора $[40, 80]^{\mathrm{T}}$ и бесконечная прямая в направлении этого вектора.

```
plot(
   d$Cl, d$Na,
   pch = 1,
   col = "blue",
   xlab = "Cl-, mg/L",
   ylab = "Na+, mg/L",
   xlim = c(0, 120),
   ylim = c(0, 120)
)
grid()
abline(a = 0, b = 80/40, lwd = 2, col = "pink")
arrows(0, 0, 40, 80, lwd = 2, col = "red", length = 0.1)
```

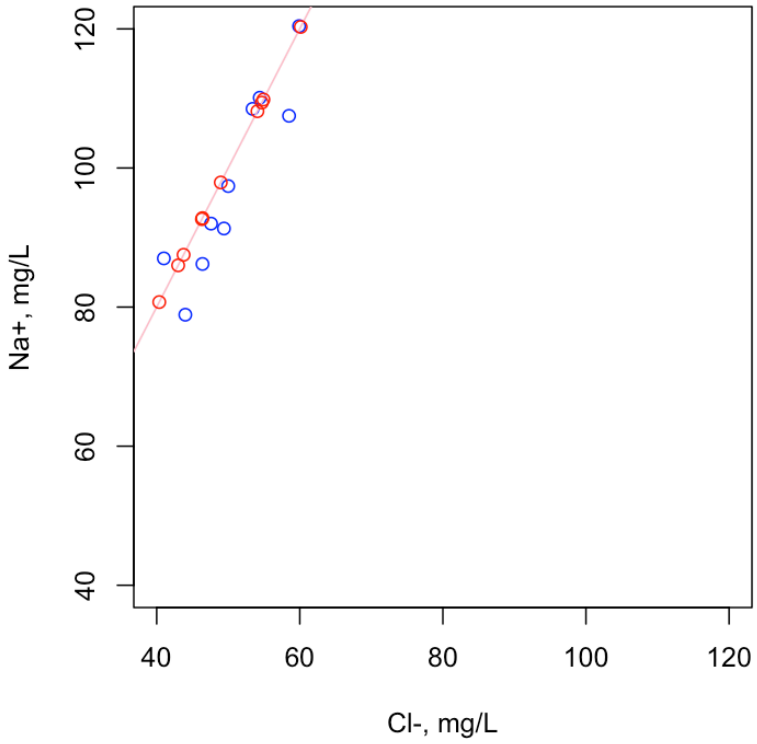


Рис. 1.10. Результат выполнения кода

1.4 Статистическое описание данных

Данный раздел посвящен статистическому описанию экспериментальных данных. Вообще говоря, предполагается, что читатель уже знаком с методами прикладной статистики, поэтому изложение материала будет в меру компактным, нацеленным в первую очередь на то, чтобы освежить необходимые знания, нежели приобрести новые.

Для чего вообще нужна статистика? Ответ на этот вопрос непосредственно связан с тем, как устроен наш мир. Дело в том, что почти все события в нем имеют случайный, вероятностный характер. Одни события более вероятны, чем другие, но, тем не менее, почти ни одно событие не имеет 100% вероятность. Скажем, если вы планируете завтра в 8 утра быть в университете, то на самом деле вы окажетесь там либо немного раньше, либо немного позже 8:00. Шанс, что вы переступите его порог ровно в 8:00 очень мал (здесь мы говорим о точном времени, измеренным, скажем, с точностью до миллисекунды). Может так случится, что опоздание будет достаточно большим (например, транспорт будет ходить с задержками), а может и наоборот, вы выйдете раньше и вам повезет с транспортом.

Такая неточность, или неопределенность (англ. *uncertainty*), сопровождает почти любое событие. Скажем, если вы планируете эксперимент, то исход этого эксперимента — тоже событие. И предугадать этот исход с абсолютной точностью, в принципе, невозможно. Даже если вы повторите подряд несколько идентичных экспериментов, их исход будет немного различаться.

Попробуйте, например, определить сколько железа есть в стакане воды, взятой из какого-то источника, скажем, небольшого пруда в парке. А после этого наполните стакан водой из этого пруда еще раз и повторите измерение. Каждый раз, когда вы будете делать новое измерение, его результат будет отличаться от предыдущего. Т.е. узнать точно, сколько железа содержится в этом пруду — невозможно.

Причин для этого несколько. Во-первых, измерительный прибор (предположим, вы используете для этого оптический эмиссионный спектрометр с индуктивно-связанной плазмой, ИСП-ОЭС) всегда имеет погрешность. Тут надо сказать, что обычно погрешность прибора вносит наименьшую неопределенность. Подготовка образца для измерения (например, вам нужно приготовить раствор) внесет большую неточность, особенно, если в подготовке принимает участие человек и пробоподготовка предполагает несколько этапов. Ну и, наконец, химический состав воды в пруду тоже не является полностью однородным. Вода, взятая в разных местах, будет иметь разную концентрацию железа. Все это вместе и дает итоговую неопределенность.

Собственно, статистика — это инструмент, который позволяет справиться с этой проблемой. Делается это в первую очередь с помощью оценки неопределенности. Скажем, с помощью статистических методов и правильно поставленного эксперимента мы можем определить, что концентрация железа в стакане воды находится между 0.2 и 0.4 мг на один литр или, другими словами, 0.3 ± 0.1 мг/л. В данном случае ± 0.1 это и есть та самая неопределенность.

Зная неопределенность, можно также оценивать различные вероятности. Например, можно оценить, сколько раз измеренная концентрация железа будет меньше 0.25 мг на литр, если мы повторим измерение для одного и того же стакана 100 раз, или возьмем 100 стаканов воды из одного и того же источника. Или же — каковы шансы получить измеренное значение больше 0.4 мг/л, если мы повторим эксперимент.

Оценивать неопределенность измерения, сделанного для одного объекта, и для их совокупности, это разные задачи, но они используют одни и те же статистические методы. В этом разделе мы будем говорить в первую очередь о неопределенности измерений, сделанных для некоторого множества объектов. Но для начала давайте определим, что мы подразумеваем под этим множеством.

1.4.1 Выборка и генеральная совокупность

Генеральная совокупность, или популяция (англ. *population*) — это все объекты, которые представляют интерес для исследования. Если вы, к примеру, хотите знать, загрязнена ли вода в озере продуктами хлора, то генеральная совокупность представляет собой всю воду в этом озере, до последней капли. Поиск лекарства, которое позволит бороться с мигренью, подразумевает, что ваша популяция — все люди на планете, которые страдают от этого недуга.

Понятно, что работать с таким большим числом объектов на практике невозможно, поэтому обычно прибегают к следующему упрощению. Из генеральной совокупности извлекается *выборка* (англ. *sample*) — ограниченный набор объектов, достаточный, чтобы более или менее точно описать всю совокупность в целом. Далее все измерения делаются для этого набора, а результаты обобщают на всю совокупность.

Таким образом, статистические методы и подходы можно разделить на две группы. Первая позволяет описать полученные для выбранных объектов экспериментальные данные, оценить их неопределенность и другие характеристики. Эта группа методов так и называется — *описательная статистика* (англ. *descriptive statistics*). Об этих методах мы и будет говорить в этом разделе. Вторая группа методов, позволяющая обобщить результаты и сказать что-то о генеральной совокупности целиком, называется *индуктивная статистика* (англ. *inferential statistics*), она будет рассмотрена в следующем разделе.

Рисунок 1.11 схематично показывает отношения между выборкой и генеральной совокупностью.

Основные этапы статистического анализа:

1. Получение выборки (пробоотбор).
2. Статистическое описание измерений, сделанных для выборки.
3. Индуктивный анализ, обобщающий результаты на генеральную совокупность.

Часто в литературе по прикладной статистике и в научных статьях для обозначения числа объектов в генеральной совокупности (если оно измеримо) используют прописную латинскую N, а для обозначения числа объектов в выборке строчную n. Параметры популяции также часто обозначают греческими буквами, как на рисунке, тогда как соответствующие характеристики выборки — латинскими. Например, среднее

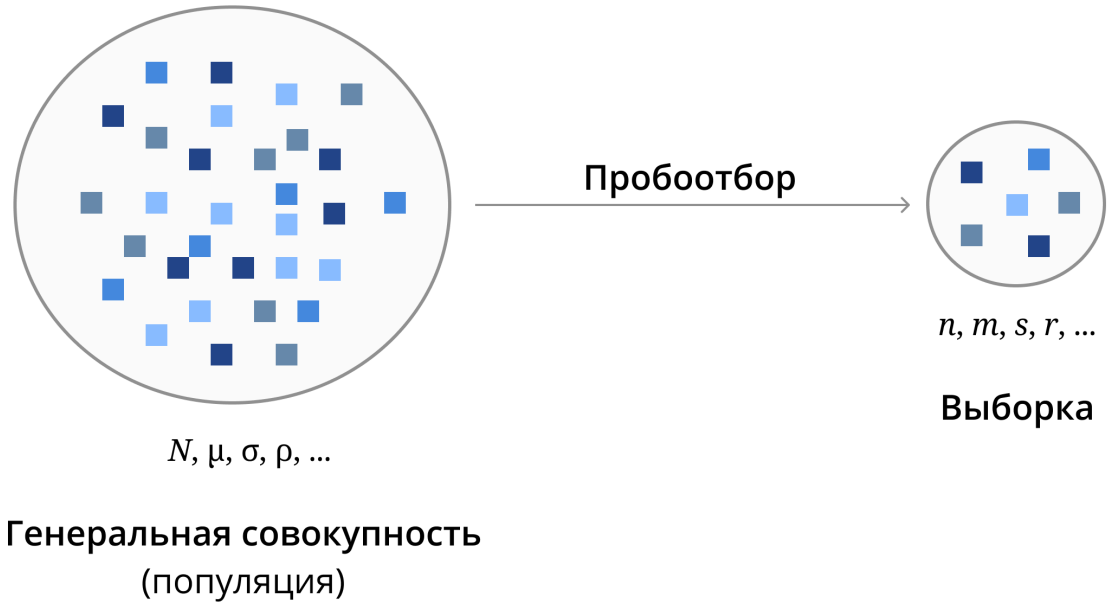


Рис. 1.11. Схематическое представление отношений между выборкой и генеральной совокупностью.

значение обозначается как $\mu(x)$, если речь идет о параметре популяции и $m(x)$, если речь идет о значении, полученном для выборки. В некоторых случаях используют только латинские буквы, а для различия между параметром популяции и статистикой выборки, значения, полученные для последней, обозначают с символом "крышки" сверху, $\hat{m}(x)$, который означает, что значение является оценкой (англ. *estimate*) искомого параметра.

Процедура получения выборки из генеральной совокупности носит название *пробоотбора* (англ. *sampling*). Строго говоря, пробоотбор не является напрямую частью статанализа, но очень сильно влияет на его результат. Известная среди специалистов по анализу данных поговорка: "Мусор на входе — мусор на выходе" (англ. *garbage in, garbage out*), относится именно к неправильному пробоотбору. Можно использовать самые современные и эффективные методы анализа, все они будут бессильны если выборка была получена неправильно.

Теория пробоотбора — отдельная дисциплина, с которой мы настоятельно рекомендуем ознакомиться подробнее. Если же говорить коротко и упрощенно, то хорошая выборка должна удовлетворять двум критериям:

1. Быть случайной. Это значит, что изначально все объекты генеральной совокупности имели равный шанс оказаться в вашей выборке. Скажем, если вы решили провести опрос студентов всего

университета на определенную тему, но поленились и спросили только своих одногрупников, такая выборка не будет случайной (обычно в этом случае ее называют смещенной, англ. *biased*).

2. Быть репрезентативной. Это значит, что она должна как можно лучше представлять вашу генеральную совокупность. С репрезентативностью непосредственно связаны два важных момента — это размер выборки и однородность популяции. Чем сильнее неоднородность популяции тем сложнее получить репрезентативную выборку относительно небольшого размера.

Приведем несколько простых примеров для иллюстрации. Предположим, вы решили оценить шансы выпадения орла, или решки для неизвестной монеты. Если вы подкинете монету три раза (т.е. размер вашей выборки будет равен трём) и все три раза выпадет орел, то такая выборка не будет репрезентативной, так как она вообще не предполагает появления решки. Если же подкинуть монету 20 раз, то шанс, что все 20 раз выпадет только орел, или только решка очень мал (если предположить, что монета сбалансирована и вероятность выпадения орла или решки одинакова и равна 50%, то искомый шанс $0.5^{20} \approx 0.00000095 \approx 0.0001\%$, или один на миллион).

Все методы, которые будут рассмотрены в этой книге, предполагают, что ваша выборка удовлетворяет этим двум критериям. В противном случае выводы, к которым может привести анализ такой выборки, будут неправильными.

1.4.2 Параметрический и непараметрический подходы

Итак, вы получили случайную и репрезентативную выборку и провели необходимые измерения, что дальше? В первую очередь нужно описать полученные результаты статистически. Так как измеренные значения будут варьироваться случайным образом, их описывают как интервал, оценивая три основных характеристики этого интервала:

1. Центр
2. Разброс (как далеко значения могут отдаляться от центра)
3. Плотность значений на разных участках

Существует два разных подхода к оценке этих характеристик. Первый основывается на предположении, что измеренные величины подчиняются определенным математическим законам — теоретическим распределениям. Эти законы представляют собой обычные математические функции с одним, или несколькими параметрами. В этом случае задача описания по сути сводится к определению этих параметров, соответственно, такой подход и называется *параметрическим*. Второй подход не предполагает использования теоретических предположений о распределении значений и основан исключительно на полученных данных. Это *непараметрический* подход.

У каждого подхода есть свои плюсы и минусы. Параметрический подход предпочтителен в силу нескольких причин. Во-первых, число параметров обычно варьируется от 1 до 3, а значит для их оценки не требуется

большого числа измерений. Зная эти параметры и вид теоретической функции, можно затем “забыть” об исходных данных и делать нужные вычисления (интервалы неопределенности или вероятности) с помощью этой функции.

Однако, чтобы использовать параметрический подход, необходимо, чтобы ваши экспериментальные данные хорошо описывались такой функцией. Другими словами они должны соответствовать некоторым стандартам качества, что не всегда возможно. В этом случае и поможет непараметрический подход. Его основной минус — он требует наличия относительно большого числа измерений (обычно от 20 и выше) по причине которую мы рассмотрим ниже.

1.4.3 Непараметрическое описание данных

Основная статистика, используемая в непараметрическом описании числовых данных — *квантиль* и его производные (квартиль, персентиль и т.д.). Идея заключается в том, чтобы разбить все полученные значения на равные порции. Например, если у вас 12 значений (содержание железа в 12 стаканах воды), то их можно разбить на четыре группы — по три значения в каждом. Три самых маленьких значения (25%), три следующих по величине и т.д. Процедура разбиения выглядит следующим образом:

1. Упорядочить все значения чтобы получилась неубывающая последовательность (каждое следующее число равно или больше предыдущего).
2. Разбить упорядоченные значения на равные порции (в каждой порции — равное число значений)
3. Найти границы интервалов этих порций, которые и будут служить квантилями

Число квантилей всегда на один меньше числа порций (например, чтобы разбить все значения на две равных порции нужен один квантиль).

Рассмотрим эту процедуру на следующем примере, значения для которого возьмем из таблицы с результатами химического анализа 10 проб воды, использованной нами в предыдущем разделе. Для простоты возьмем только первый столбец этой таблицы с содержанием хлорид-ионов и упорядочим все значения по возрастанию:

#	1	2	3	4	5	6	7	8	9	10
Cl-, мг/л	41.0	44.0	46.4	47.6	49.4	50.0	53.4	54.4	58.5	59.9

Предположим, нам нужно разбить всю последовательность на четыре равные группы. Для этого нужно вычислить три квантиля — границы между группами. Традиционно, для случая из четырех групп такие границы называются *квартилями*, так как в каждой группе находится 25% значений или одна четверть (англ. *quarter*).

Самое простое в этом случае воспользоваться последовательным делением пополам: сначала разбить все значения на две равных группы и затем сделать тоже самое для каждой группы отдельно. Так как число

исходных значений в нашей выборке четное, в этом случае граница между половинками будет проходить посередине между пятым (49.4 мг/л) и шестым (50.0 мг/л) значениями, что дает нам $(49.4 + 50.0)/2 = 49.7$ мг/л. Эта граница, делящая все значения выборки на две равные группы (50% значений меньше 49.7 мг/Л и 50% больше) называется вторым квартилем, $Q2$, или *медианой*.

Далее можно найти первый квартиль, который будет соответствовать медианному значению для первой группы. Так как число значений нечетное, третье значение в группе (46.4 мг/л) как раз находится посередине (два значения меньше него и два больше). Т.е. оно и будет первым квартилем, $Q1$. Рассуждая таким же образом получаем третий квартиль — $Q3 = 54.4$ мг/л.

Надо сказать, что это не единственный способ расчета квантилей и есть множество других. Например функция `quantile()` в R может делать вычисления девятью различными способами. Для выборок большого размера (от 30 и выше) итоговый результат будет различаться незначительно, если же выборка маленькая, как в нашем случае, то расхождение может быть существенным.

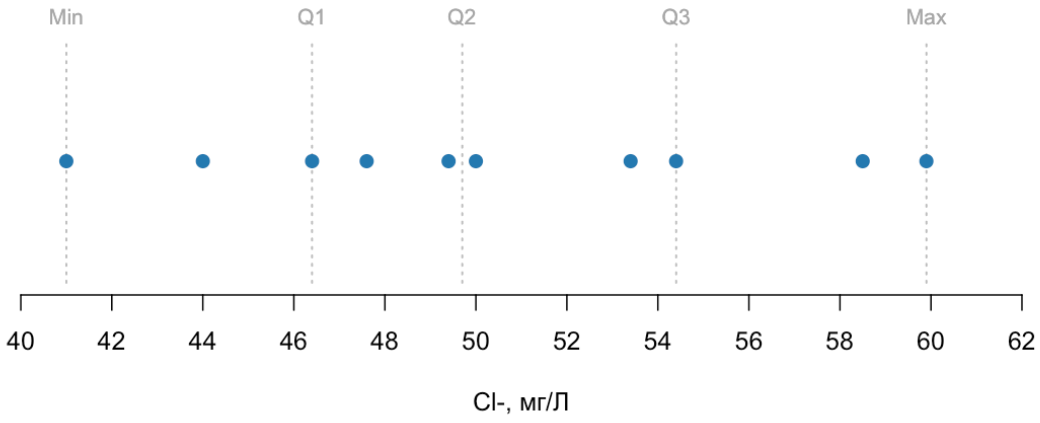


Рис. 1.12. Числовая ось со значениями концентраций и пятью статистиками.

Кроме квартилей мы также можем легко найти наименьшее и наибольшее значения. График на рисунке Figure 1.12 показывает числовую ось со всеми измеренными значениями, а также положение полученных нами пяти статистик (они показаны в виде вертикальных линий). Глядя на их расположение, можно увидеть, что данные распределены более или менее равномерно по числовой прямой. И относительно симметрично, можно видеть, что медиана расположена немного ближе к первому квартилю (Q1), однако, это скорее всего случайный эффект, связанный с маленьким размером выборки.

Полученные пять статистик принято отображать в виде отдельного графика, который называется *диаграммой размаха*. На английском языке этот график имеет забавное название — *коробка с усиками* (*box*

and whiskers), что буквально описывает его вид. Коробка (box) образуется с помощью квартилей, которые формируют ее левую (Q1) и правую (Q3) стороны. Медиана (Q2) обозначается вертикальной линией внутри коробки. Усики же обычно показывают минимальное и максимальное значения.

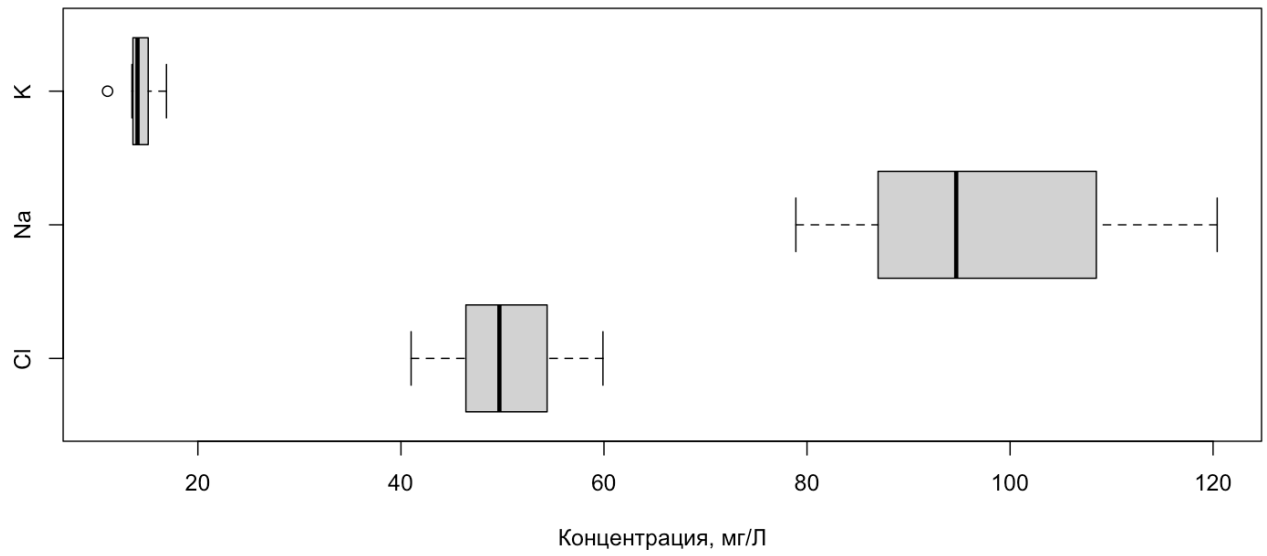


Рис. 1.13. Диаграмма размаха для концентрации трех видов ионов.

Если в данных имеются выбросы — значения, которые экстремально малы, или экстремально велики по сравнению с другими, то обычно они отделяются от остальных данных и показываются на диаграмме размаха отдельно в виде точек. Делается это так. Сначала вычисляется межквартильное расстояние (англ. *inter quartile range*, IQR): $IQR = Q3 - Q1$. Очевидно, что IQR — это диапазон значений для 50% данных, причем расположенных посередине, т.е. наименее экстремальных. Далее вычисляются границы для выбросов, для левой стороны как: $Q1 - 1.5IQR$ и для правой как $Q3 + 1.5IQR$. Если какое-то значение находится за пределами этих границ, то оно отделяется от данных и показывается на графике отдельно, а положение усов определяется без выбросов.

Рисунок Figure 1.13 показывает диаграммы размаха для каждого столбца из таблицы, использованной в предыдущем разделе. Так, самый нижний график показывает содержание хлорид-ионов и расположение частей коробки и усов соответствует тому, что мы показали ранее на рисунке Figure 1.12. Можно также заметить, что ионы натрия (график посередине) имеют наибольшую концентрацию в наших образцах, а так же довольно сильно варьируются от образца к образцу (от почти 80 до 120 мг/Л).

График для ионов калия выглядит немного необычно. Во-первых, очевидно, что один из образцов имеет довольно маленькую концентрацию, которая была определена, как выброс (точка слева). Из-за этого сама диаграмма имеет сильный скос — левая ее часть намного короче правой. Так иногда бывает, если

строить диаграмму размаха для небольшого числа значений, как в нашем случае. Как уже говорилось ранее, для использования непараметрических методов требуется большое число данных, чтобы правильно аппроксимировать положение квантилей.

Стоит также сказать, что диаграмму размаха часто отображают в вертикальном виде.

1.4.4 Собственные квантили значений

Для каждого отдельного значения в выборке можно оценить, сколько значений меньше и сколько больше данного. Такая оценка обычно называется *собственными квантилями значений*. Как и в случае с обычными квантилями, имеется несколько разных способов такой оценки, мы воспользуемся самым простым, результаты которого показаны в следующей таблице:

#	1	2	3	4	5	6	7	8	9	10
Cl-, мг/л	41.0	44.0	46.4	47.6	49.4	50.0	53.4	54.4	58.5	59.9
Квантиль	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9

Возьмем, к примеру, второе по порядку значение, 44.0 мг/л. Очевидно, что в нашей выборке есть только одно значение меньше него (41.0 мг/л). Так как всего значений 10, то можно сказать, что 44.0 мг/л это 10-й персентиль в нашей выборке (т.е. 10% значений меньше него). Это и будет собственным квантилем этого значения.

Таким образом, каждое значение в нашей таблице можно представить в двух разных системах отсчета. Одна — исходная, где концентрация измеряется в обычных единицах, и вторая — относительная, где каждое значение имеет свое место и то, насколько оно велико, показано с помощью собственных квантилей.

Такое представление можно сделать и графически, как на рисунке Figure 1.14. Здесь ось абсцисс показывает концентрацию в мг/л, а ось ординат — сколько значений в процентах меньше данного. Можно видеть, что процесс вычисления квантилей для выборки, который мы рассматривали выше, по сути представляет собой разбиение оси ординат на равные части. Для примера серые горизонтальные прямые показывают расположение квартилей.

1.4.5 Гистограмма распределения частоты и плотности значений

Гистограмма распределения позволяет оценить плотность значений на различных участках всего интервала. Если для нахождения квантилей мы делили значения на равные порции (разбивали ось Y на рисунке Figure 1.14 на равные участки), то для построения гистограммы распределения нужно разделить на равные участки непосредственно интервал значений (ось X на этом рисунке).

Для начала необходимо построить таблицу частот с помощью следующей процедуры:

1. Разбить весь интервал значений на равные участки или сегменты (англ. *bins*)

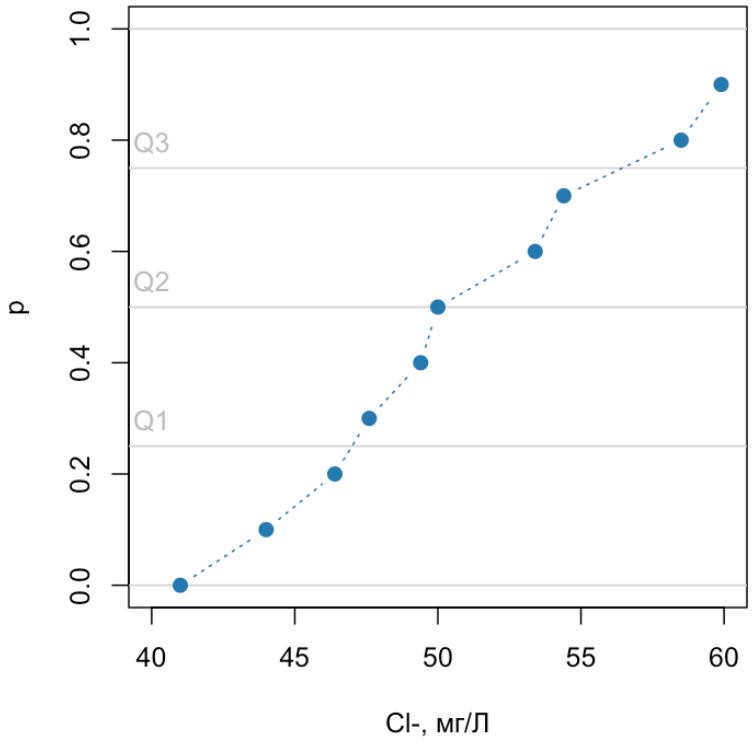


Рис. 1.14. График собственных квантилей в зависимости от оригинальных значений концентрации хлорид-ионов.

2. Посчитать сколько значений попадает в каждый сегмент
3. Нормировать подсчеты, так чтобы их сумма равнялась единице

Если продолжить работу со значениями концентраций хлорид-ионов, мы можем заметить, что наименьшее значение в нашей выборке равно 41.0, а наибольшее — 59.9. Возьмем для простоты чуть больший диапазон $[40, 60)$ и разобьем его на три равных части: [40, 45), [45, 50), [50, 44), [55, 60). Здесь форма скобки означает включается ли значение в этот интервал, или нет. Так, если измеренное значение равно ровно 50 мг/л, то оно попадет в третий интервал, а не во второй. Но если оно равно 49.999999 то оно окажется во втором интервале.

Далее мы можем посчитать число значений в каждом сегменте и построить следующую таблицу:

Интервал	Число значений	Частота	Плотность
[40, 45)	2	0.2	0.04
[45, 50)	3	0.3	0.06
[50, 55)	3	0.3	0.06
[55, 60)	2	0.2	0.04

Для того, чтобы отобразить эту таблицу графически воспользуемся столбцовой диаграммой, в которой ширина столбцов равна ширине сегментов, а высота — числу значений в каждом сегменте (или частоте). Пример такого графика показан на рисунке Figure 1.15.

Из графика, как и из таблицы, можно заметить, что частота значений в середине интервала больше чем на его краях. Это свойство многих случайных чисел — группироваться посередине — позволяет описывать их одной статистикой — положением центра. Для непараметрических методов такой статистикой является медиана.

Гистограмма распределения частоты значений, которую мы только что построили, имеет один недостаток — высота ее столбцов напрямую зависит от числа разбиений. Скажем, если мы увеличим число разбиений в два раза, частота в каждом интервале уменьшится в среднем тоже в два раза, т.е. столбцы гистограммы станут гораздо ниже, при этом общая ширина всего интервала значений останется неизменной. Для того, чтобы избавиться от этой проблемы можно построить распределение плотности частоты.

Плотность — это удельная частота, посчитанная на единицу измерения. Скажем в нашем случае это средняя частота на один мг/л. Таким образом, плотность можно посчитать разделив частоту на ширину интервала, как показано в четвертом столбце таблицы. Если увеличить, или уменьшить число сегментов в это случае, средняя плотность не изменится.

Одно из важных характеристик гистограммы распределения плотности частот — ее независимость ни от единиц измерения, ни от размера выборки, ни от числа сегментов. Какими бы не были эти три параметра,

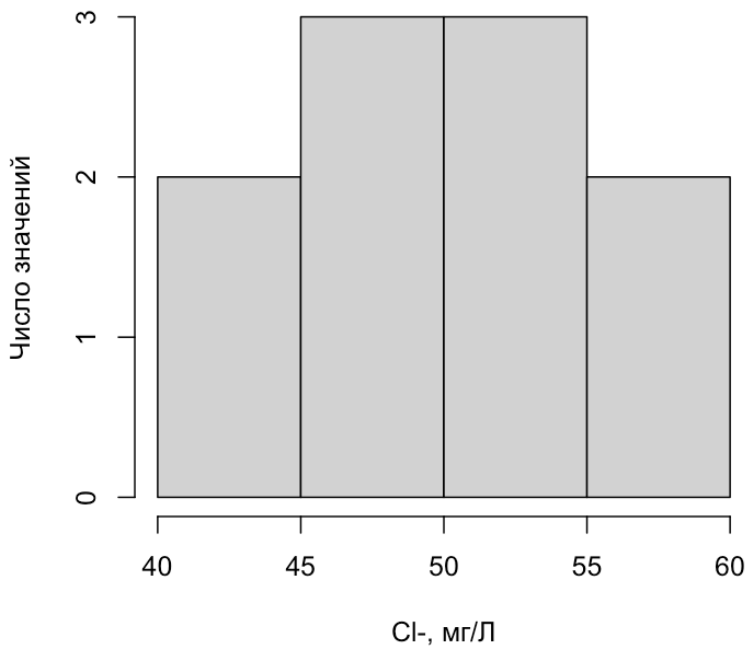


Рис. 1.15. Гистограмма распределения значений значений концентрации хлорид-ионов.

площадь каждого отдельного столбца — это частота значений, встречающихся в соответствующем сегменте. Соответственно, общая площадь столбцов всегда равна единице.

Для визуальной оценки распределения значений нет никакой разницы, что использовать в качестве высоты столбцов — число значений, частоту или ее плотность. Но плотность важна, если мы хотим описать ее с помощью теоретической функции — распределения плотности вероятностей, о которой пойдет речь в следующем разделе.

1.4.6 Параметрическое описание данных, вероятность и частота

Теоретическое распределение — это математическая функция, которая показывает, как должна меняться плотность на различных участках значений (обычно обозначается как $f(x)$). Представьте себе, что в вашей выборке бесконечное число значений (ну или просто очень большое) и число сегментов, на которые вы разбиваете интервал значений, тоже очень велико. Такое гипотетическое разбиение и описывает эта функция.

Но, также как и для того, чтобы построить параболу, вам не нужно бесконечное число точек, а достаточно лишь трех (ну или чуть больше, чтобы учесть их случайную вариацию), то и для того, чтобы описать теоретической кривой экспериментальное распределение плотности частот, не нужно большого числа измерений. По сути, речь идет об аппроксимации экспериментальных точек (в данном случае плотности частот) с помощью математической функции — своего рода нелинейная регрессия.

Необходимо, однако, строго разделять, что гистограмма, построенная по экспериментальным данным, и теоретическая функция распределения описывают не совсем одно и то же. В первом случае мы имеем дело с частотой, тогда как во-втором — с вероятностью. В чем разница?

Частота — это наблюдаемое число исходов какого-то эксперимента. Если вы подкинете монету 12 раз и получите 3 решки, то частота выпадения решки будет 25%. Если вы измерите концентрацию железа в 12 стаканах воды и в 9 из них она будет больше 0.3 мг/л, то частота такого события 75%.

Вероятность — это теоретическое, ожидаемое значение числа исходов. Можно сказать, что вероятность — это ожидаемая частота при повторении эксперимента бесконечное число раз.

Соответственно, когда мы говорим о теоретических предположениях и гипотезах, мы говорим о вероятности. Вероятность всегда относится к событиям, которые еще не произошли. Если же речь идет о наблюдаемых явлениях, которые уже случились, то для них мы используем термин “частота”.

Рассмотрим самое простейшее теоретическое распределение — равновероятное (англ. *uniform* или *even*). При таком распределение любой исход имеет одинаковый шанс, который равен отношению единицы к общему числу исходов. Скажем если мы бросаем монету и она сбалансирована, то вероятность получить решку, $P = 1/2 = 0.5$. Если мы бросаем кость, то общее число исходов равно шести, поэтому вероятность получить, например, тройку $P = 1/6 = 0.166667$.

Что делать, если число исходов гораздо больше? Скажем, мы измеряем pH водопроводной воды с точностью до одной десятой, чему равна вероятность получить pH = 7.5? Предположим, что мы не ожидаем, что из крана потечет геотермальная вода с очень низким pH. Ограничим диапазон возможных значений от 6.5 до 8.5 и будем считать, что любое значение из этого диапазона равновероятно (на самом деле, конечно же, это не так). Тогда общее число исходов равно (если посчитать все значения от 6.5 до 8.5 с шагом 0.1) = 21, т.е. вероятность получить измеренное значение pH = 7.5 равно $P = 1/21 = 0.048$ или 4.8%, если, на самом деле, все эти исходы равновероятны.

> *Попробуйте воспользоваться этим примером и посчитать следующие вероятности: а) что полученное значение будет меньше 7.5; б) что полученное значение будет больше 8. Теперь представьте, что точность измерения составляет одну сотую (0.01). Какова вероятность, что измеренное вами значение будет равно 7.49?*

Если мы продолжим эти рассуждения и представим, что pH измеряется с очень (бесконечно) высокой точностью, то мы увидим, что получить какой-то конкретный точный результат невозможно. Дело в том, что в этом случае число уникальных исходов бесконечно велико и вероятность получить один из них равна $P = 1/\infty = 0$.

Но если речь идет об интервале значений, скажем 7.0 ± 0.1 тогда вероятность такого исхода можно посчитать как площадь прямоугольника шириной 0.2 и высотой, равной обратному значению ширины всего интервала, $1/(8.5 - 6.5) = 0.5$, т.е. искомая вероятность $P(6.9 \geq x > 7.1) = 0.2 \times 0.5 = 0.1$ или 10%.

Чтобы понять логику вычислений, посмотрим на график, изображенный на рисунке Figure 1.16. Черная линия на этом графике — это теоретическая функция, описывающая плотность вероятностей на всем участке возможных значений pH от 6.5 до 8.5. Площадь под этой линией равна единице, что соответствует вероятности получить любое значение из этого диапазона (как бы мы не измеряли, полученный результат всегда будет от 6.5 до 8.5). Площадь для искомого сегмента показана на рисунке с помощью серого прямоугольника — это вероятность получить значение pH от 6.9 до 7.1. Площадь этого прямоугольника и равна 0.1, что соответствует вероятности получить значение в этом ограниченном диапазоне.

Другими словами, для того, чтобы вычислить вероятность получения события из какого-то диапазона (например, что измеренное значение будет лежать в нужном интервале), нужно вычислить площадь под кривой, которую описывает теоретическое распределение плотности вероятностей для этих событий. Что мы и сделали выше. Впрочем, чтобы облегчить нам жизнь, в любом статистическом пакете реализована другая функция, которая позволяет вычислять вероятности напрямую, без вычисления площадей. Мы поговорим о ней в секции, посвященной использованию R.

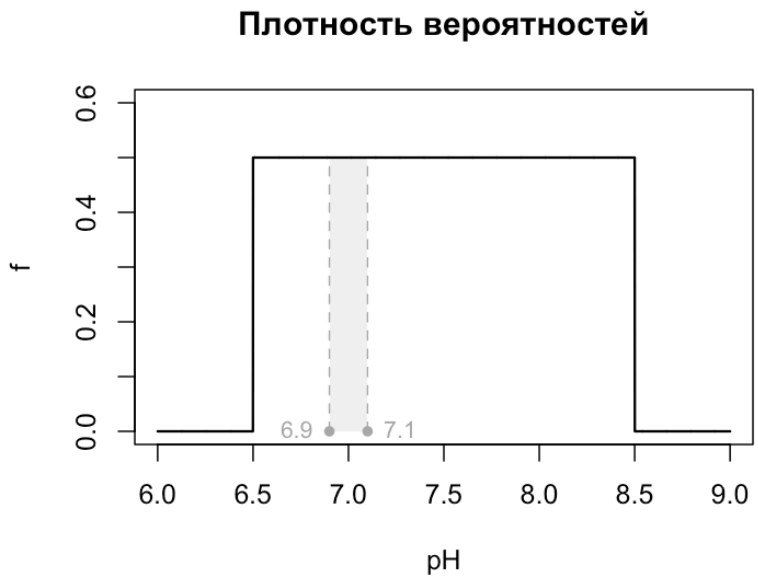


Рис. 1.16. Теоретическая функция плотности вероятностей для равновероятного распределения.

Во многих статистических программах есть возможность эмулировать случайную выборку, взятую из генеральной совокупности со значениями, распределенными согласно какому-то теоретическому распределению. Это делается с помощью так называемых генераторов случайных чисел. Для равновероятного распределения в R можно воспользоваться функцией `runif(n, min, max)`, где первый аргумент — это число значений в выборке, а следующие два — границы значений. Это хороший способ сымитировать различные явления и эксперименты, а также сравнить теоретические ожидания и эмуляцию эмпирических результатов.

Графики на рисунке Figure 1.17 сделаны с помощью такого генератора. Синяя линия на всех графиках показывает ожидаемую, теоретическую плотность вероятностей, а столбцы — гистограмму распределений плотности частот значений, полученных для выборки из генератора случайных чисел. Каждый график соответствует разным размерам выборки: 10, 30, 100 и 1000.

Можно заметить, что с ростом размера выборки уменьшается и разница между теоретической кривой и эмпирическими результатами. Это происходит в первую очередь потому, что для гистограммы необходимо достаточно значений, чтобы накопить полученную информацию в каждом столбце. В случае, когда интервал разбивается на 8 сегментов (как на картинке), а число значений всего 10, любое отклонение будет выглядеть как выброс. Поэтому не рекомендуется оценивать распределение экспериментальных данных с помощью гистограммы, если число значений меньше 30.

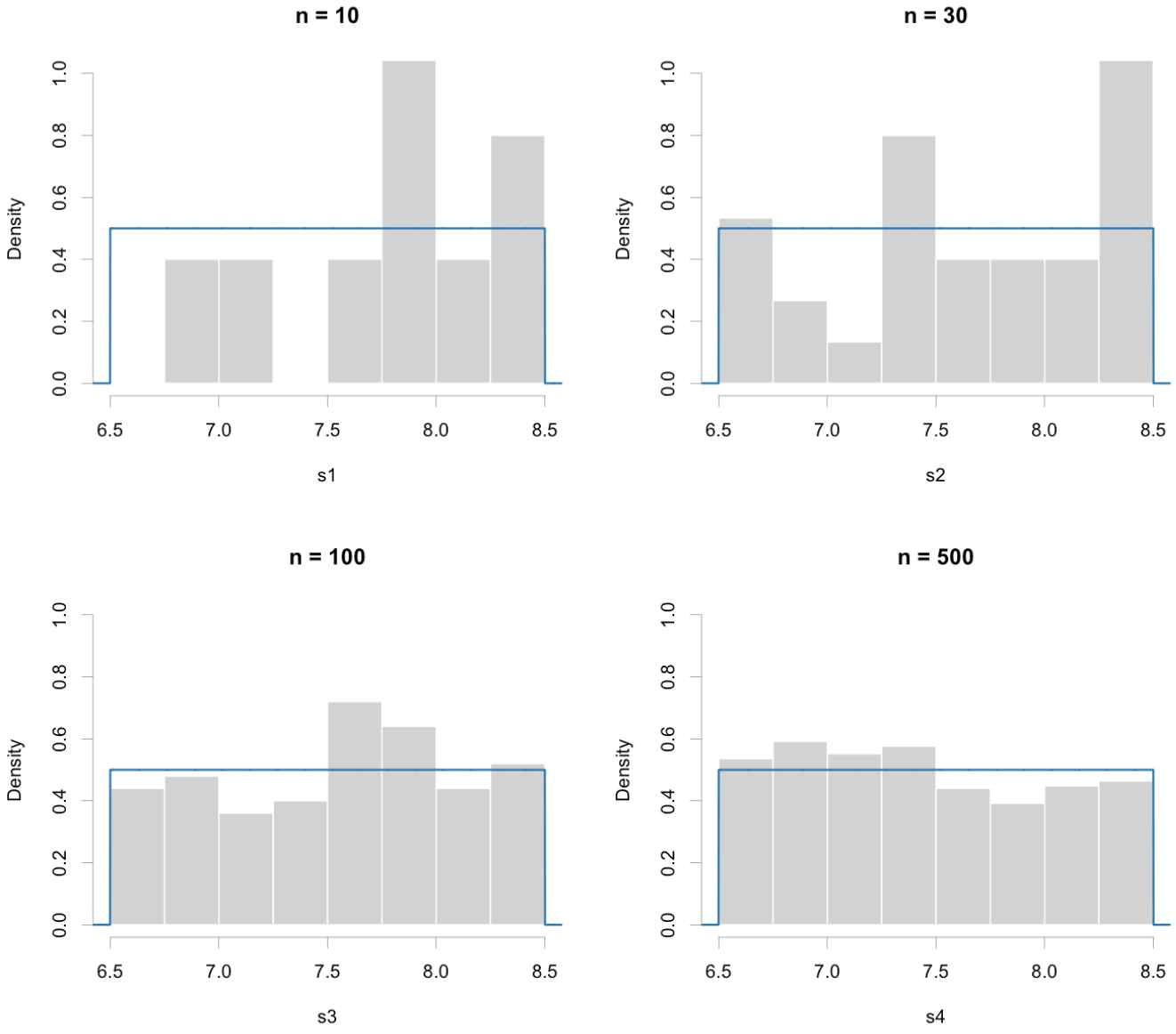


Рис. 1.17. Теоретическая функция плотности вероятностей для равновероятного распределения и эмпирические распределения плотности частот, полученные для выборок разного размера.

1.4.7 Нормальное распределение

В предыдущем подразделе, в примере с кислотностью водопроводной воды, мы упомянули, что невозможно, чтобы измеренные значения были равновероятны. Это верно и для многих других экспериментов и наблюдений. Скажем сколько человек с IQ > 160 вы знаете? Я, навскидку, помню двух — Альберт Эйнштейн и Шерон Стоун. Конечно, во всем мире их гораздо больше, но все же такой индекс интеллекта встречается очень редко. Как и индекс меньше 90, если говорить о взрослых людях без серьезных проблем со здоровьем. Большинство людей имеет значение этого индекса между 100 и 120. То же самое касается роста людей, размеров животных, повторных измерений, сделанных для одного и того же образца и многих других величин. Почти все по-настоящему случайные величины, которые не ограничены условиями измерения, следуют теоретическому распределению, которое называют нормальным, или Гауссовым, по имени немецкого математика, одним из первых описавшего его.

Функция распределения плотности вероятностей нормального распределения выглядит следующим образом:

$$f(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \frac{1}{e^{\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}}$$

Попробуем понять, что показывает эта функция. Во-первых, очевидно, что это функция одной переменной, x, — это как раз измеренное, или наблюдаемое значение. Во-вторых, можно видеть, что эта функция содержит много констант ($\pi, e, 2$) и два параметра — μ и σ. Другими словами, вид нормального распределения зависит только от этих двух параметров.

Рассмотрим, что случится, если измеренное значение x будет точно равно параметру μ. Подставляя $x = \mu$ в наше соотношение, можно видеть, что показатель экспоненты будет равен нулю, т.е. сама экспонента будет равна единице. Что даст нам точное значение плотности вероятности в этой точке:

$$f(x = \mu) = \frac{1}{\sqrt{2\pi\sigma^2}}$$

Далее мы начнем менять x, делая его меньше, или больше μ. Так как разность между x и μ возводится в квадрат, то оба этих случая дают ту же самую плотность, т.е. наше распределение абсолютно симметрично относительно точки $x = \mu$. Так вот, чем больше разница $(x - \mu)$ тем больше будет показатель экспоненты и тем меньше сама плотность.

Другими словами, μ — это центр нормального распределения, точка, где плотность является максимальной. Этот параметр называют также *математическим ожиданием*. Для нормального распределения он также имеет название *среднего арифметического значения*, или просто среднее значение. Вычислить его для популяции можно с помощью формулы:

$$\mu(x) = \frac{1}{N}\sum_{i=1}^{N} x_i$$

В случае выборки можно сделать оценку этого параметра для данной выборки:

$$m(x) = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Надо сказать, что m никогда не будет точно равно μ, и чем меньше размер выборки, тем большим может быть расхождение между ними. Каким образом разрешить эту дилемму, мы поговорим в следующем разделе.

Скорость убывания плотности регулируется параметром σ, который носит название *стандартного отклонения* (англ. *standard deviation*). Чем больше значение сигма — тем медленнее убывает плотность и наоборот, другими словами этот параметр показывает, как далеко значения могут располагаться от центра: чем дальше от центра, тем меньше шанс получить такое значение.

Квадрат стандартного отклонения называется *дисперсией* (англ. *variance*) и его можно вычислить для генеральной совокупности, как среднеквадратичное отклонение значений от центра:

$$\sigma^2(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

Оценку дисперсии для выборки можно сделать с помощью следующего соотношения:

$$s^2(x) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - m)^2$$

Заметим, что в данном случае сумма квадратов расстояний от значений до среднего делится на $(n-1)$, а не на n. Это связано с тем, что одна и та же выборка значений использовалась сначала для оценки m, а затем для вычисления s^2. Т.е., другими словами, значения в выборке при вычислении дисперсии не являются полностью независимыми друг от друга, любое значение можно вычислить, зная среднее и оставшиеся значения.

Число независимых значений в статистике называют числом *степеней свободы* (англ. *degrees of freedom*), при вычислении дисперсии для выборки это число и равно $n-1$. Такая ситуация, когда один и тот же набор данных используется на нескольких этапах вычислений, и каждый следующий этап зависит от результата предыдущего, приводит к потере степеней свободы.

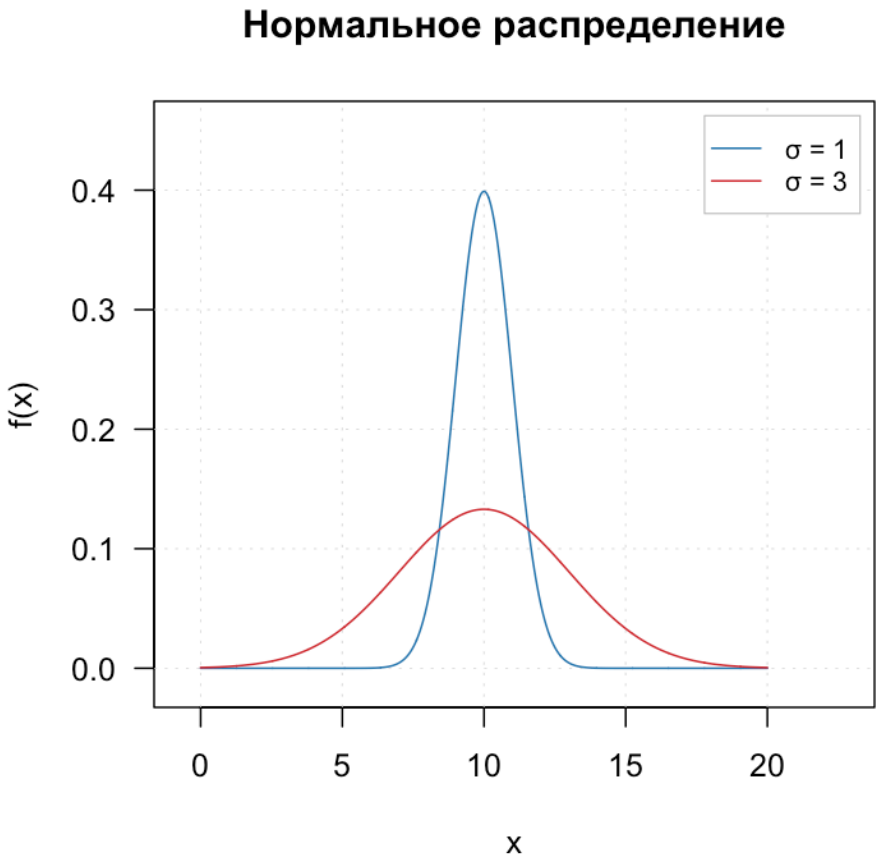


Рис. 1.18. Теоретическая функция плотности вероятностей для нормального распределения ($\mu = 10$).

Рисунок Figure 1.18 показывает график распределения плотности вероятностей для $\mu(x) = 10$ и $\sigma(x) = 1$ (синяя линия) и $\sigma(x) = 3$ (красная линия). Как можно заметить, пик распределения приходится как раз на значение $x = \mu(x) = 10$. Высота же этого пика, как и ширина самой кривой, зависит от $\sigma(x)$ — там где стандартное отклонение больше, разброс возможных значений шире. При этом площадь под каждой кривой равна единице.

Какова вероятность того, что случайное значение из генеральной совокупности будет находится в определенных пределах, например не дальше одного стандартного отклонения от центра? Мы уже знаем, что вероятность можно вычислить, как площадь под кривой распределения. В принципе, если вы помните основы математического анализа, то можно попробовать сделать это самостоятельно, вооружившись листком бумаги и карандашом (точный вид функции вам уже известен).

Однако, это можно сделать быстрее. Во-первых, для нормального распределения есть правило 68-95-99, которое имеет смысл запомнить. Это правило дает примерную оценку вероятности получить значение в пределах одного стандартного отклонения (68.3%), двух (95.5%) и трех (99.7%). Т.е. почти все значения (в теории 997 из 1000) лежат в пределах трех стандартных отклонений от среднего значения. Это можно заметить и на рисунке Figure 1.18, так для синей кривой плотность для значений меньше 7 (это, как раз, на три стандартных отклонения меньше среднего) и больше 13 (на три стандартных отклонения больше) плотность практически нулевая.

Функция распределения вероятностей

Функция вероятности (англ. *cumulative density function, CDF*), $F(x)$, позволяет для любого значения $x = x_0$ найти вероятность встретить значение меньше данного: $P(x < x_0) = F(x_0)$. Другими словами, она возвращает площадь участка под кривой распределения плотности вероятностей, которая находится слева от заданного значения.

Вернемся снова к примеру с кислотностью водопроводной воды. Пусть известно, что кислотность распределена нормально со средним значением $\mu = 7.5$ и стандартным отклонением $\sigma = 0.25$. Какова вероятность того, что если мы возьмем пробу воды в этой системе, измеренное значение окажется меньше 7.25? Именно эту вероятность и позволяет получить функция вероятности.

Рисунок Figure 1.19 содержит два графика — распределение плотности вероятностей (слева) и функцию вероятности (справа) для этого примера. Если вспомнить, что площадь участка под кривой вычисляется с помощью определенного интеграла, математически можно записать это следующим образом:

$$P(x < x_0) = F(x) = \int_{-\infty}^{x_0} f(x)dx$$

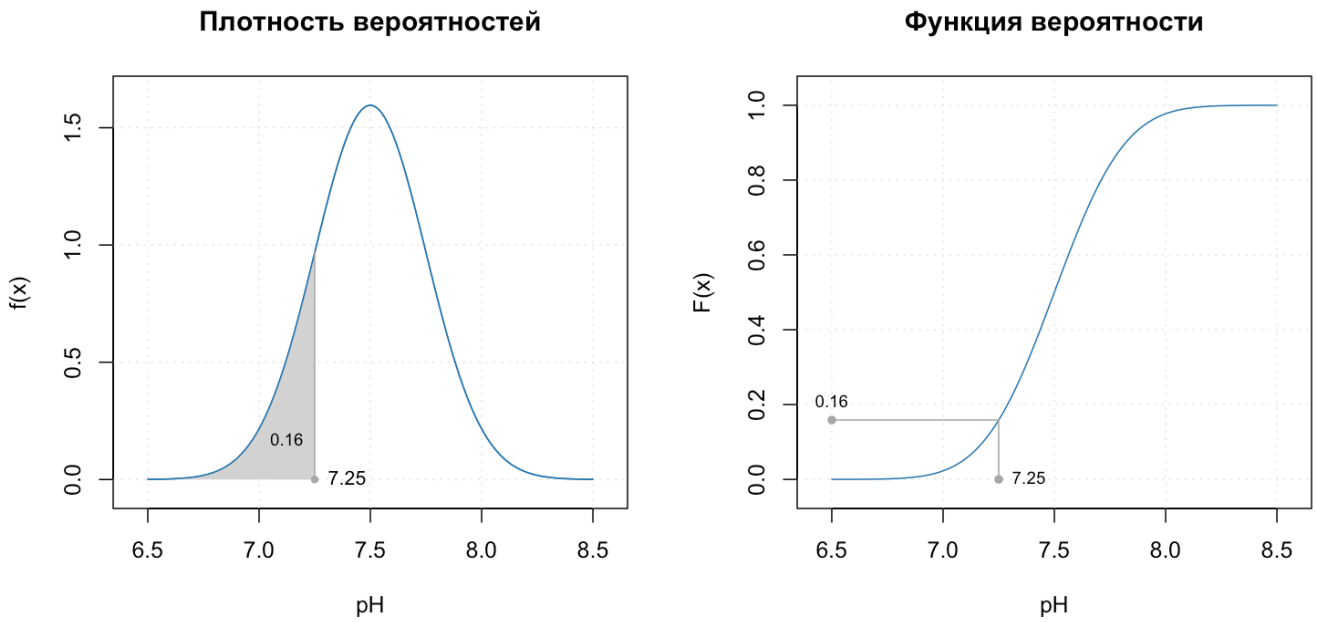


Рис. 1.19. Функция распределения плотности вероятностей (слева) и функция вероятностей (справа) для примера с водой.

Оба графика показывают, что искомая вероятность того, что измеренное значение окажется меньше 7.25, равно 0.16 или 16%. Этот результат можно интерпретировать и по другому: если взять 1000 проб воды из этого источника, то примерно 160 проб (16% от 1000) будут иметь кислотность меньше 7.25. Мы употребили слово “примерно” потому, что вероятность эта — теоретическая, на практике она будет наблюдаться точно, если число проб по-настоящему велико.

Если нужно вычислить вероятность для какого-то интервала значений, нужно просто воспользоваться функцией вероятности для правой границы интервала и вычесть из нее значение функции на левой границы:

$$P(a \leq x < b) = F(b) - F(a)$$

Нужно лишь помнить, что функция вероятностей всегда возвращает площадь левой части распределения, т.е. вероятность найти значение меньше данного.

Часто нужно решить обратную задачу — найти значение, или интервал значений, для заданной вероятности. В этом случае используют обратную функцию, $x = F^{-1}(p)$. Скажем, если нужно найти границу для 5% наименьших значений, то нужно просто воспользоваться этой функцией с аргументом p = 0.05. Другими словами, обратная функция позволяет найти квантили для теоретического распределения.

Предположим, что содержание хлора в пруду распределено нормально со средним значением $\mu(x) = 60$ мг/л и стандартным отклонением $\sigma(x) = 5$ мг/л. Какое значение вы получите, если вычислите значение функции вероятностей для $x = 50$? Какое значение вы получите, если вычислите значение обратной функции с аргументом $p = 0.9775$? Для того, чтобы ответить на этот вопрос, нужно воспользоваться правилом 68-95-99 описанным выше. Вы можете проверить свои ответы чуть позже, когда мы научимся решать такие задачи с помощью R.

1.4.8 Стандартизация данных

Давайте еще раз посмотрим на функцию распределения плотности вероятностей для нормального распределения:

$$f(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \frac{1}{e^{\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}}$$

И сделаем следующую замену:

$$z = \frac{x - \mu}{\sigma}$$

В этом случае выражение для нашей функции будет гораздо проще:

$$f(z) = \frac{1}{\sqrt{2\pi}} \frac{1}{e^{\frac{z^2}{2}}}$$

Можно показать, что среднее значение z всегда равно нулю ($\mu(z) = 0$), а стандартное отклонение — единице ($\sigma(z) = 1$). Распределение значений z называется *стандартным*, а сама операция вычисления этих значений — *стандартизацией*. По сути, стандартизация состоит из двух операций: *центрирования*, которое делает среднее значение равным нулю, и *шкалирования*, которое делает стандартное отклонение равным единице. В хемометрике стандартизацию часто называют автошкалированием (англ. *autoscaling*).

Стандартизация имеет два важных приложения. Во-первых, она позволяет избавится от единиц измерения, так как значения z — безразмерные. Это позволяет сравнивать измерения, сделанные в разных единицах, например, температуру и давление. Во-вторых, она позволяет оценить экстремальность отдельных значений. Например, мы знаем, что 95.5% нормально распределенных значений находятся на расстоянии максимум двух стандартных отклонений от среднего. Т.е. если у нас появится значение, для которого $z = 3$, мы будем знать, что оно маловероятно, и это позволит нам обратить на такое значение более пристальное внимание.

Все это, впрочем, работает только если ваши данные представляют собой случайную выборку, взятую из нормально распределенной генеральной совокупности. Как это проверить?

1.4.9 Тест на нормальность значений

Как мы уже выяснили, гистограмма распределения плотности частот не очень подходит для надежной оценки формы распределения, особенно, если размер выборки небольшой. Однако, есть другой график, который позволяет сделать это визуально — график теоретических и эмпирических квантилей (англ. *quantile quantile plot* или *QQ plot*).

Идея состоит в следующем. Для каждого значения в выборке вычисляется два вида квантилей. Первый — эмпирический, основанный на положении каждого значения в неубывающей последовательности. Второй — теоретический, сделанный с помощью функции вероятности и оценочных параметров распределения. Если значения на самом деле получены из популяции, для которой теоретическое распределение хорошо работает, то эти два вида квантилей должны иметь схожие по величине значения.

Если отобразить величины этих квантилей графически, например, располагая значения теоретических квантилей вдоль оси абсцисс, а эмпирических — вдоль оси ординат, то это и будет графиком квантиль-квантиль. Для нормального распределения построить такой график можно немного проще — по оси ординат откладывают реальные измеренные значения, а по оси абсцисс — число стандартных отклонений на которое данное измерение отличается от среднего (z-значения).

Рисунок 1.20 представляет пример такого графика для значений концентрации хлорид-ионов в 10 образцах воды, использованных нами ранее. Линия показывает идеальную ситуацию, чем ближе точки к этой линии, тем больше шанс, что данные на самом деле взяты из нормально распределенной популяции.

Возьмем к примеру второе значение снизу, 44 мг/л. Так как число значений в нашей выборке не превышает 10, R в этом случае по умолчанию вычисляет собственные квантили, используя довольно непростое выражение (для выборки большего размера оно гораздо проще и похоже на то, что мы использовали, когда обсуждали вычисление квантилей в непараметрическом подходе):

$$p = \frac{i - 3/8}{n + 1/4}$$

Для второго значения имеем:

$$p = \frac{2 - 3/8}{10 + 1/4} = 0.15854$$

Далее, используя обратную функцию вероятности, находим $F^{-1}(0.15854) \approx -1$, которое и используется на графике.

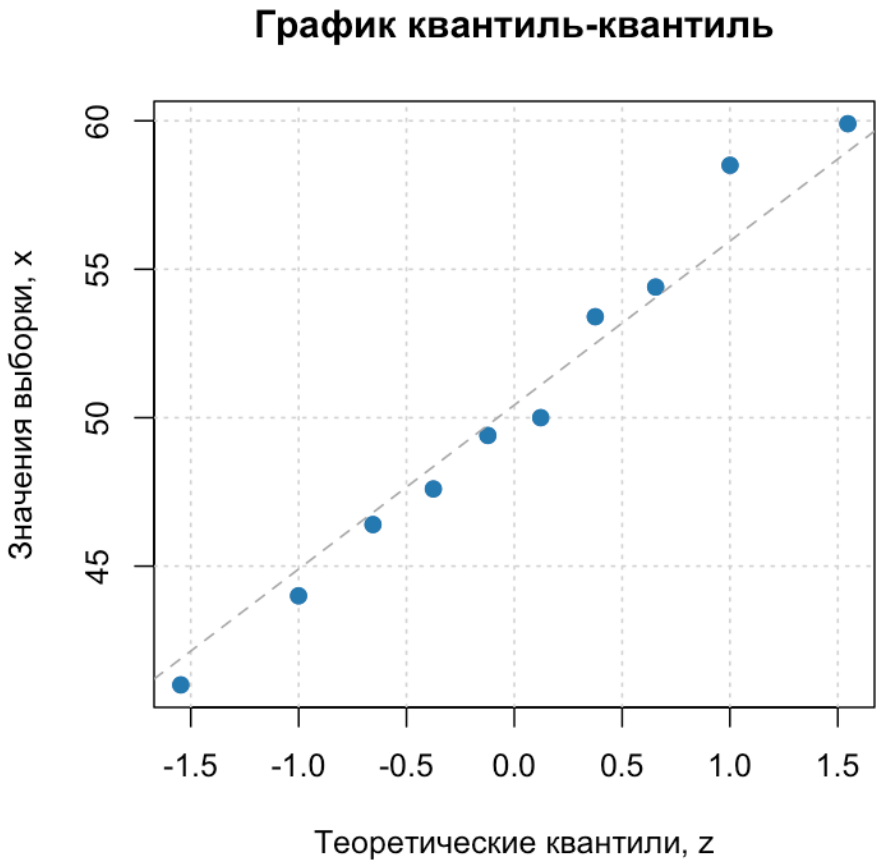


Рис. 1.20. График квантиль-квантиль для значений концентраций хлорид-ионов.

Иногда по виду графика трудно сказать наверняка, насколько хорошо нормальное распределение описывает наши значения. В этом случае визуальный анализ можно подкрепить тестом на нормальность. Для этого мы предполагаем, что наши данные на самом деле взяты из нормально распределенной популяции (это наша основная гипотеза в этом тесте) и оцениваем насколько экстремально выглядит наша выборка для такого предположения.

Эта мера экстремальности оценивается, как вероятность того, что если мы повторим пробоотбор, то новая выборка будет настолько же экстремальна (по отношению к предположению, что популяция распределена нормально), как наша, или более экстремальна. Если эта вероятность мала (обычно меньше 5% или 1%), то гипотезу отвергают и используют другие методы, например, непараметрические.

Существует несколько вариантов такого теста, мы будет пользоваться методом Шапиро и Уилка, реализованном в ряде ПО, включая R. Для нашей выборки этот тест дает вероятность $p = 0.9226$, которая позволяет нам использовать гипотезу о нормальном распределении.

Часто проверку нормальность используют для того, чтобы убедиться, что данные на самом деле случайны. Например, когда исследуют остатки в различных моделях — регрессии, дисперсионном анализе, и т.д. В этом случае мы предполагаем, что полностью случайные данные хорошо описываются нормальным распределением.

1.4.10 Статистический анализ данных в R

Непараметрический анализ

В таблице ниже собраны основные функции, позволяющие вычислить статистики, описывающие вектор значений x без использования каких-либо теоретических знаний о распределении. Как мы обсуждали ранее, основной функцией является функция вычисления квантилей, `quantile()`.

Статистика	Функция и параметры
Минимальное значение	`min(x)`
Максимальное значение	`max(x)`
Медиана	`median(x)`
Квантили	`quantile(x, p)`
Собственные квантили	`ppoints(x)`

Если нужно просто упорядочить значения в векторе, то можно воспользоваться функцией `sort(x)`.

Для построения диаграммы размаха используется функция `boxplot(x)`. Основной аргумент `x` может быть вектором, матрицей или формулой. Про формулы мы поговорим в другом разделе, в случае матрицы функция построит несколько “коробок-с-усами” — по одной на каждый столбец матрицы. Функция имеет довольно большое число аргументов, настоятельно рекомендуем изучить справку внимательно.

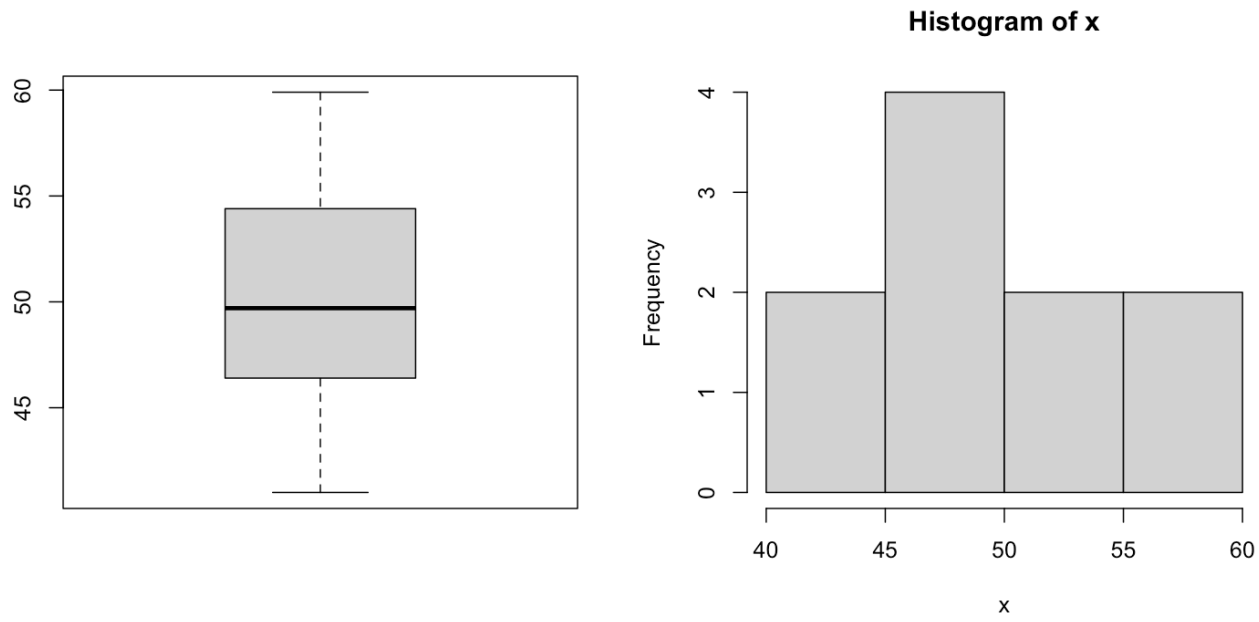
Функция `hist(x)` позволяет построить гистограмму распределения числа значений, или их плотности. По умолчанию число разбиений и их границы вычисляются автоматически. Однако, если нужно, их можно указать с помощью дополнительного аргумента.

Блок с кодом ниже показывает пример использования некоторых из этих функций.

```r
# вектор с исходными значениями (уже в правильном порядке)
x <- c(41.0, 44.0, 46.4, 47.6, 49.4, 50.0, 53.4, 54.4, 58.5, 59.9)

# квартили и пограничные значения
Q1 <- quantile(x, 0.25)
Q2 <- quantile(x, 0.50)
Q3 <- quantile(x, 0.75)
mn <- min(x)
mx <- max(x)

# собираем все статистики вместе
stat <- c(mn, Q1, Q2, Q3, mx)
names(stat) <- c("Min", "Q1", "Q2", "Q3", "Max")
show(stat)
```

```
  Min    Q1    Q2    Q3   Max
41.00 46.70 49.70 54.15 59.90
```

```r
# диаграмма размаха и гистограмма
par(mfrow = c(1, 2))
boxplot(x)
hist(x)
```



Теоретические распределения

В R имеются функции, позволяющие работать с большинством имеющихся теоретических распределений. Для любого распределения доступны четыре функции:

1. PDF (*Probability Density Function*) — функция распределения плотности вероятностей. Для любого значения x и заданных параметров распределения она вычисляет соответствующую плотность $f(x)$. В R такие функции начинаются с буквы `d` и далее идет сокращенное название распределения. Например, `dunif()` для равномерного распределения или `dnorm()` для нормального. Эти функции можно использовать для построения теоретических кривых, чтобы показать форму распределения и сравнить ее с экспериментально полученным распределением плотности частот.

2. CDF (*Cumulative Distribution Function*) — функция вероятности. Эта функция позволяет получить вероятность, P, того, что новое измерение будет меньше некоторого заданного значения, x. Эта функция позволяет вычислять вероятности для различных интервалов, как будет показано на примерах ниже. В R эти функции начинаются с буквы `p` и далее, как и в случае с PDF, добавляется сокращенное название распределения, например, `punif()`, `pnorm()`.

3. ICDF (*Inverse Cumulative Distribution Function*) — обратная функция вероятности. Эта функция позволяет сделать операцию обратную CDF, а именно, получить значение x для известной вероятности, P. В R эти функции начинаются с буквы `q` (по сути, эта функция вычисляет теоретические квантили

распределения) и далее добавляется сокращенное название распределения, например, `qunif()`, `qnorm()`.

4. Генератор случайных чисел, который позволяет сымитировать получение случайной выборки заданного размера из генеральной совокупности, распределенной по нужному закону. В R такие функции начинаются с буквы `r` и дополняются названием распределения, как показано выше, например, `runif()` и `rnorm()`.

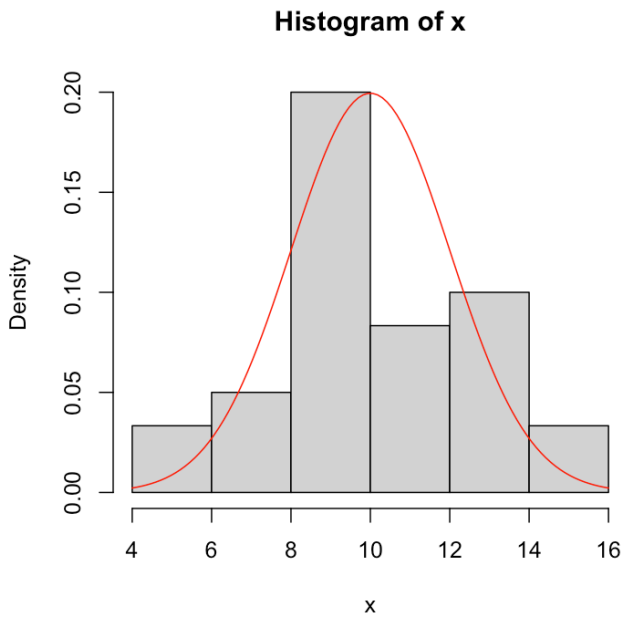
Для каждой из четырех функций нужно задать основной аргумент (вектор значений для PDF и CDF, вектор вероятностей для ICDF, и размер выборки для генератора случайных чисел) и параметры распределения. Например, для равновероятного распределения нужно задать границы интервала, а для нормального — среднее значение и стандартное отклонение. Если параметры не указаны, то R будет использовать стандартные значения, там, где это возможно. Например, для равновероятного распределения будет использоваться отрезок $[0, 1)$ а для нормального распределения — параметры $\mu = 0$ и $\sigma = 1$.

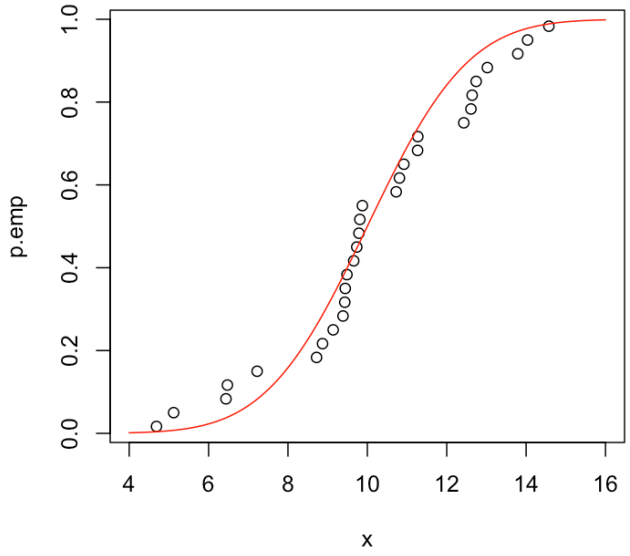
Блок кода ниже делает следующее. Во первых, он извлекает выборку размером $n = 30$ из нормально распределенной популяции с $\mu = 10$ и $\sigma = 2$. Далее, он строит две графика — эмпирическое распределение и собственные квантили, в зависимости от значений выборки. После этого мы вычисляем теоретические значения этих двух статистик и накладываем их поверх эмпирических в виде красных кривых. Параметр `freq` для функции `hist()` отвечает за то, что будет использоваться для высоты столбцов в гистограмме, если он равен `TRUE`, то число значений, а если `FALSE`, то плотность.

Функция `set.seed()` используется для воспроизводимости результатов, чтобы сгенерированные значения были одинаковыми у всех, кто запустит этот код в R. Т.е. сгенерированные значения, на самом деле, не являются полностью случайными. Результат выполнения кода показан на рисунке ниже.

```r
# задать параметры распределения
mu <- 10
sigma <- 2

# задать границы диапазона
left <- mu - 3 * sigma
right <- mu + 3 * sigma

# установить генератор случайных чисел
set.seed(42)

# сгенерировать выборку из 30 значений с указанными параметрами
x <- rnorm(30, mean = mu, sd = sigma)
```

```r
# сортировать значения по возрастанию
x <- sort(x)

# вычислить эмпирические квантили для сгенерированных значений
p.emp <- ppoints(x)

# сгенерировать вектор значений x от mu - 3 * sigma до mu + 3 * sigma
x.theor <- seq(left, right, length.out = 100)

# посчитать плотность и вероятности для этих значений
# и заданных параметров распределения
f <- dnorm(x.theor, mean = mu, sd = sigma)
p.theor <- pnorm(x.theor, mean = mu, sd = sigma)

# построить графики
par(mfrow = c(1, 2))

hist(x, freq = FALSE, xlim = c(left, right))
lines(x.theor, f, col = "red")

plot(x, p.emp, type = "p", xlim = c(left, right))
lines(x.theor, p.theor, col = "red")
```

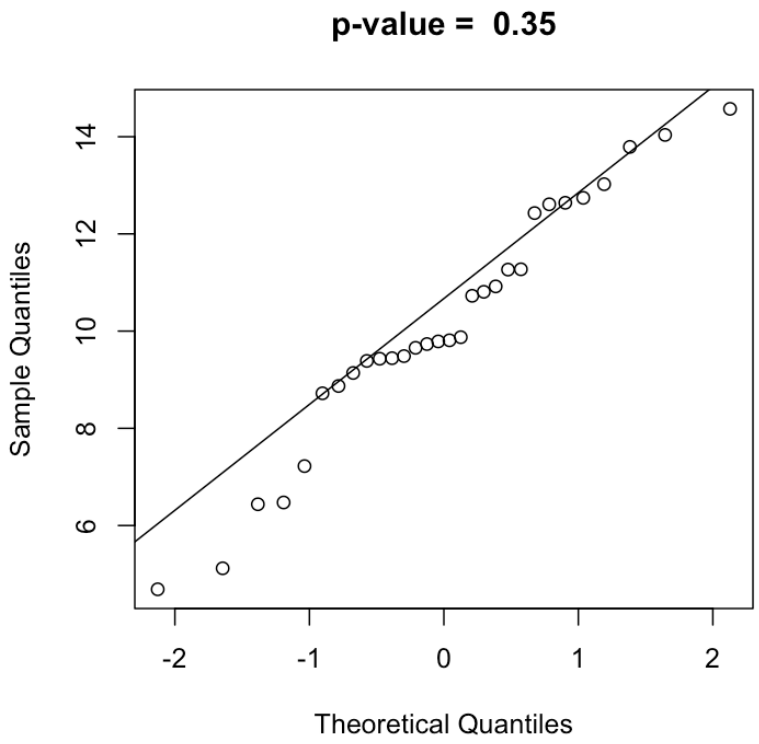




Как можно видеть из графиков, гистограмма распределения не очень хорошо описывается теоретической кривой. Но, если посмотреть на второй график, то соотношение выглядит совсем неплохо. Для того, чтобы быть более уверенным, нужно сделать проверку на нормальность.

Код ниже делает это следующим образом. Сначала мы делаем тест Шапиро-Уилка и сохраняем результат в переменную `res`. Это список, который, помимо всего прочего, содержит искомую вероятность, *p-value*. Далее мы строим график квантиль-квантиль для нормального распределения (функция `qqnorm()`) и добавляем в заголовок этого графика полученную вероятность, округленную до третьего знака после запятой. Функция `qqline()` в конце добавляет идеальную прямую, к которой должны стремиться точки.

```
res <- shapiro.test(x)
qqnorm(x, main = paste("p-value = ", round(res$p.value, 3)))
qqline(x)
```



Как мы можем видеть, не смотря на то, что точки не идеально ложатся на теоретическую прямую, тест показывает, что вероятность получить такую выборку (или хуже) из нормально распределенной генеральной совокупности составляет 35%, что довольно много.

Для оценки параметров нормального распределения на основе значений выборки можно использовать функции `mean()` и `sd()`.

1.5 Сравнение выборок и генеральных совокупностей

В этом разделе мы обсудим, как обобщить информацию, полученную с помощью статистического описания выборки, на соответствующую генеральную совокупность. Формально эту задачу можно представить следующим образом. При описании выборки мы вычисляем различные статистики. Например, если мы используем нормальное распределение, это будут среднее значение, $m(x)$, и стандартное отклонение, $s(x)$. Однако, наша цель — определить соответствующие параметры популяции, $\mu(x)$ и $\sigma(x)$. Можно ли сделать это, и если да, то как?

Начнем с небольшого вычислительного эксперимента. Воспользуемся функцией `rnorm()` чтобы сгенерировать значения для 10 случайных выборок, по 10 значений в каждой ($n = 10$). Затем вычислим среднее значение и стандартное отклонение для каждой выборки. В качестве параметров популяции будем использовать $\mu(x) = 10$ и $\sigma(x) = 2$. Результат эксперимента представлен в таблице ниже:

	1	2	3	4	5	6	7	8	9	10
m	11.09	9.67	9.64	9.27	9.96	10.04	11.08	9.56	10.50	9.83
s	1.67	3.26	2.31	2.23	1.72	2.42	1.39	1.54	1.52	2.14

Как мы можем видеть, ни одна из 10 выборок не имеет статистик, совпадающих с параметрами популяции. Иногда разница довольно мала (например, среднее значение для шестой выборки равно 10.04), иногда различие весьма велико. Можно заметить, что некоторые статистики меньше соответствующего параметра, а некоторые — больше. Проблема в том, что:

1. В реальности выборка у нас обычно одна (а не 10, как в нашем примере)
2. Мы не знаем настоящие значения параметров генеральной совокупности

Тем не менее, решить задачу оценки параметров популяции по известным статистикам выборки можно, причем ничего нового не нужно изобретать. Дело в том, что статистики выборки — это тоже случайные числа, они распределены определенным образом вокруг искомого значения параметра популяции. Если знать, какое теоретическое распределение описывает поведение статистики, то можно сделать следующее:

1. Оценить разброс, или неопределенность статистики, т.е. как далеко она может быть от параметра популяции.
2. Оценить возможность того, что данная статистика получена для выборки из популяции с определенным значением соответствующего параметра.

Собственно, это два основных подхода индуктивной статистики, о которых мы коротко расскажем в этом разделе, используя в качестве примера среднее значение. Чтобы применить эти подходы для другой статистики (дисперсии, корреляции и т.п.) — нужно лишь знать какое теоретическое распределение описывает ее, и как оценить параметры этого распределения. Начнем с простейшего случая.

1.5.1 Среднее значение для одной выборки

Каким образом можно описать поведение средних значений выборок, случайно взятых из одной и той же генеральной совокупности?

Средние значения случайных выборок, $m(x)$ всегда распределены нормально, независимо от того, как распределены сами величины x в популяции. Этот закон носит название *центральной предельной теоремы* и может быть доказан математически. Параметры этого распределения:

$$\mu(m) = \mu(x)$$

$$\sigma(m) = \sigma(x)/\sqrt{n}$$

Т.е. другими словами, насколько далеко среднее значение случайной выборки может находиться от среднего значения генеральной совокупности, зависит от двух вещей — стандартного отклонения самих значений, $\sigma(x)$, и размера выборки, n. Чем больше этот размер, тем выше шанс получить выборку со средним значением близким к среднему значению популяции. Например, для выборок размером $n = 100$ разброс средних значений будет в два раза меньше по сравнению с выборками размера $n = 25$ (попробуйте показать математически, что это на самом деле так).

Параметр $\sigma(m)$ — это стандартное отклонение средних значений выборок. Чтобы не путать его с $\sigma(x)$ — стандартным отклонением для измеренных или наблюдаемых значений, стандартное отклонение, посчитанное для статистик, обычно называют *стандартной ошибкой* (англ. *standard error*). По сути, стандартная ошибка показывает то, насколько далеко статистика выборки (например, среднее значение) может находится от соответствующего параметра генеральной совокупности.

> *Стандартную ошибку и стандартное отклонение очень часто путают. Для того, чтобы избежать путаницы, нужно четко разделять, с какими именно величинами мы имеем дело. Если речь идет об измерениях, или наблюдениях (например, концентрация веществ в различных образцах, или кислотность воды в разных пробах), то разброс таких значений при использовании нормального распределения описывается стандартным отклонением. Но, если мы говорим о статистиках, например, среднем значении выборки, взятой из генеральной совокупности, то разброс статистик описывается стандартной ошибкой.*

Распределение Стьюдента

К сожалению, чтобы использовать нормальное распределение для описания средних значений выборки, нужно либо чтобы выборка была достаточно большой (сотни значений), либо требуется точно знать стандартное отклонение генеральной совокупности, $\sigma(x)$. И то, и другое зачастую невозможно.

Однако, в свое время, Уильям Госсет — математик и химик, работающий на пивоваренном заводе Гиннесс в начале 20 века, придумал, как обойти это ограничение. Его идея заключалась в том, чтобы использовать новое теоретическое распределение, своего рода нормальное распределение с поправкой на маленький размер выборки. В этом распределении разброс значений напрямую связан с размером выборки — чем она меньше, тем разброс, а значит и неопределенность, — больше.

Это своего рода штраф, который мы платим за возможность работы с небольшими выборками. С одной стороны, мы экономим время и деньги на пробоотборе и измерениях, а с другой — получаем большую неопределенность в оценке параметров популяции.

Такое распределение носит название *распределение Стьюдента* или *t-распределение* (англ. *Student's t-distribution*). Дело в том, что Госсет публиковал свои статьи под псевдонимом Стьюдент, отсюда и взялось такое название. В общем случае у распределения три параметра:

1. Среднее или центр распределения
2. Стандартное отклонение, или мера вариации значений
3. Число степеней свободы

Первые два параметра, по сути, идентичны параметрам нормального распределения. Третий параметр — это число независимых значений в выборке, он влияет на ширину распределения при фиксированном стандартном отклонении.

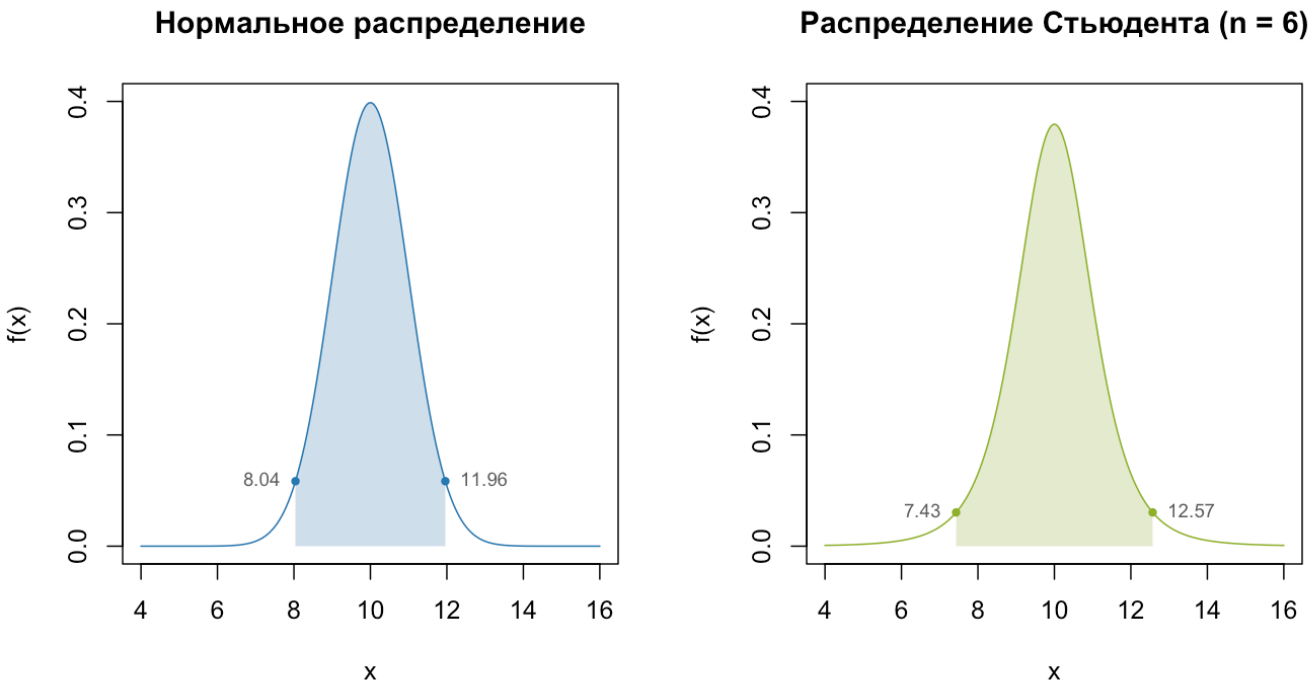


Рис. 1.21. Нормальное распределение и распределение Стьюдента с разным число степеней свободы.

На рисунке 1.21 представлены кривые распределения плотности вероятностей для двух распределений. Во всех случаях параметры распределения заданы, как $\mu = 10$, $\sigma = 1$. График слева показывает нормальное распределение с этими параметрами, закрашенная область соответствует 95% значений распределенных вокруг центра. Как можно видеть, границы этой области равны 8.04 и 11.96, что соответствует нашим

знаниям о нормальном распределении (95% значений находятся на максимальном расстоянии $\pm 1.96 \times \sigma$ от центра).

График справа показывает кривую плотности для распределения Стьюдента с числом степеней свободы, DoF = 5 (т.е. для выборки с $n = 6$ значениями). Он также имеет закрашенную область с 95% значений, которая, как можно видеть, охватывает более широкий диапазон значений, а именно [7.43, 12.57]. Т.е. в случае распределения Стьюдента значения имеют больший разброс, не смотря на то, что стандартное отклонение точно такое же, как у нормального распределения.

В данном случае, чтобы охватить 95% значений нужно "отойти" от центра на 2.57 стандартных отклонений (а не на 1.96, как в случае с нормальным распределением). Для выборки меньшего размера этот разброс будет еще больше. Например, для DoF = 2 ($n = 3$) интервал для 95% значений лежит в диапазоне [5.7, 14.3], или 4.3 стандартных отклонения.

Обычно в ПО используют стандартное распределение Стьюдента, которое аналогично стандартному нормальному распределению, т.е. описывает распределение значений с нулевым средним и единичным стандартным отклонением. Такие значения называются *t-значения* (англ. *t-values*), это прямой аналог z-значений в нормальном распределении. В этом случае для его нахождения нужного t-значения необходимо воспользоваться обратной функцией вероятности для распределения Стьюдента:

$$t_{p,DoF} = F^{-1}(p, DoF)$$

где p — это вероятность встретить значение на расстоянии равном, или меньшим, чем t стандартных отклонений (область под кривой распределения плотности, которая находится слева от значения t).

Возьмем для примера последний график на рисунке 1.21, найдем t-значение для правой границы интервала. Так как внутри интервала находятся 95% значений и распределение симметрично, то на краях суммарно 5% значений — по 2.5% на каждом. Т.е. p для правой границы равно $0.95 + 0.025 = 0.975$. Число степеней свободы в этом случае равно 5. Т.е.

$$t = F^{-1}(0.975, 5) \approx 2.57$$

Можете проверить правильность вычисления воспользовавшись функцией `qt()` в R. Помня о том, что $\mu = 10$ и $\sigma = 1$ получаем границу правого интервала, как $10 + 2.57 \times 1 = 12.57$.

Доверительный интервал для среднего значения

При использовании распределения Стьюдента для описания среднего значения выборки размера n, выбранной случайным образом из генеральной совокупности с истинным средним значением $\mu(x)$,

параметры определения оценивают следующим образом:

1. Центр распределения: среднее значение популяции, $\mu(x)$
2. Стандартная ошибка: $s(x)/\sqrt{n}$, где $s(x)$ — стандартное отклонение для данной выборки
3. Число степеней свободы, DoF = $n - 1$

Формально взаимоотношение между μ и m можно записать следующим образом:

$$m(x) = \mu(x) + t_{p,n-1}\frac{s(x)}{\sqrt{n}}$$

Как можно заметить, большинство значений в этом соотношении — это статистики выборки. Однако, есть и два неизвестных. Первое — это, собственно $\mu(x)$, среднее значение генеральной совокупности. Второе — это вероятность, p, встретить значение $m(x)$, отстоящее от $\mu(x)$ на расстоянии $t\frac{s(x)}{\sqrt{n}}$, или t стандартных ошибок, или дальше. Собственно, чтобы решить это уравнение, нужно сделать предположение для одного из этих двух неизвестных и найти второе.

1. Если мы предполагаем вероятность известной, то мы можем найти $\mu(x)$ с точностью до этой вероятности. Этот подход называется вычислением *доверительного интервала.*
2. Если мы предполагаем параметр $\mu(x)$ известным, то мы можем оценить шанс, который у нас был до пробоотбора, получить выборку с данными свойствами из генеральной совокупности с известным значением $\mu(x)$. Этот подход называется *тестированием гипотезы* о среднем значении.

Рассмотрим для начала, как посчитать доверительный интервал. Во-первых, нужно определиться со значением *доверительной вероятности* (англ. *confidence level*), с которым вы хотите работать. Доверительная вероятность — это своего рода уровень воспроизводимости результатов, его смысл мы обсудим чуть позже. Пока лишь определимся, что мы будем работать с 95% вероятностью.

Рассмотрим 95% всех значений $m(x)$, распределенных симметрично вокруг $\mu(x)$. Левую и правую границы этого интервала можно посчитать как:

$$m_{left} = \mu + t_{0.025,n-1}\frac{s}{\sqrt{n}}$$

$$m_{right} = \mu + t_{0.975,n-1}\frac{s}{\sqrt{n}}$$

Но, так как распределение Стьюдента симметрично относительно центра, мы можем сделать замену $t = -t_{0.025,n-1} = t_{0.975,n-1}$, и переписать эти соотношения более компактно (не забывая, что значение t, на самом деле, зависит от вероятности и числа степеней свободы):

$$m = \mu \pm t\frac{s}{\sqrt{n}}$$

Другими словами, величина:

$$\pm t\frac{s}{\sqrt{n}}$$

и есть та самая мера неопределенности, по-английски ее часто называют *error margin*. Она показывает максимальное расстояние, на которое среднее значение выборки может отличаться от среднего значения популяции для данной доверительной вероятности. Например, для вероятности 95%, если взять 100 выборок из одной и той же генеральной совокупности, то, в теории, 95 из них будут удовлетворять этому условию, а 5 — нет.

Или, по-другому, если мы работаем с доверительной вероятностью 95%, то, когда мы извлекаем новую выборку из генеральной совокупности, у нас есть 95% шанс, что среднее значение этой выборки будет не далее $\pm t\frac{s}{\sqrt{n}}$ от среднего значения совокупности.

Если наша выборка, на самом деле, удовлетворяет этому условию, то:

$$\mu \in m \pm t\frac{s}{\sqrt{n}}$$

т.е. значение μ будет находится внутри интервала $m \pm t\frac{s}{\sqrt{n}}$.

Этот интервал и называется *доверительным интервалом*. Он позволяет оценить возможное значение среднего значения генеральной совокупности по случайной выборке. Чем больше выборка, тем меньше будет этот интервал, и тем точнее мы сможем сделать эту оценку. Процедура вычисления доверительного интервала также проиллюстрирована на рисунке 1.22.

Значение доверительной вероятности выбирается в самом начале, когда вы только планируете эксперимент, потому что, по сути — это вероятность получить выборку с нужными свойствами. Если эта вероятность равна 95%, то у вас есть 5% шанс получить выборку, доверительный интервал которой не будет содержать значений $\mu(x)$. Это значение называется уровнем значимости (англ. *significance level*), мы о нем поговорим подробнее в следующем подразделе.

А пока давайте вернемся к нашему примеру с содержанием хлорид-ионов в 10 пробах воды, взятых из одного и того же источника, и попробуем оценить, чему равно среднее значение концентрации в самом источнике. Будем использовать доверительную вероятность 95%.

Приведем еще раз измеренные значения для выборки:

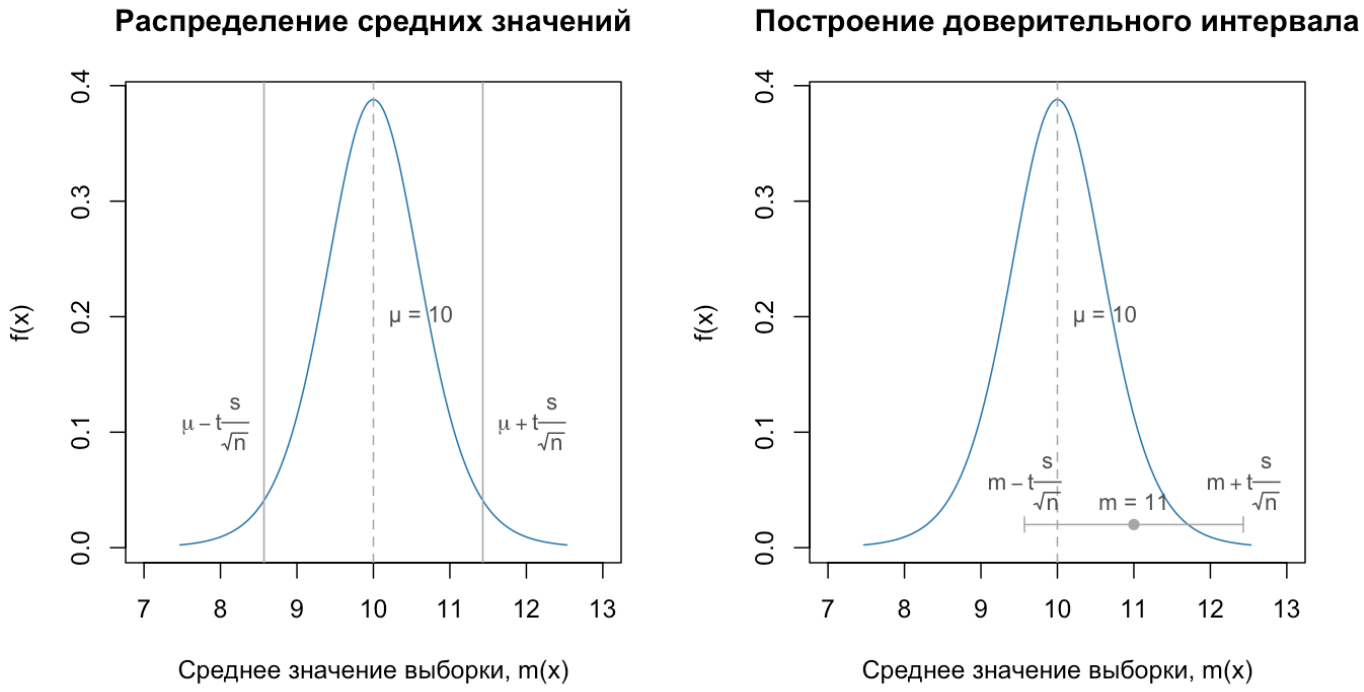


Рис. 1.22. Иллюстрация вычисления доверительного интервала. Слева: распределение средних значений выборки относительно $\mu(x)$. Справа: схема вычисления доверительного интервала для данной выборки. Как можно видеть, в данном случае, не смотря на то, что среднее значение выборки ($m = 11$) довольно далеко от среднего значения популяции ($\mu = 10$), доверительный интервал будет содержать μ внутри.

#	1	2	3	4	5	6	7	8	9	10
Cl-, мг/л	41.0	44.0	46.4	47.6	49.4	50.0	53.4	54.4	58.5	59.9

Простой подсчет дает нам следующие статистики: $m(x) = 50.5$ и $s(x) = 6.1$. Очевидно, что $n = 10$, т.е. $DoF = 9$. Найдем сначала t-значение для данной вероятности (точнее для правой границы будущего интервала):

$$t = F^{-1}(0.975, 9) = 2.26$$

Теперь мы можем найти величину неопределенности (*error margin*):

$$em = \pm t\frac{s}{\sqrt{n}} = \pm 2.26\frac{6.1}{\sqrt{10}} \approx \pm 4.4$$

Теперь строим интервал вокруг значения $m(x)$:

$$\mu(x) = 50.5 \pm 4.4 = [46.1, 54.9]$$

Т.е. с доверительной вероятностью 95% мы можем полагать, что средняя концентрация хлорид-ионов в источнике находится в интервале от 46.1 мг/л до 54.9 мг/л.

Тестирование гипотезы о среднем значении

Приведем еще раз выражение, описывающее взаимоотношение между средним значением генеральной совокупности, $\mu(x)$ и средним значением выборки, $m(x)$ в компактной форме, опуская некоторые индексы (но помня о них).

$$m = \mu + t\frac{s}{\sqrt{n}}$$

Рассмотрим теперь немного другую ситуацию, предположим, что у нас есть знание о том, чему равно $\mu(x)$. На первый взгляд это звучит немного странно, но на самом деле, часто при решении подобных задач у нас есть некоторые референсные значения. И, собственно, наша задача заключается в том, чтобы оценить насколько реальное значение больше, или меньше референсного.

Скажем, если речь идет от загрязненности водоема определенными химикатами, мы можем использовать допустимый лимит в качестве такого референсного значения. Предположим, для нашего случая

концентрация хлорид-ионов не должна превышать значение 46.0 мг/л. Таким образом, наше предположение относительно средней концентрации во всем водоеме будет следующим:

$$\mu(x) \leq 46.0$$

Такое предположение называют основной, или *нулевой гипотезой* (англ. *null hypothesis*). И, собственно, все вычисления строятся на том, что нулевая гипотеза верна. Поэтому очень важно выбрать ее правильно.

Далее, нужно вычислить вероятность для следующего события. Если мы повторим еще раз наш эксперимент (т.е. возьмем другие 10 проб воды из этого источника, и измерим содержание хлорид-ионов в этих пробах), каков шанс, что среднее значение концентрации для этих новых проб будет таким же, как мы получили до этого, или более экстремальным по отношению к нулевой гипотезе? Таким образом, мы пытаемся оценить, насколько "нормальна" наша выборка по отношению к нулевой гипотезе, насколько она "возможна" в таких условиях.

Если вспомнить вычисления, сделанные для доверительного интервала, то для нашей выборки $m(x) = 50.5$, но мы полагаем, что она взята из популяции, где $\mu(x) \leq 46.0$. Понятно, что наша выборка не очень хорошо подходит для нашей гипотезы, но мы живем в мире случайностей, где почти любое событие возможно. Давайте просто посчитаем эту вероятность.

Для начала положим, что $\mu(x) = 46.0$ и вычислим, насколько сильно значение нашей выборки отличается от нашей гипотезы:

$$t = \frac{m - \mu}{\frac{s}{\sqrt{n}}} = \frac{50.5 - 46.0}{\frac{6.1}{\sqrt{10}}} = 2.31$$

Т.е., другими словами, среднее значение нашей выборки на 2.31 стандартных ошибки больше, чем среднее значение популяции, согласно нашей гипотезе о ней. Но какова вероятность, если мы еще раз повторим этот эксперимент, получить выборку со средним значением на таком же, или даже большем расстоянии (т.е. $t \geq 2.31$). Эту вероятность можно проиллюстрировать с помощью двух графиков, показанных на рисунке 1.23.

По сути, они показывают одно и тоже, но левый график оперирует средними значениями выборки, а правый — построен для стандартного распределения Стьюдента, т.е. оперирует t-значениями. На обоих графиках видно, что шанс получить выборку с $m = 50.5$ (или $t = 2.31$), или больше из генеральной совокупности с $\mu \leq 46.0$ не очень высок. Он соответствует закрашенной области на обоих графиках и может быть посчитать с помощью функции вероятности:

$$P(t \geq 2.31) = 1 - F(2.31, 9) = 0.0231$$

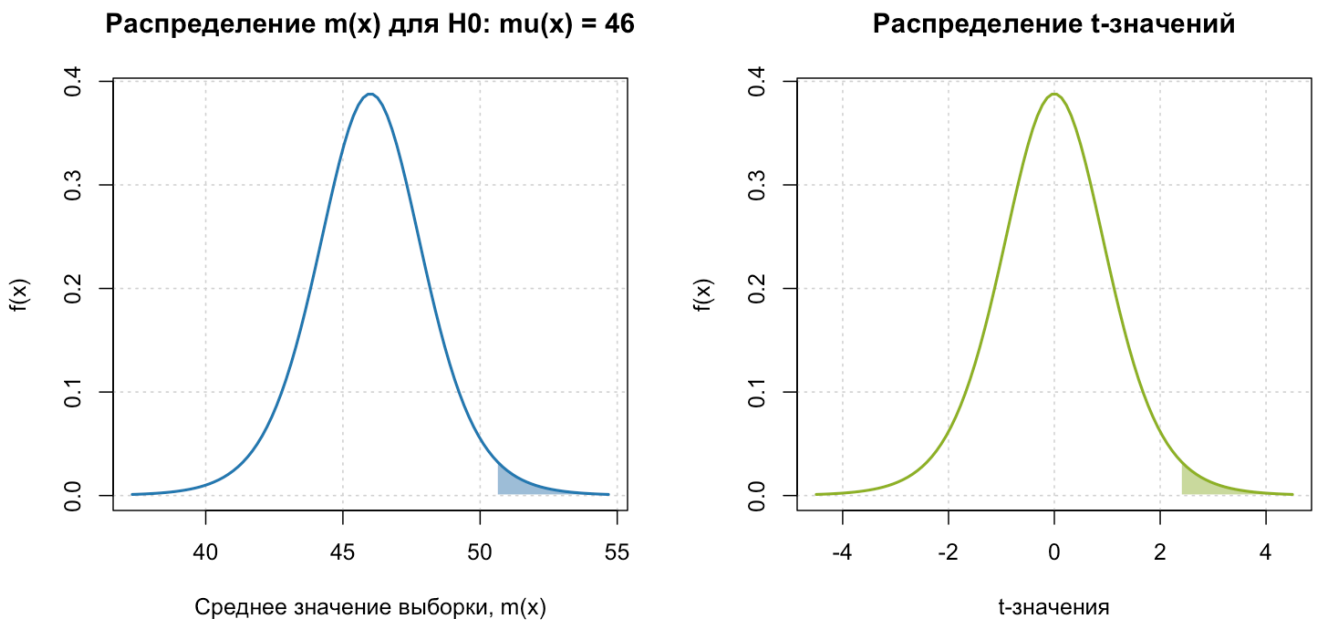


Рис. 1.23. Иллюстрация вычисления p-значения для H0: слева — для распределения средних значений, $m(x)$, справа — для t-значений. На обоих графиках p-значение для H0: $\mu \leq 46$ соответствует закрашенной области.

Здесь мы вычитаем результат из единицы, так как искомая вероятность — это область по правую сторону от нашего значения, а функция вероятности всегда возвращает область по левую сторону (помним, что общая площадь под кривой плотности вероятностей равна единице).

Т.е. если мы повторим наш эксперимент, скажем, тысячу раз, только в 23 случаях среднее значение выборки будет равным, или меньшим чем 50.5 мг/л, если (важно!) наша гипотеза о среднем значении популяции верна.

Эта вероятность называется *p-значением* (англ. *p-value*), она является основным результатом любого статистического теста. По сути — это мера экстремальности нашей выборки по отношению к нулевой гипотезе, чем меньше p-значение, тем более экстремальной (а значит и менее возможной) она является. Другая интерпретация p-значений — это мера воспроизводимости результата эксперимента для данной нулевой гипотезы.

Часто при анализе необходимо принять решение полагать ли нулевую гипотезу верной, или отбросить ее, как ложную. Например, в нашем случае понять, нужно ли применять какие-то меры по очистке источника, или считать источник незагрязненным. В этом случае используется пограничное значение для p — *уровень значимости* (англ. *significance level*), обычно обозначаемое греческой буквой α. Если полученное p-значение меньше заданного уровня значимости, то нулевую гипотезу отвергают, как неверную, если нет — то принимают.

Однако, мы не знаем наверняка, верна ли нулевая гипотеза, или нет. Даже если p-значение очень мало, все равно может так случится, что выборка получена из популяции, где нулевая гипотеза верна. В конце концов, люди болеют редкими заболеваниями, выигрывают в лотерею и случаются другие маловероятные события. Поэтому, во-первых, не нужно воспринимать ситуацию, когда p-значение меньше уровня значимости, как некую индульгенцию и полагать, что нулевая гипотеза точно невозможна. Во-вторых, важно понимать последствия вашего решения и правильно их интерпретировать, о чем пойдет речь в следующем разделе.

1.5.2 Ошибки тестирования и мощность тестов

Таблица ниже показывает все возможные гипотезы о среднем, которые мы можем протестировать, для одной выборки:

Нулевая гипотеза, H0	Альтернативная гипотеза, H1
$\mu \leq \mu_0$	$\mu > \mu_0$
$\mu \geq \mu_0$	$\mu < \mu_0$
$\mu = \mu_0$	$\mu \neq \mu_0$

Как можно видеть, для каждой нулевой гипотезы имеет место альтернативная гипотеза, которую мы принимаем, если нулевая гипотеза отбрасывается, как неверная. Таким образом, для любой гипотезы возможны четыре ситуации, показанные схематично на рисунке 1.24.

Рис. 1.24. Ошибки первого и второго рода.

В двух из этих четырех ситуаций мы принимаем ошибочное решение, а в двух — верное. Эти ошибки называются, соответственно, ошибками первого (Type I) и второго (Type II) рода. Рассмотрим коротко, какие факторы на них влияют, и как можно снизить вероятность этих ошибок.

Начнем с ошибки первого рода, тут все достаточно просто, ее вероятность равна выбранному уровню значимости. Т.е., если вы работаете с $\alpha = 0.01$, то в 1 случае из 100 ваше решение отвергнуть нулевую гипотезу как неверную будет ошибочным. Мы уже говорили немного о том, что нулевая гипотеза выбирается, как нечто наиболее вероятное. Например, в исследованиях почти всегда нулевая гипотеза означает отсутствие какого-то эффекта (метод ничего не меняет, лекарство не работает, и т.д.). В этом случае ошибка первого рода — это шанс ошибочно констатировать наличие эффекта. Часто такую ситуацию называют *ложноположительным решением* (англ. *false positive*).

Вероятность ошибки второго рода зависит от нескольких факторов, главными из которых являются размер выборки, величина эффекта и, также, уровень значимости. Последний в этом случае работает в обратную сторону — чем выше уровень значимости, тем меньше вероятность допустить ошибку второго рода.

Рассмотрим конкретный пример. Пусть снова речь идет о загрязненности водоема хлорид-ионами и предельно допустимое значение равно 146 мг/л. Если мы живем в экологически благополучном районе, то естественно будет ожидать, что водоем не загрязнен, т.е. нулевая гипотеза: $H0 : \mu \leq 146$.

Однако, предположим, что это не так, и что истинная средняя концентрация хлорид ионов в водоеме равна $\mu(x) = 150$ мг/л, и стандартное отклонение равно $\sigma(x) = 2$ мг/л. Мы это, конечно, не знаем и будем тестировать нашу нулевую гипотезу $H0 : \mu \leq 146$. Какова вероятность, что мы не сможем ее отвергнуть, т.е. сделаем ошибку в данном случае?

Давайте посмотрим на несколько графиков. Графики на рисунке 1.25 показывают следующее. Синяя линия — это распределение средних значений выборок для нулевой гипотезы (т.е. вокруг $\mu = 146$). Зеленая линяя — это распределение средних значений выборок для альтернативной гипотезы (т.е. вокруг $\mu = 150$). И в том, и в другом случае мы предполагаем, что размер нашей выборки $n = 10$.

Другими словами, каждый раз, когда мы берем выборку (скажем 10 проб воды из этого водоема), то среднее значение для этой выборке будет в диапазоне, который описывает зеленая кривая. Но мы будем думать, что это распределение описывается синей кривой. Когда мы сможем заметить, что наше предположение ошибочно? Только если p-значение будет меньше уровня значимости и мы отвергнем H0.

График слева показывает ситуацию для $\alpha = 0.05$, а справа — для $\alpha = 0.10$. Красная область показывает вероятность того, что среднее значение нашей выборки будет достаточно близко к H0, что мы не сможем ее отвергнуть. Синяя область показывает обратную ситуацию — вероятность того, что среднее значение выборки будет достаточно далеко от H0, чтобы ее отвергнуть.

Другими словами, площадь красной области — это и есть вероятность ошибки второго рода, обычно обозначаемая греческой буквой β. А площадь зеленой области — это вероятность того, что мы не сделаем

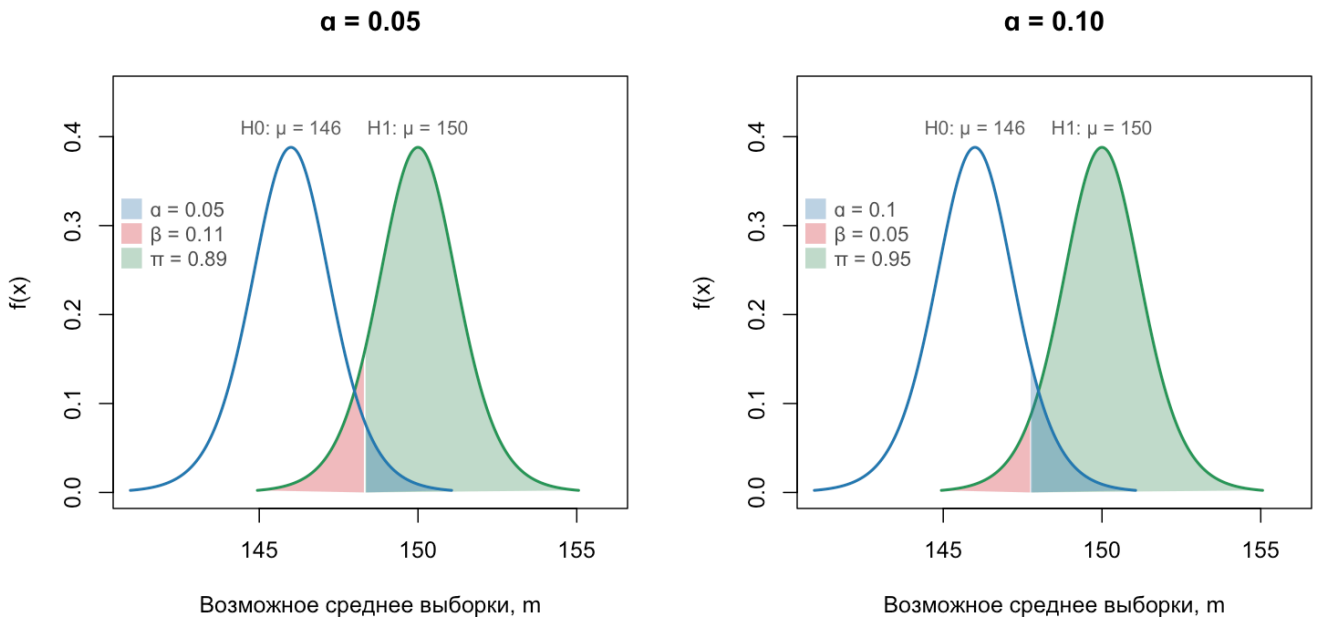


Рис. 1.25. Оба графика показывают, как величина ошибки второго рода (β) зависит от уровня значимости (α). График слева построен для $\alpha = 0.05$, и, если на самом деле H1 верна, то в 11 из 100 случаев мы не сможем это констатировать (так как не сможем отвергнуть H0 на этом уровне значимости). Увеличение уровня значимости до $\alpha = 0.10$ уменьшает этот шанс до 0.05, как показывает график справа. С другой стороны, увеличение α ведет к увеличению вероятности сделать ошибку первого рода, так что это не самый лучший способ.

эту ошибку, обозначаемая как π. Очевидно, что $\pi = 1 - \beta$. Эта вероятность называется *мощностью теста* (англ. *power of test*), она показывает, насколько хорошо тест позволяет распознать эффект, если он на самом деле имеет место быть.

Вернемся к двум графикам. Как можно видеть, площади красной и зеленой областей напрямую зависят от уровня значимости — чем он меньше, тем выше шанс допустить ошибку второго рода и наоборот. Это первый фактор, но не самый главный.

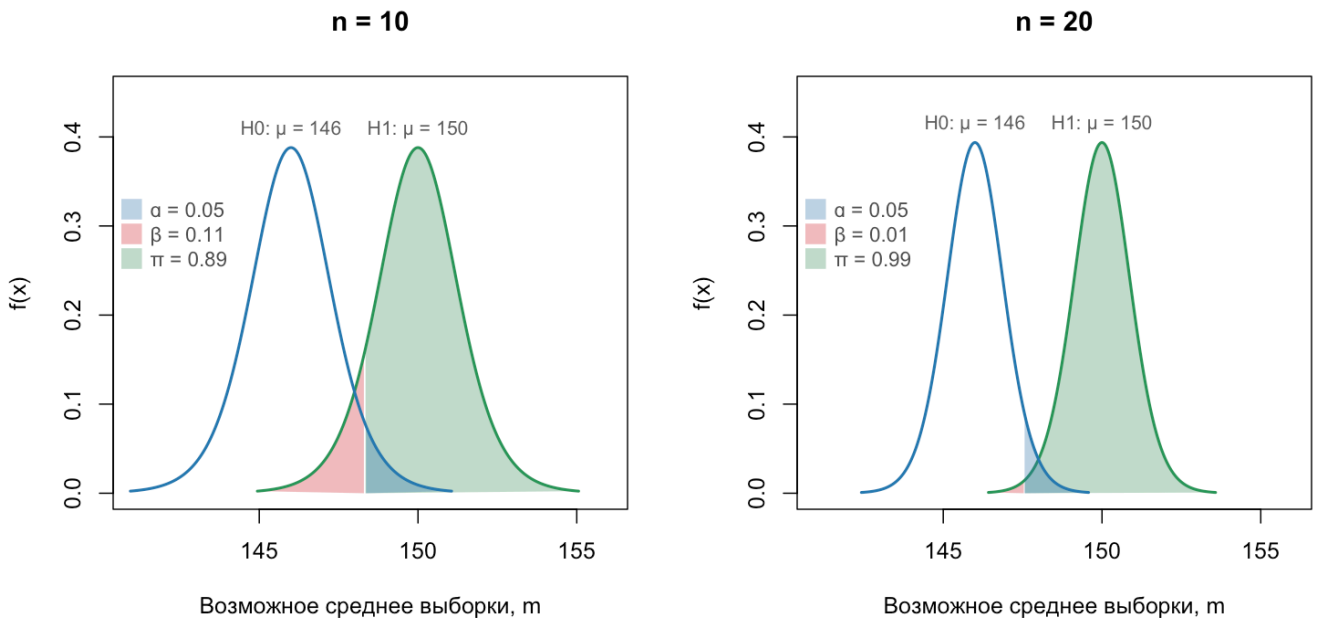


Рис. 1.26. Оба графика показывают, как величина ошибки второго рода (β) зависит от размера выборки (n). График слева построен для $n = 10$, и, если на самом деле H1 верна, то в 11 из 100 случаев мы не сможем это констатировать (так как не сможем отвергнуть H0 на этом уровне значимости). Увеличение размера выборки до $n = 20$ уменьшает этот шанс до 0.01, как показывает график справа.

Теперь посмотрим на два графика, показанных на рисунке 1.26. И в том, и в другом случае мы используем одинаковый уровень значимости, но разный размер выборки: $n = 10$ для левого графика и $n = 20$ для правого. Очевидно, что размер выборки напрямую влияет на величину неопределенности для средних значений, и чем больше размер, тем эта неопределенность меньше. Меньшая неопределенность приводит к тому, что разница между H0 и H1 становится более явной, что влияет на вероятность ошибки второго рода, как можно видеть из графиков. Так, шанс не заметить загрязнение для выборки с 10 пробами равен 11% ($\beta = 0.11$), но если увеличить размер выборки до 20 проб, то этот шанс уменьшается до 1% ($\beta = 0.01$).

Наконец, третья ситуация, показанная на рисунке 1.27. Здесь левый график построен для уровня значимости $\alpha = 0.05$ и размера выборки $n = 10$. Второй график построен для тех же самых значений, но уровень хлорид-ионов в источнике немного выше ($\mu = 151$), т.е. разница между H0 и H1 стала больше. Это, конечно же, тоже влияет на вероятность ошибки, так как сдвигает распределение возможных средних

значений вправо, дальше от H0. И не “заметить” такой уровень загрязненности становится труднее — шанс допустить ошибку второго рода равен 3% против 11% для $\mu = 150$.

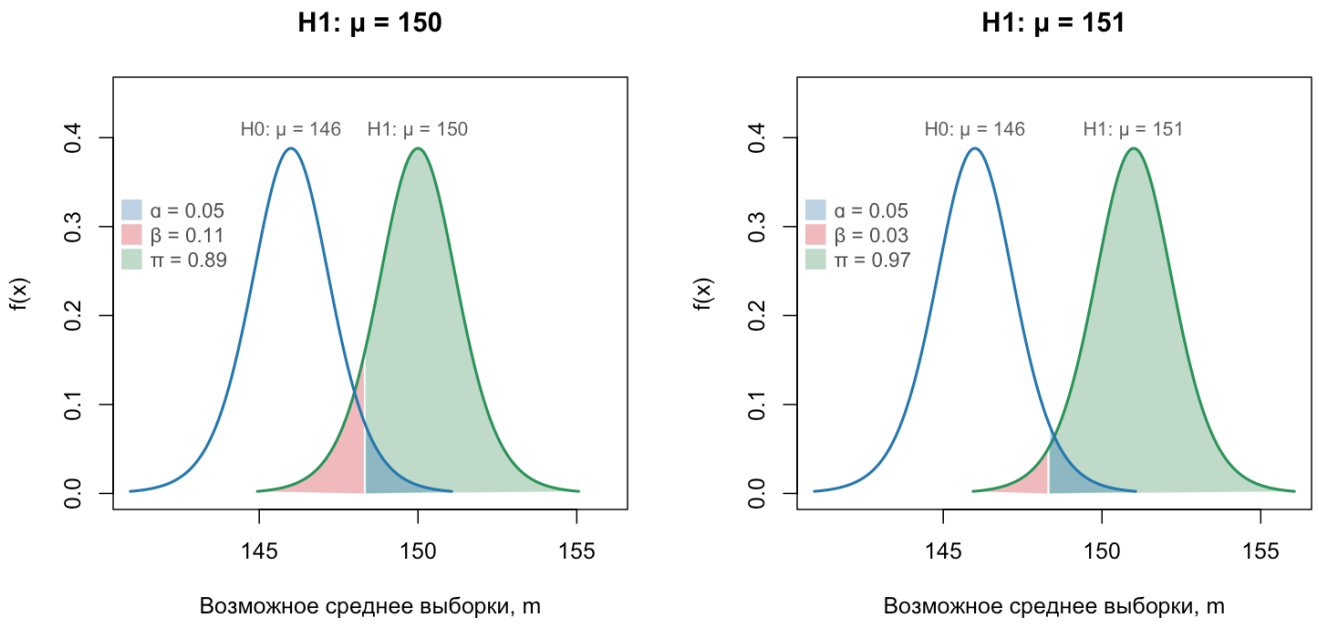


Рис. 1.27. Графики показывает как величина ошибки второго рода (β) зависит от размера эффекта (разницы между H0 и H1). График слева построен для $H1 : \mu = 150$, а справа — для $H1 : \mu = 151$. Повышение уровня загрязненности водоема на 1 мг/л в этом случае уменьшает шанс допустить ошибку второго рода с 0.11 до 0.03.

Другими словами, вероятность ошибки второго рода и способность теста распознать эффект зависит от размера этого эффекта и от уровня неопределенности (шума). Увеличивая первый и уменьшая второй, мы улучшаем отношение сигнал/шум и делаем статистический тест более чувствительным.

1.5.3 Сравнение средних значений двух выборок

Поговорим теперь о том, как сравнивать средние значения двух генеральных совокупностей, используя статистики соответствующих выборок. Формально задачу можно описать следующим образом.

Имеется некоторая генеральная совокупность с параметрами μ_1, σ_1. Извлечем из нее случайную выборку размера n_1, которая будет иметь статистики m_1, s_1. Далее возьмем другую генеральную совокупность с параметрами μ_2, σ_2 и соответствующую выборку размера n_2 со статистиками m_2, s_2. Как сравнить μ_1 и μ_2, используя только статистики, посчитанные для выборок?

Например, у нас есть два источника воды, которые мы хотим сравнить между собой (в плане концентрации химических веществ), или сравнить эффективность двух разных лекарственных средств, двух разных методов очистки воды от пестицидов и т.п.

Ниже мы для простоты рассмотрим простой случай, когда $\sigma_1 = \sigma_2$. Этот случай можно обобщить и на исходный вариант, выкладки будут похожими. Из предыдущей части мы помним, что средние значения выборки и популяции можно связать между собой с помощью распределения Стьюдента:

$$m_1 = \mu_1 + t\frac{s_1}{\sqrt{n}}$$

$$m_2 = \mu_2 + t\frac{s_2}{\sqrt{n}}$$

Можно показать, что разность средних значений выборок, $m_1 - m_2$, также описывается распределением Стьюдента с числом степеней свободы DoF = $n_2 - n_1 - 2$. Она будет распределена симметрично относительно разницы средних значений популяций, $\mu_1 - \mu_2$, следующим образом:

$$m_2 - m_1 = (\mu_2 - \mu_1) + t\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$$

Таким образом, мы можем делать ровно тоже, что и для случая одной выборки, а имеено:

1. Вычислить доверительный интервал для $\mu_2 - \mu_1$
2. Протестировать гипотезу о значении $\mu_2 - \mu_1$

Так как при сравнении двух популяций нам важно знать какая из них имеет большее среднее значение (если они не одинаковы), то нулевую гипотезу в этом случае можно свести к следующим трем случаям:

Ситуация	Нулевая гипотеза, H0	Альтернативная гипотеза, H1
$\mu_1 = \mu_2$.	$\mu_1 - \mu_2 = 0$	$\mu_1 - \mu_2 \neq 0$
$\mu_1 \geq \mu_2$	$\mu_1 - \mu_2 \geq 0$	$\mu_1 - \mu_2 < 0$
$\mu_1 \leq \mu_2$	$\mu_1 - \mu_2 \leq 0$	$\mu_1 - \mu_2 > 0$

Такой тест называется тестом Стьюдента для двух выборок (англ. *two sample t-test*). Пример ниже показывает как он работает.

Предположим нам нужно сравнить два сорта яблок, какой из них содержит больше сахара (имеется в виду общее количество простых сахаров — фруктозы, глюкозы и т.д.) в плодах, если условия роста, орошения, удобрения будут одинаковыми. Для этого у нас есть 10 яблок каждого сорта, отобранные из популяции случайным и правильным образом. Характеристики яблок собраны в следующей таблице (здесь мы используем процент от массы плодов в качестве единиц измерения):

Сорт А	Сорт Б
$n_1 = 10$	$n_2 = 10$
$m_1 = 9.6$	$m_2 = 10.2$
$s_1 = 1.2$	$s_2 = 1.6$

Как мы можем видеть, если сравнивать выборки, то сорт Б содержит в среднем больше сахара чем сорт А.

Будем тестировать гипотезу о том, что на самом деле между популяциями нет никакой разницы ($\mu_1 = \mu_2$) и будем считать, что еще до сбора яблок мы выбрали доверительную вероятность 99% ($\alpha = 0.01$). Для начала посчитаем совокупную стандартную ошибку:

$$s(m) = \sqrt{\frac{1.2^2}{10} + \frac{1.6^2}{10}} = \sqrt{\frac{4}{10}} \approx 0.63$$

Далее, так как мы работаем с 99% доверительной вероятностью, то 1% будет за пределами доверительного интервала, по 0.5% на каждом конце. Таким образом, искомое для доверительного интервала t-значение можно вычислит с помощью обратной функции вероятности для $p = 0.995$ и DoF = $(10 - 1) + (10 - 1) = 18$:

$$t = F^{-1}(0.995, 18) \approx 2.88$$

Следовательно, доверительный интервал будет следующим:

$$\mu_1 - \mu_2 = (9.6 - 10.2) \pm 2.88 \times 0.63 \approx -0.6 \pm 1.81 = [-2.41, 1.21]$$

Как можно видеть, разница среднего содержания сахара в двух сортах может быть как отрицательной (т.е. $\mu_1 < \mu_2$), так и положительной (т.е. $\mu_1 > \mu_2$). Т.е. доверительный интервал на этом уровне значимости и для данных выборок не позволяет увидеть какую-либо разницу между двумя сортами. Есть ли она на самом деле, мы не знаем, исходя из нашего эксперимента у нас нет достаточных оснований (англ. *evidence*) для того, чтобы утверждать, что разница имеется.

Попробуем теперь посчитать p-значение для теста о разнице средних значений. Начнем с t-значения:

$$t = \frac{9.6 - 10.2}{\sqrt{\frac{1.2}{10} + \frac{1.6}{10}}} = \frac{-0.6}{0.63} \approx -0.95$$

Уже по t-значению можно видеть, что наши выборки не выглядят экстремальными по отношению к нулевой гипотезе — разница между средними значениями выборок меньше одной стандартной ошибки. Так как в данном случае нулевая гипотеза симметрична, т.е. и положительные, и отрицательные значения

t будут одинаково экстремальны для этой гипотезы, чтобы вычислить p-value нужно вычислить шанс получить выборку с $t \leq -0.95$, затем шанс получить выборку с $t \geq 0.95$ и потом эти две вероятности сложить:

$$p = F(-0.95, 18) + (1 - F(0.95, 18)) \approx 0.177 + 0.177 = 0.354$$

Так как распределение Стьюдента симметрично, то можно вычислить вероятность только для одной стороны и затем просто умножить ее на два. Как можно видеть, p-значение весьма велико для того, чтобы отвергнуть нулевую гипотезу о том, что средние значения популяций равны.

1.5.4 Индуктивная статистика в R

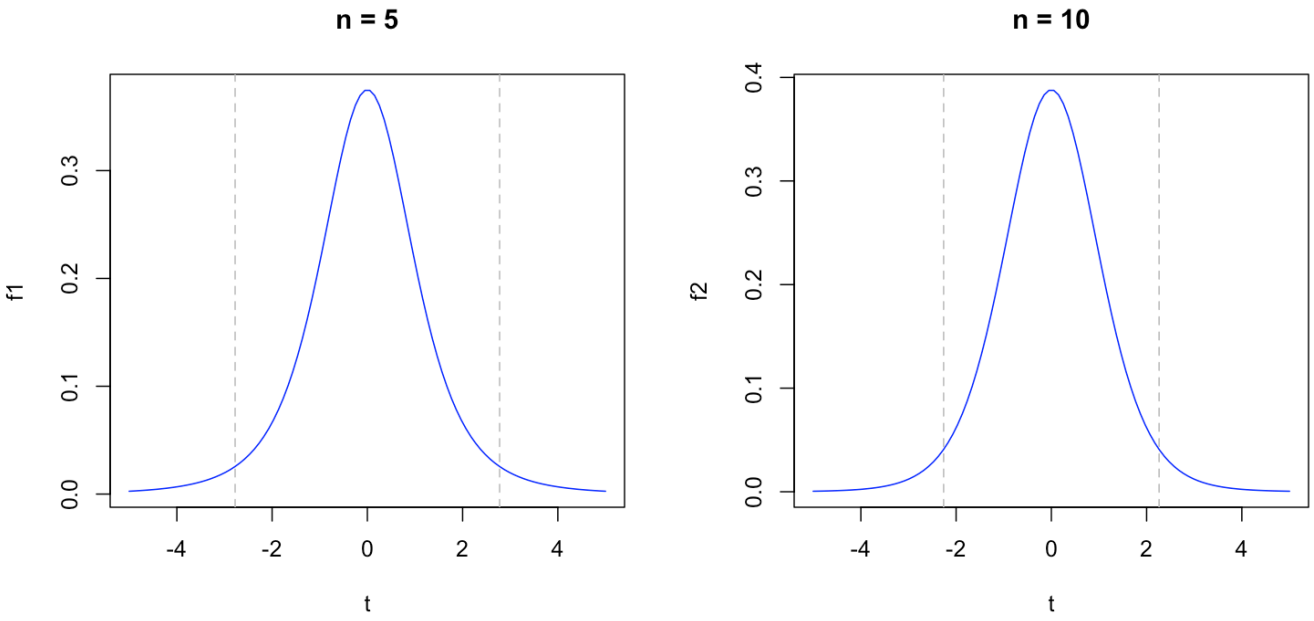
Распределение Стьюдента и t-тест для одной выборки

Как и любое другое теоретическое распределение, распределение Стьюдента представлено в R четырьмя функциями (для вычисления плотности вероятностей, вероятности, квантилей и случайных чисел). В отличие от нормального распределения, эти функции реализованы только для стандартного распределения, т.е. описывают распределение t-значений и имеют только один параметр — число степеней свободы.

Блок кода внизу показывает, как построить графики распределения плотности вероятностей для двух выборок разного размера и добавить границы 95% интервала вокруг центра в виде вертикальных линий. Из полученного графика можно видеть, что для выборки меньшего размера этот интервал шире.

```r
# задаем размеры выборок
n1 <- 5
n2 <- 10

# генерируем t-значения в нужном диапазоне
t <- seq(-5, 5, length.out = 100)

# вычисляем плотности
f1 <- dt(t, n1 - 1)
f2 <- dt(t, n2 - 1)

# вычисляем левые границы для 95% интервала
t1l <- qt(0.025, n1 - 1)
t2l <- qt(0.025, n2 - 1)
```

```
# вычисляем правые границы для 95% интервала
t1r <- qt(0.975, n1 - 1)
t2r <- qt(0.975, n2 - 1)

# строим графики
par(mfrow = c(1, 2))
plot(t, f1, type = "l", col = "blue", main = paste0("n = ", n1))
abline(v = c(t1l, t1r), col = "gray", lty = 2)
plot(t, f2, type = "l", col = "blue", main = paste0("n = ", n2))
abline(v = c(t2l, t2r), col = "gray", lty = 2)
```



Теперь покажем, как вычислить 95% доверительный интервал для выборки с концентрациями хлорид-ионов в 10 пробах воды. Сначала сделаем это вручную, с помощью пошаговых вычислений, как это описано выше в данном разделе:

```
# вектор с исходными значениями
x <- c(41.0, 44.0, 46.4, 47.6, 49.4, 50.0, 53.4, 54.4, 58.5, 59.9)

# считаем статистики
m <- mean(x)
```

```
s <- sd(x)
n <- length(x)

# вычисляем критическое значений t для 95% (правую часть)
t <- qt(0.975, n - 1)

# вычисляем стандартную ошибку и величину неопределенности
se <- s / sqrt(n)
em <- t * se

# вычисляем интервал и показываем его
ci <- c(m - em, m + em)
names(ci) <- c("2.5%", "97.5%")
show(round(ci, 1))
```

```
 2.5% 97.5%
 46.1  54.8
```

В R есть специальная функция для теста Стьюдента, которая также вычисляет доверительный интервал. В примере ниже показано, как это сделать (мы подразумеваем, что у вас уже есть вектор с исходными значениями x).

```
res <- t.test(x)
show(res)
```

```

    One Sample t-test

data:  x
t = 26.149, df = 9, p-value = 8.444e-10
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 46.09467 54.82533
sample estimates:
mean of x
    50.46
```

Как можно видеть, результаты выполнения функции `t.test()` содержат много информации. По сути мы делаем тест на то, что среднее значение популяции равно нулю. Но при этом функция также вычисляет доверительный интервал, который, как можно заметить, совпадает с вычисленным нами ранее. Величину доверительной вероятности можно поменять, используя параметр `conf.level`, как показано в примере ниже.

```
# вычисление доверительного интервала для вероятности 99%
res <- t.test(x, conf.level = 0.99)
show(res)
```

```
    One Sample t-test

data:  x
t = 26.149, df = 9, p-value = 8.444e-10
alternative hypothesis: true mean is not equal to 0
99 percent confidence interval:
 44.18872 56.73128
sample estimates:
mean of x
    50.46
```

Для того, чтобы протестировать гипотезу о среднем значении генеральной совокупности нужно задать предполагаемое значение в виде аргумента `mu` и, при необходимости, указать какая именно гипотеза тестируется (больше или равно, меньше или равно, равно). Это делается с помощью параметра `alternative`, т.е. мы указываем тип альтернативной гипотезы. Этот параметр может принимать следующие значения (в скобках указана соответствующая нулевая гипотеза): `"greater"` (H0: $\mu \leq \mu_0$), `"less"` (H0: $\mu \geq \mu_0$), `"two.sided"` (H0: $\mu = \mu_0$).

Блок кода ниже показывает использование функции для тестирования гипотезы $\mu \leq 46$ из примера, рассмотренного нами выше.

```
res <- t.test(x, mu = 46, alternative = "greater")
show(res)
```

```
    One Sample t-test
```

```
data:  x
t = 2.3112, df = 9, p-value = 0.02307
alternative hypothesis: true mean is greater than 46
95 percent confidence interval:
 46.9226     Inf
sample estimates:
mean of x
    50.46
```

Опять же, как можно заметить, полученное таким образом p-значение (0.0231) совпадает с вычисленным нами “вручную” ранее.

Сравнение средних значений для двух групп

Для сравнения средних значений двух генеральных совокупностей, основываясь на двух случайных выборках, используется та же самая функция, `t.test`. Единственная разница по сравнению со случаем одной выборки — нужно указать два вектора значений. Если также известно, что дисперсия значений в популяциях равна, то можно также указать это с помощью параметра `var.equal`.

Блок кода ниже воспроизводит наш пример, где сравниваются два сорта яблок. В этом случае вместо статистик мы используем векторы с измеренными значениями сахара в яблоках, при этом статистики довольно близки к тем, которые мы использовали в примере. Соответственно, результаты теста (число степеней свободы, t-значение, p-значение, доверительный интервал) также довольно близки к вычисленным нами выше значениям.

```r
# содержание сахара в яблоках из двух выборок
x1 <- c(9.6, 9.8, 9.4, 9.2, 9.6, 9.8, 8.1, 10.9, 11.9, 7.8)
x2 <- c(10.0, 10.5, 10.2, 9.4, 11.2, 10.1, 10.2, 10.8, 13.0, 6.6)

res <- t.test(x1, x2, var.equal = TRUE, conf.level = 0.99)
show(res)
```

```
    Two Sample t-test

data:  x1 and x2
t = -0.93693, df = 18, p-value = 0.3612
alternative hypothesis: true difference in means is not equal to 0
```

```
99 percent confidence interval:
 -2.402605  1.222605
sample estimates:
mean of x mean of y
     9.61     10.20
```

1.6 Дисперсионный анализ

1.6.1 Проблема множественного сравнения

Что делать, если нужно сравнить не две группы значений, а три, или более? Например, ваша цель — увеличить выход продукта химической реакции с помощью подбора нужного катализатора, и нужно проверить как минимум три разных. Или, пойти еще дальше и исследовать эффект катализаторов в разных температурных условиях. Как быть в этом случае?

Рассмотрим простой пример. Пусть нам нужно сравнить три разных метода по очистке воды от пестицидов (для простоты обозначим их буквами A, B, C). Для этого мы подготовили 30 проб воды из одного и того же источника. Мы измерили начальную концентрацию пестицидов в каждом образце, затем разбили их случайным образом на три группы по 10 проб в каждой и использовали один из трех методов для очистки каждой пробы. Результатом этого эксперимента стала относительная разница между исходной концентрацией пестицидов и конечной, измеренной после очистки.

Пусть метод A в среднем снизил концентрацию на 10% ($m_A = 10$), метод B на 12% ($m_B = 12$) а метод C на 8% ($m_C = 8$). Значит ли, что метод B наиболее эффективен для очистки воды от пестицидов? Чтобы понять это, нам нужно протестировать гипотезу о равенстве средней эффективности методов, которая в этом случае будет следующей:

$$H0 : \mu_A = \mu_B = \mu_C$$

Т.е., изначально мы полагаем, что методы в среднем имеют одинаковую эффективность. Теперь нам нужно оценить, насколько экстремальны результаты, полученные нами в этом эксперименте, по отношению к нулевой гипотезе. Т.е., если мы повторим эксперимент, какова вероятность, что разница между средней эффективностью методов будет такая же, как полученная нами в текущем эксперименте, или еще больше?

Загвоздка в том, что нам пока известен лишь один способ сравнения средних значений — тест Стьюдента, который может делать это только для двух выборок. Т.е. для получения результата нам нужно сравнить наши методы попарно с помощью трех тестов:

$$H0 : \mu_A = \mu_B$$

$$H0 : \mu_A = \mu_C$$

$$H0 : \mu_B = \mu_C$$

Однако, в этом случае мы столкнемся со следующей проблемой. Пусть мы работаем на уровне значимости 5% ($\alpha = 0.05$) и пусть на самом деле все три метода имеют одинаковую эффективность, т.е. исходная гипотеза верна. Тогда в каждом из трех тестов у нас есть 5% шанс принять неправильное решение, и 95% — правильное. Мы сможем подтвердить нашу гипотезу, только если в каждом из трех тестов мы примем правильное решение (т.е. поддержим нулевую гипотезу). Вероятность такого события равна:

$$0.95 \times 0.95 \times 0.95 = 0.857$$

Другими словами, если мы будет использовать три теста Стьюдента для проверки одной гипотезы, шанс, что мы примем неправильное решение (сделаем ошибку первого рода) равен $1 - 0.857 = 0.143$ или 14.3%. Что почти в три раза выше исходного значения уровня значимости.

Эта проблема называется проблемой *множественного тестирования*. Чем больше статистических тестов необходимо провести, чтобы проверить одну глобальную гипотезу, тем больше шанс, что вы “увидите” эффект там, где его нет. Это касается любых тестов, не только теста Стьюдента. Это как если вы будете бросать одну и ту же кость несколько раз подряд, шанс, что в итоге выпадет 6, увеличивается с каждым разом.

Решений для этой проблемы несколько:

1. Корректировать уровень значимости в каждом отдельном тесте так, чтобы в итоге получилось правильное значение. Например, если в каждом из трех тестов в примере выше, мы будем принимать решение используя уровень значимости $0.05/3 = 0.017$, то итоговый шанс сделать ошибку первого рода будет равен $1 - 0.983^3 = 0.0501$ т.е. те самые 5% на которые мы изначально рассчитывали. Этот подход называется *поправкой Бонферрони* (англ. *Bonferroni correction*).
2. Использовать один тест для тестирования сложной гипотезы. Для сравнения средних значений таким тестом является дисперсионный анализ, о котором и пойдет речь далее в этом разделе.
3. Использовать специальное распределение, которое позволяет учесть количество одновременных тестов. Например, для сравнения средних значений, можно использовать студентизированное распределение диапазона. На этом распределении основан тест достоверно значимой разности Тьюки, который мы также рассмотрим.

1.6.2 Однофакторный дисперсионный анализ

Дисперсионный анализ (англ. *analysis of variance* или *ANOVA*) позволяет сравнить средние значения для нескольких групп. При этом разделение на группы может быть обусловлено как наличием одного фактора (например, тип катализатора в реакции), так и нескольких. Соответственно, различают однофакторный (англ. *one-way*) и многофакторный (англ. *n-way*) дисперсионный анализ. Кроме работы с множеством факторов, дисперсионный анализ также позволяет оценить наличие взаимодействия между факторами — когда величина одного фактора влияет на эффект, полученный от другого. В этом разделе мы рассмотрим каждый из этих случаев.

> *Можно сказать, что дисперсионный анализ позволяет построить математическую модель влияния одного, или нескольких факторов на отклик — измеряемую величину. С этой точки зрения, дисперсионный анализ очень похож на линейную регрессию, которая будет рассмотрена далее в этой книге.*

Однофакторный дисперсионный анализ подразумевает, что у нас есть несколько групп значений, и эти группы заданы величинами (уровнями) одного фактора. Т.е. по сути фактор должен быть либо качественным, либо количественным, но с ограниченным (дискретным) набором значений. Продолжим пример с методами очистки воды от пестицидов и рассмотрим следующие данные:

A	B	C
9	11	7
8	14	10
10	12	8
12	10	6
11	13	9

Числа в таблице означают процент пестицидов, который удалось убрать из данного образца. Для простоты мы будем работать с положительными числами. Т.е. каждый метод был опробован на пяти разных образцах воды и показал, как мы видим, разную эффективность. Однако, в среднем метод A позволил уменьшить концентрацию пестицидов на 10%, метод B на 12%, и метод C на 8%. Нулевая гипотеза в этом случае предполагает, что если мы проведем достаточно большое число экспериментов, то разницы между средней эффективностью методов не увидим:

$$\mu_A = \mu_B = \mu_C$$

Если мы вспомним, каким образом сравниваются средние значения в тесте Стьюдента, то заметим, что основная идея состоит в вычислении своего рода отношения сигнала и шума. Сигнал в этом случае — это наблюдаемая разница ($m_1 - m_2$), а шум — это неопределенность, или случайный разброс результатов внутри каждой группы, который в тесте Стьюдента оценивается с помощью стандартной ошибки.

Проблема в том, что разницу можно вычислить только для двух значений, что делать, если средних значений три, или больше? Нужно вычислить *дисперсию*! По сути, дисперсия — это аналог квадрата разницы для большого числа значений, эта статистика (как и разница) показывает, как далеко значения находятся друг от друга. Таким образом, идея дисперсионного анализа заключается в следующем:

1. Вычислить дисперсию для "сигнала" — наблюдаемого эффекта
2. Вычислить дисперсию для "шума" — случайной вариации данных
3. Вычислить отношение двух дисперсий, F-значение
4. Используя соответствующее распределение, оценить насколько высоко отношение сигнала к шуму, и какова вероятность получить такое отношение случайно для данной нулевой гипотезы

> *Можно показать (проверьте позже это на практике с помощью R), что результаты дисперсионного анализа для двух групп идентичны тесту Стьюдента. Таким образом, ANOVA — это обобщение теста Стьюдента на случай большего числа групп.*

Основные этапы однофакторного дисперсионного анализа представлены на рисунке 1.28. На самом деле, есть разные подходы к его реализации. Тот, что показан на рисунке, нужен лишь для простоты иллюстрации и объяснения механизма теста.

Рис. 1.28. Основные этапы однофакторного дисперсионного анализа.

Первый шаг — это избавление от смещения — мы вычисляем общую среднюю эффективность всех методов ($m = 10$) и вычитаем ее из исходных значений. Это нужно для того, чтобы легче было делать вычисления на следующих этапах. При этом мы не меняем самого главного — взаимного отношения значений.

Так, самое первое значение в таблице вместо 9 станет равным −1 т.е. эффективность, показанная в данном эксперименте, была на 1% меньше общей средней эффективности. Последний эксперимент для метода B имеет новое значение 3 (вместо исходного 13), что означает на 3% выше общей средней эффективности. Разница между этими двумя экспериментами по-прежнему составляет 4%, как и в случае исходных значений.

Нужно также заметить, что исходные данные содержат 15 независимых друг от друга измерений, т.е. число степеней свободы равно 15. Однако, после того как мы вычислили общее среднее значение и вычли его из данных, число степеней свободы стало равным 14. Так как любое значение теперь можно вычислить, зная среднее и остальные 14 значений. Мы можем вычислить дисперсию исходных данных, просуммировав квадраты значений (70) и разделив эту сумму на число степеней свободы (14), получив 5, как показано на рисунке.

Следующий шаг — разделить исходные данные на сигнал и шум. Представим на минуту, что мы сделали идеальный эксперимент, в котором случайной вариации нет вообще. В этом случае будет логичным ожидать, что каждый метод покажет одинаковую эффективность для всех пяти проб. В качестве оценки такой эффективности мы возьмем среднее значение, полученное для каждого метода в нашем эксперименте. Этот новый блок значений, который мы назовем матрицей эффектов, показан зеленым цветом на рисунке.

Число степеней свободы для этого случая равно двум, так как в данном случае мы сравниваем группы, а не исходные значения. Просуммировав квадраты значений в этой матрице, и разделив на число степеней свободы, мы получим дисперсию сигнала, равную 20.

Далее мы можем получить таблицу со случайной вариацией данных, которая не связана с тем, что мы используем разные методы. Это можно сделать путем простого вычитания матрицы с эффектами из исходных данных, в связи с этим, часто такие значения называются *остатками* (англ. *residuals*). Матрица остатков показана на рисунке оранжевым цветом. По сути, мы разделили матрицу данных на сумму двух матриц — эффектов (сигнал) и остатков (шум).

Чтобы правильно вычислить дисперсию для остатков нужно вспомнить, что изначально число степеней свободы было равно 15. Затем мы вычислили общее среднее и вычли его, потеряв одну степень. После этого мы потеряли две степени свободы на сравнение групповых средних. Т.е. в итоге число степеней свободы сократилось до 12, что мы и будем использовать для оценки дисперсии, которая в этом случае равна 2.5.

Теперь у нас есть оценки дисперсии и для сигнала, и для шума. Их отношение в дисперсионном анализе называется F-значение, или F-статистика. Если нулевая гипотеза верна, то эта величина хорошо

описывается распределением Фишера с двумя параметрами — числом степеней свободы, которые мы использовали для вычисления каждой дисперсии.

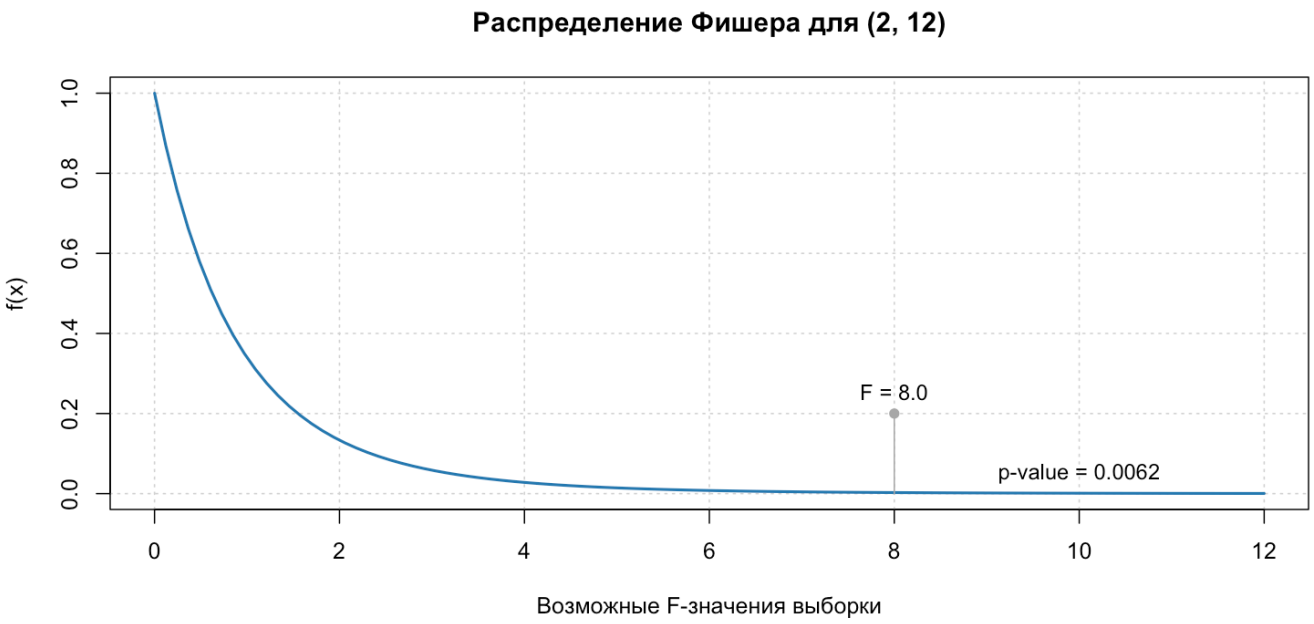


Рис. 1.29. Распределение Фишера для примера со сравнением методов очистки воды. Точка соответствует статистике (F-значению), полученной для экспериментальных данных.

F-значение для нашего примера равно $20/2.5 = 8$, т.е. дисперсия сигнала в 8 раз больше дисперсии шума. Это довольно много, чтобы быть случайным эффектом. Очевидно, что в идеальном случае, если наша нулевая гипотеза верна, то F-значение должно равняться нулю. Но в реальном мире, особенно когда число экспериментов ограничено, всегда есть шанс получить ненулевое F-значение. Однако, чем больше это значение, тем меньше шанс его получить. График на рисунке показывает распределение F-значений для нашего примера (DoF = 2 и 12).

Статистика, полученная нами, обозначена точкой, а область под кривой справа от нее соответствует вероятности получить F-значение такое же, как наше, или больше, если H0 верна. Т.е. это и есть p-значение для нашего теста, в этом случае оно равно 0.0062, т.е. шанс примерно 6 из 1000. В данном случае будет логичным отвергнуть нулевую гипотезу и констатировать наличие разницы между средней эффективностью методов.

Собственно, результаты ANOVA позволяет нам предположить, что гипотеза о том, что методы имеют одинаковую эффективность скорее всего неверна. Однако, дисперсионный анализ не позволяет увидеть детали — какой из методов на самом деле наиболее эффективен и насколько велик может быть эффект от смены одного метода другим. Для этого понадобится еще один тест — тест Тьюки.

Требования к качеству данных

Использование дисперсионного анализа накладывает определенные ограничения на качество данных. Во-первых, предполагается, что значения в каждой популяции распределены нормально. Надо сказать, что небольшое отклонение от нормального распределения почти не влияет на результат, но, тем не менее, проверить нормальность ваших данных важно.

Однако, особенно при наличии большого числа групп, это не всегда просто, поэтому наиболее приемлемый вариант будет провести анализ и исследовать распределение остатков. Если модель построена правильно, учтены все факторы, которые могут давать систематический эффект, то остатки для такой модели должны представлять собой случайные числа. И, если исходные данные были получены из нормально распределенных генеральных совокупностей, то и остатки также будут удовлетворять этому условию. Используйте график квантиль-квантиль и тест Шапиро-Уилка для этого.

Второе важное требование — это однородность дисперсий в группах. Другими словами, мы предполагаем, что дисперсия, или разброс значений в разных группах, а точнее в соответствующих генеральных совокупностях, одинакова. Оценить это можно визуально, с помощью диаграмм размаха, или с помощью специального теста, например, с помощью теста Бартлета, или Левене. И визуальную оценку, и тест лучше всего также проводить для остатков.

1.6.3 Тест Тьюки

Тест достоверно значимой разности Тьюки (англ. *Tukey's honest significance difference test*) очень похож на попарный тест Стьюдента, который мы рассматривали в конце предыдущего раздела. С двумя важными поправками:

1. Стандартная ошибка вычисляется на основе дисперсии остатков, полученной в ходе дисперсионного анализа. В связи с этим часто тест Тьюки используется после дисперсионного анализа, если последний обнаружил возможное наличие какого-то эффекта. По этой причине его называют апостериорным тестом (англ. *post-hoc test*).
2. Для оценки p-значения используется стьюдентизированное распределение диапазона (англ. *Studendized range distribution*). Это распределение подразумевает, что выборки взяты из нескольких популяций с одинаковыми параметрами, т.е. учитывает количество тестов, что решает проблему множественных сравнений.

В остальном использование теста Тьюки идентично попарному сравнению с помощью t-теста. Для каждой пары вычисляется доверительный интервал и p-значения для H0: $\mu_1 = \mu_2$.

В таблице ниже показаны результаты теста Тьюки для нашего примера, рассмотренного выше.

	$m_1 - m_2$	2.5%	97.5%	p-значение
B-A	2	-0.6679	4.6679	0.1546
C-A	-2	-4.6679	0.6679	0.1546
C-B	-4	-6.6679	-1.3321	0.0046

Для каждой пары результаты, собранные в таблице, показывают разницу средних значений, полученную в эксперименте (например, в случае B и A можно видеть, что метод B в нашем эксперименте был на 2% эффективнее), доверительный интервал для этой разницы (в нашем случае для доверительной вероятности 95%) и p-значения для H0: $\mu_1 = \mu_2$.

Как мы можем заметить, только наблюдаемая разница между средней эффективностью методов C и B достаточно велика, чтобы быть полученной из популяций с одинаковой эффективностью (шанс для этого составляет 0.46%). Разница, наблюдаемая для остальных пар вполне может быть получена случайным образом и мы не можем отвергнуть нулевую гипотезу в этих двух случаях.

Другими словами, если вы уже используете метод A для очистки воды, то повода менять его на другой метод нет. Если же вы используете метод C, то имеет смысл поменять его на метод B. В этом случае вы можете ожидать улучшение эффективности в среднем от 1.3% до 6.7%.

1.6.4 Многофакторный дисперсионный анализ

В многофакторном дисперсионном анализе подразумевается не один, а несколько источников систематической вариации данных. Количество факторов теоретически может быть любым, однако следует помнить два важных момента:

1. Чем больше факторов, тем большее измерений необходимо для анализа. Каждый раз, когда вы добавляете фактор с M дискретными значениями (уровнями), вам нужно $M - 1$ дополнительных степеней свободы.
2. Чтобы эффекты от факторов были правильно оценены, факторы должны варьироваться независимо. Другими словами, не должно быть взаимной корреляции между их уровнями. Этого можно добиться с помощью грамотного планирования (например с помощью латинских квадратов, или полнофакторного эксперимента).

Если оба условия соблюдены, то процедура дисперсионного анализа не сильно отличается от однофакторного случая. Изменим немного пример, рассмотренный нами ранее. Будем теперь считать, что на эффективность очистки воды влияет не только использованный метод, но и ее кислотность. Пусть у нас имеется возможность регулировать кислотность воды перед очисткой, так, чтобы pH был около 6, или около 8. Задача теперь заключается в одновременной проверки двух гипотез:

1. Влияют ли методы на эффективность очистки, H0: $\mu_A = \mu_B = \mu_C$
2. Влияет ли кислотность воды на эффективность очистки, H0: $\mu_{pH=6} = \mu_{pH=8}$

Для того, чтобы протестировать обе гипотезы одновременно, мы провели полнофакторный эксперимент в котором протестировали все комбинации двух факторов по три раза. Результаты эксперимента показаны на рисунке 1.30 (первая таблица).

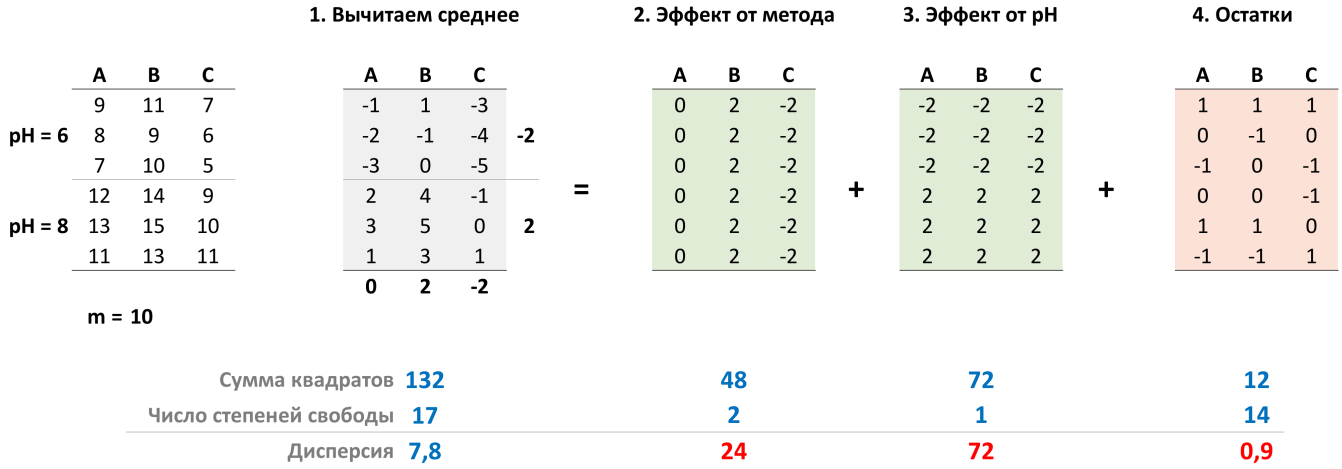


Рис. 1.30. Основные этапы многофакторного дисперсионного анализа.

Как можно видеть из таблицы, в среднем, более высокое значение pH дает более высокую эффективность очистки (на 4%), тогда как средняя разница между отдельными методами такая же, как и в предыдущем примере.

Собственно, логика и вычисления в этом случае очень похожи. Единственное исключение в том, что в этом случае нам нужно выделить две независимых друг от друга матрицы эффектов — одну для метода, и одну для кислотности воды. Соответственно, остатки могут быть получены вычитанием обеих матриц из исходных результатов (здесь, как и в первом примере, мы убрали общее смещение для упрощения вычислений), как и показано на рисунке. Ну и конечно, число степеней свободы будет немного отличаться, так как в этом случае число изначальных независимых измерений равно 18.

Теперь мы можем вычислить два F-значения и оценить их для каждого фактора независимо. На рисунке значение дисперсии для матрицы остатков округлено до одного десятичного знака, в реальности оно равно $12/14 \approx 0.857$, что мы и будем далее использовать для вычислений. Так, для метода F-значение будет равным $F_1 = 24/0.857 \approx 28$, что довольно много для того, чтобы быть случайным эффектом.

Соответствующее p-значение может быть вычислено как:

$$p = 1 - F(28, DoF1 = 2, DoF2 = 14) \approx 0.000013$$

Для кислотности воды F-значение равно $F_2 = 72/0.857 = 84$ и соответствующее p-значение будет очень маленьким, практически нулевым.

Т.е., в данном случае можно констатировать, что и метод и кислотность воды влияют на эффективность очистки. Насколько велика эта эффективность, можно оценить с помощью теста Тьюки, который нужно сделать для каждого фактора отдельно.

Самое интересное, что если вдруг в этом случае мы "забудем" учесть влияние кислотности в нашей ANOVA модели, то этот вклад будет засчитан, как шум и, соответственно, дисперсия шума будет равна $(72 + 12)/(1+14) = 5.6$. Это приведет к тому, что F-значение для эффекта, обусловленного использованием разных методов очистки, станет меньше почти в 6 раз и p-значение составит примерно 3.4%. Т.е., если изначально наш уровень значимости был равен 0.01, мы не сможем отвергнуть H0 в этом случае.

Избежать таких ситуаций частично позволяет изучение распределений остатков, о котором мы писали чуть выше. Так, график квантиль-квантиль часто позволяет обнаружить систематическое поведение значений остатков, что напрямую указывает на наличие "забытого" фактора.

Взаимодействие между факторами

При наличии нужного числа степеней свободы и правильно спланированного эксперимента, ANOVA также позволяет оценить наличие взаимодействия между факторами. Взаимодействие означает, что один фактор может влиять на эффект от другого фактора.

Для нашего примера это может означать, например, что для низкой кислотности эффективность метода A будет наилучшей, а для высокой — наиболее эффективен метод B. Или для одной кислотности методы хорошо различаются по эффективности, а для другой — нет.

Как сделать такой анализ технически мы покажем в следующем подразделе.

1.6.5 Дисперсионный анализ в R

Однофакторный анализ

Для дисперсионного анализа в R проще всего использовать функцию `aov()`. Для этого данные должны быть представлены в виде отдельных векторов или фрейма. Очень важно, чтобы векторы, или столбцы со значениями факторов были представлены в формате `factor` в R.

Для иллюстрации повторим примеры, рассмотренные нами ранее. Начнем с однофакторного случая и первым делом организуем данные правильно, так, чтобы переменные были представлены в виде отдельных столбцов.

```r
# измеренные значения в виде вектора
Efficiency <- c(9, 8, 10, 12, 11, 12, 14, 11, 10, 13, 7, 8, 10, 6, 9)

# вектор с номерами использованных методов
Method <- c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3)

# преобразуем эти значения в реальный фактор
# и добавляем названия методов в виде меток
Method <- factor(Method, labels = c("A", "B", "C"))

# объединяем все во фрейм с данными
d1 <- data.frame(Efficiency, Method)
```

Теперь можно выполнить анализ.

```r
m1 <- aov(Efficiency ~ Method, data = d1)
summary(m1)
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
Method       2     40    20.0       8 0.0062 **
Residuals   12     30     2.5
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Функция `aov()` использует специальный ввод, который называется формулой. Знак ~ в формуле означает отношение, или зависимость, в данном случае формула читается как: “зависимость переменной Efficiency от значений переменной Method”. Аргумент `data` указывает, что обе переменных — это столбцы фрейма с данными `d1`.

Результаты анализа выводятся с помощью функции `summary()`, но отдельные его части могут быть извлечены из объекта `res`, который по сути является списком. В таблице с результатами строки соответствуют элементам разложения данных (фактор `Method` и остатки `Residuals`). Столбцы означают число степеней свободы (`Df`), сумму квадратов значений (`Sum Sq`), дисперсию, или среднеквадратичное отклонение (`Mean Sq`), F-значение и p-значение. Как можно видеть, все значения в таблице совпадают с показанными нами на рисунке 1.28.

Последняя строка под таблицей, показывает легенду, как интерпретировать символы, приведенные после столбца с p-значением. В нашем случае там находится **, что означает, что нулевая гипотеза будет

отвергнута на уровне значимости 0.01. Постарайтесь не пользоваться этой легендой, устанавливайте уровень значимости (если он нужен) до того, как начнете эксперимент, и не меняйте его до конца.

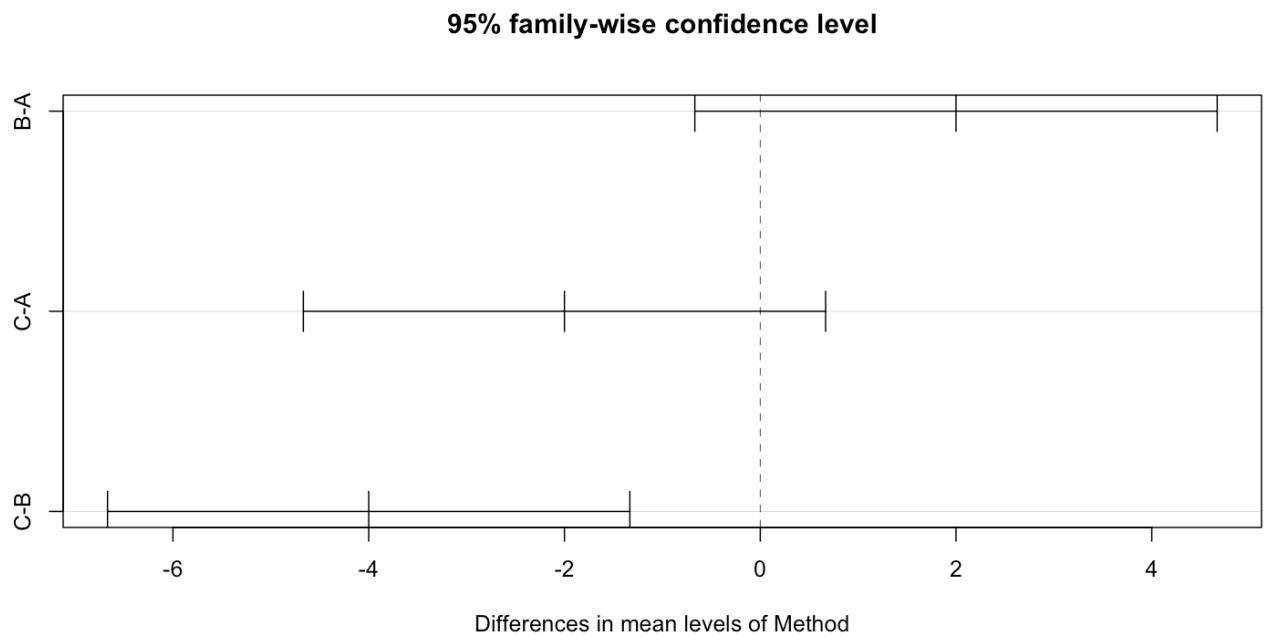
Так как p-значение, полученное в ходе дисперсионного анализа, довольно мало, мы можем оценить величину эффекта с помощью теста Тьюки, как показано в блоке кода ниже. Таблица, полученная в результате теста, идентична показанной нами ранее в этом разделе.

```
res1 <- TukeyHSD(m1)
show(res1)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Efficiency ~ Method, data = d1)

$Method
    diff        lwr        upr     p adj
B-A    2 -0.6678637  4.6678637 0.1545800
C-A   -2 -4.6678637  0.6678637 0.1545800
C-B   -4 -6.6678637 -1.3321363 0.0046341
```

Заметьте, что мы не использовали данные для `TukeyHSD()`, все нужные статистики эта функция берет из результатов дисперсионного анализа, проведенного нами ранее. Результаты теста Тьюки можно также представить графически.

```
plot(res1)
```



На этом графике показаны доверительные интервалы разности средних значений для каждой пары и референсное значение (0) в виде вертикальной линии.

Многофакторный анализ

Код ниже содержит подготовку данных для второго примера с двумя факторами.

```
# измеренные значения в виде вектора
Efficiency <- c(9, 8, 7, 12, 13, 11, 11, 9, 10, 14, 15, 13, 7, 6, 5, 9, 10, 11)

# вектор с номерами использованных методов
Method <- c(1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3)

# преобразуем эти значения в реальный фактор и добавляем названия методов в виде меток
Method <- factor(Method, labels = c("A", "B", "C"))

# вектор с со значениями кислотности
pH <- c(6, 6, 6, 8, 8, 8, 6, 6, 6, 8, 8, 8, 6, 6, 6, 8, 8, 8)

# преобразуем эти значения в фактор
```

```
pH <- factor(pH)

# объединяем все во фрейм с данными
d2 <- data.frame(Efficiency, Method, pH)
```

Код для анализа выглядит следующим образом.

```
m2 <- aov(Efficiency ~ Method + pH, data = d2)
summary(m2)
```

```
            Df Sum Sq Mean Sq F value   Pr(>F)
Method       2     48   24.00      28 1.28e-05 ***
pH           1     72   72.00      84 2.72e-07 ***
Residuals   14     12    0.86
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Как мы видим, единственная разница в том, что формула теперь указывает на зависимость эффективности методов от суммы двух эффектов. Соответственно, в результирующей таблице для каждого фактора теперь отдельная строка.

Чтобы оценить величину эффекта для каждого фактора, можно также воспользоваться тестом Тьюки. В этом случае нужно передать второй аргумент для функции `TukeyHSD()` — имя фактора, для которого нужно сделать тест. Имя должно полностью совпадать с тем, которое использовалось в формуле функции `aov()`. Пример ниже показывает, как сделать тест Тьюки для эффекта, вызванного изменением кислотности воды.

```
res2 <- TukeyHSD(m2, "pH")
print(res2)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Efficiency ~ Method + pH, data = d2)

$pH
    diff      lwr      upr p adj
8-6    4 3.063938 4.936062 3e-07
```

Как видно из итоговой таблицы, можно ожидать улучшения эффективности очистки на величину от 3 до 5 процентов, если снизить кислотность с 8 до 6.

Для того, чтобы протестировать эффект взаимодействия между факторами, необходимо добавить еще один элемент в формулу, как показано в примере ниже:

```
m3 <- aov(Efficiency ~ Method + pH + Method:pH, data = d2)
summary(m3)
```

```
            Df Sum Sq Mean Sq F value   Pr(>F)
Method       2     48      24      24 6.40e-05 ***
pH           1     72      72      72 2.05e-06 ***
Method:pH    2      0       0       0        1
Residuals   12     12       1
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

В данном случае, очевидно, эффект взаимодействия между факторами отсутствует, т.е. разница между эффективностью методов не зависит от кислотности воды.

Пример полного анализа реальных экспериментальных данных

Наконец, приведем пример более или менее полного анализа экспериментальных данных, включая предварительный обзор, дисперсионный анализ, тест на однородность дисперсии, проверку остатков, и так далее.

Начнем с собственно данных. Они были получены с помощью химического эксперимента, в котором изучалось влияние трех факторов на величину выходы конечного продукта некоторой реакции. Каждый фактор имел два уровня, значения для которых показаны в таблице ниже:

Фактор	Уровень 1	Уровень 2
Температура, °C	160	180
Концентрация катализатора, %	10	20
Тип катализатора	A	B

Далее, в ходе эксперимента были опробованы все возможные комбинации факторов (8), и каждая комбинация использовалась два раза, таким образом, в результате у нас имеются данные о выходе продукта для 16 реакций, как показано в таблице ниже:

T, °C	C, %	K	y
160	20	A	(59, 61)
180	20	A	(74, 70)
160	40	A	(50, 58)
180	40	A	(69, 67)
160	20	A	(50, 54)
180	20	A	(81, 85)
160	40	A	(46, 44)
180	40	A	(97, 81)

Последний столбец содержит величину выхода продукта, полученную для каждой реакции. Важно, что все 16 реакций были опробованы в случайном порядке, чтобы исключить возможные случайные корреляции с внешними факторами.

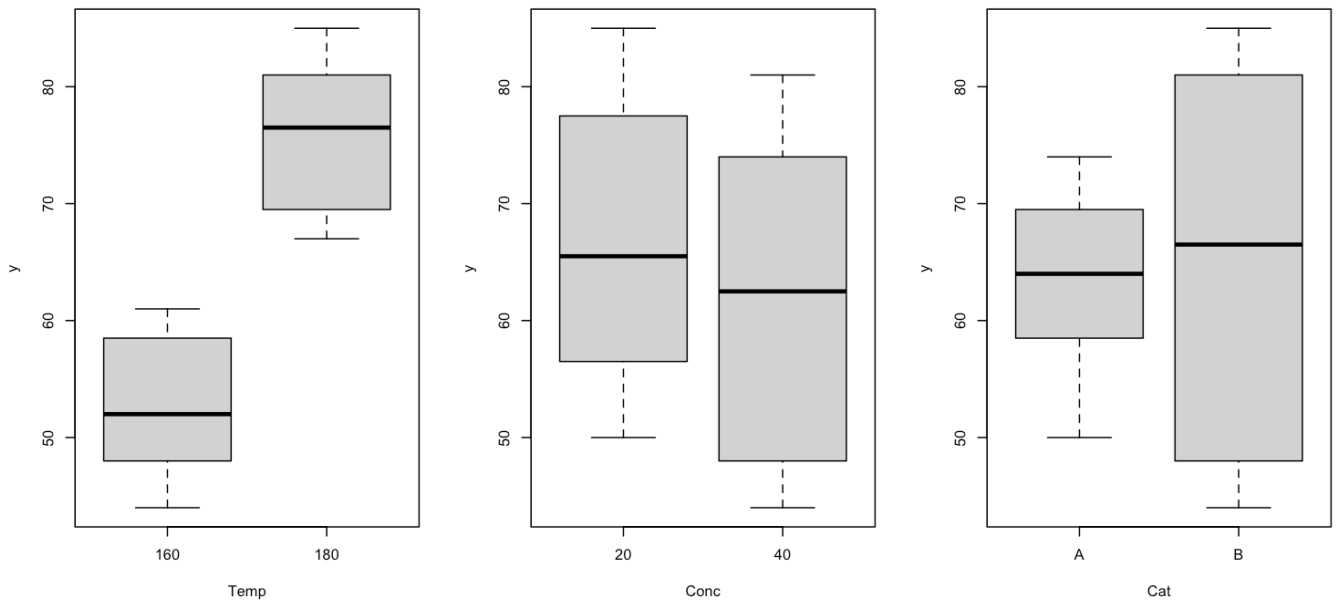
Блок кода ниже показывает, как подготовить данные к анализу. В этом случае, так как все данные вводятся вручную, мы используем функцию `rep()` для того, чтобы повторить некоторые последовательности чисел и символов вместо того, чтобы копировать их.

```
# векторы с комбинациями факторов
Temp <- as.factor( rep(c(160, 180), 4) )
Conc <- as.factor( rep(c(20, 20, 40, 40), 2) )
Cat  <- as.factor( c(rep("A", 4), rep("B", 4)) )

# измеренные значения выхода продукта реакции для каждой комбинации
y1 <- c(59, 74, 50, 69, 50, 81, 46, 79)
y2 <- c(61, 70, 58, 67, 54, 85, 44, 81)

# делаем два фрейма для каждой серии экспериментов и потом объединяем их
d1 <- data.frame(Temp, Conc, Cat, y = y1)
d2 <- data.frame(Temp, Conc, Cat, y = y2)
d <- rbind(d1, d2)
```

Начнем с общего визуального анализа данных. Воспользуемся диаграммой размаха, чтобы оценить величину эффекта для каждого фактора.

```
x = 1
par(mfrow = c(1, 3))
```

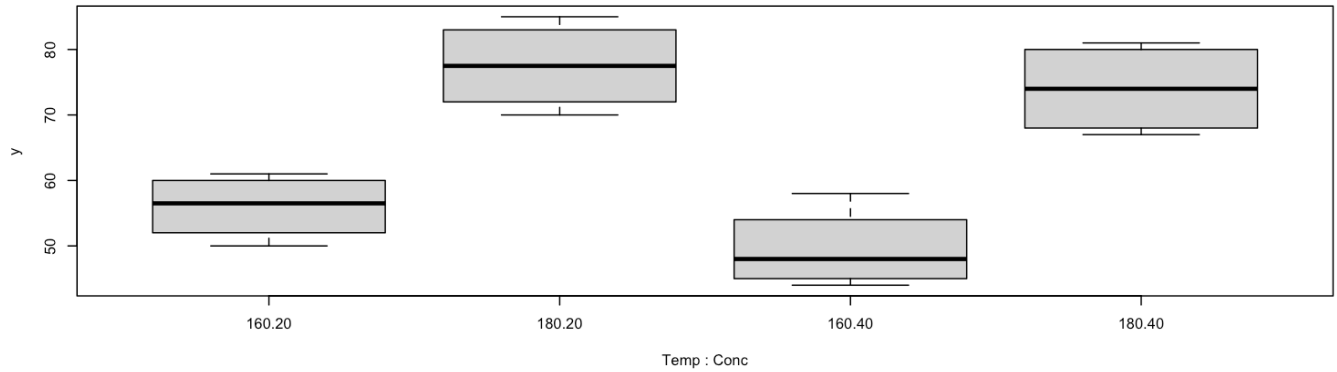
```
boxplot(y ~ Temp, data = d)
boxplot(y ~ Conc, data = d)
boxplot(y ~ Cat, data = d)
```

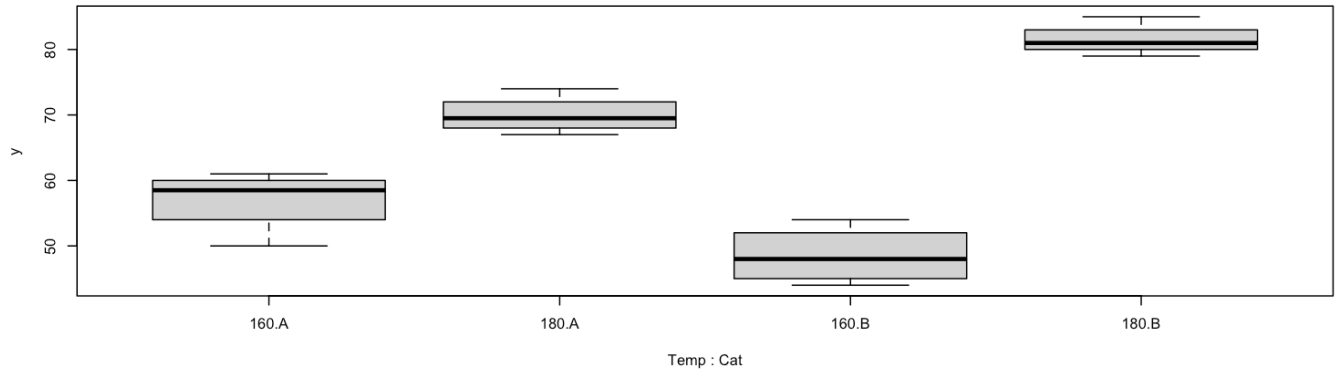


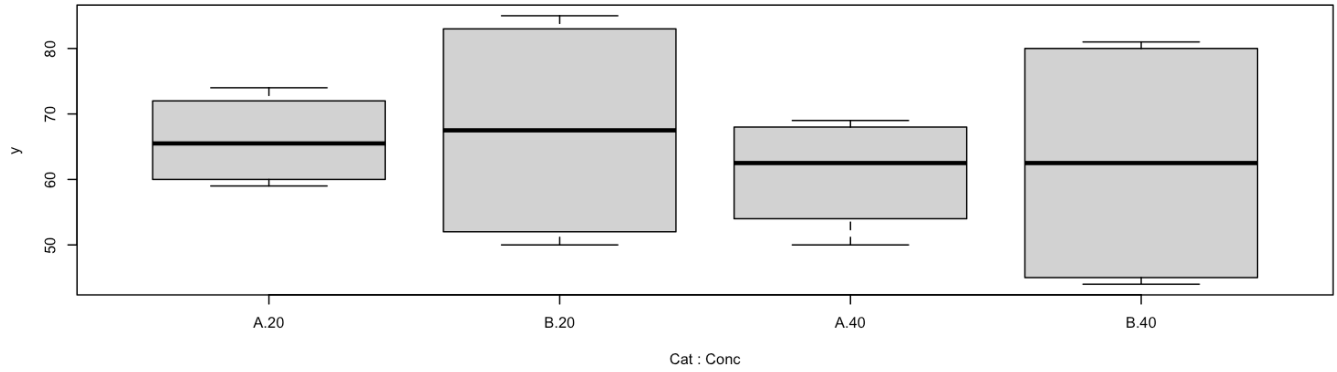
Очевидно, что имеется явный эффект от повышения температуры — диаграммы значений полученные для 160 °C и для 180 °C не пересекаются. Если сравнить медианные значения, можно также видеть небольшой негативный эффект от увеличения концентрации катализатора, и небольшой позитивный эффект от смены катализатора. Однако, в последнем случае размах значений довольно велик и, возможно, наблюдаемый эффект, на самом деле, случаен. Для того, чтобы проверить это и нужен дисперсионный анализ, который мы сделаем чуть позже.

Теперь построим диаграммы размаха для попарной комбинации факторов. Это позволит визуально оценить возможное наличие взаимодействия между факторами.

```
par(mfrow = c(3, 1))
boxplot(y ~ Temp + Conc, data = d)
boxplot(y ~ Temp + Cat, data = d)
boxplot(y ~ Cat + Conc, data = d)
```







Рассмотрим графики по порядку. Первый график показывает, как меняется эффект от температуры для разных значений концентрации катализатора. Очевидно, и при 20%, и при 40% имеется положительный эффект, и величина эффекта примерно одинаковая. Т.е., скорее всего взаимодействие между этими двумя факторами отсутствует.

Второй график показывает, как эффект от температуры зависит от типа используемого катализатора. В

данном случае мы можем видеть, что эффект от повышения температуры для катализатора A (первые две диаграммы) меньше, чем тот же самый эффект, полученный при использовании катализатора B. Является ли наблюдаемое взаимодействие случайным, или нет, опять же мы сможем проверить с помощью дисперсионного анализа. Пока же мы можем предположить, что взаимодействие имеет место.

Последний график выглядит довольно случайным и не показывает явных трендов, разница между медианными значениями гораздо меньше, чем разброс значений внутри каждой группы. Что не позволяет сделать никаких выводов из графика.

Следующий шаг — построить ANOVA модель с учетом возможных взаимодействий между факторами, и проанализировать результаты. Для начала убедимся, что у нас хватает на это измерений. У нас имеется три основных эффекта (по одному для каждого фактора) и три потенциальных эффекта от взаимодействия факторов. Так как в каждом факторе есть только два уровня, необходимо использовать одну степень свободы для каждого эффекта. Кроме этого, нам нужна одна степень свободы для вычисления общего среднего, итого семь степеней свободы. Также нам нужно несколько степеней свободы для оценки дисперсии остатков.

Теперь понятно, почему в этом случае нужно было повторить реакцию для каждой комбинации факторов два раза, если этого не сделать, то общее число значений (8) не позволит надежно оценить эффекты от возможного взаимодействия факторов. В нашем же случае измерений хватает с лихвой.

Код ниже показывает результаты дисперсионного анализа.

```
m <- aov(y ~ Temp + Conc + Cat + Temp:Conc + Temp:Cat + Cat:Conc, data = d)
summary(m)
```

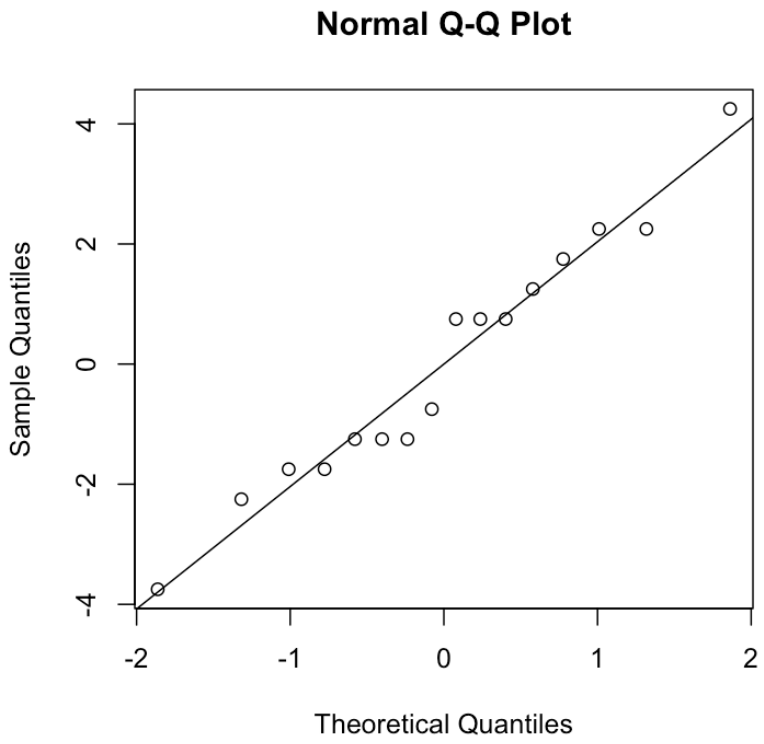
```
            Df Sum Sq Mean Sq F value   Pr(>F)
Temp         1   2116  2116.0 292.985 3.57e-08 ***
Conc         1    100   100.0  13.846  0.00476 **
Cat          1      9     9.0   1.246  0.29320
Temp:Conc    1      9     9.0   1.246  0.29320
Temp:Cat     1    400   400.0  55.385 3.92e-05 ***
Conc:Cat     1      0     0.0   0.000  1.00000
Residuals    9     65     7.2
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Собственно, эти результаты подтверждают наши предположения. Во-первых, очень маловероятно, что эффект от повышения температуры был случайным (p-значение практически равно нулю). Для эффекта от концентрации p-значение составляет 0.4%, что тоже маловероятно. И, наконец, все указывает на то,

что наблюдаемый эффект взаимодействия между температурой и типом катализатора (`Temp:Cat`) тоже не является случайным.

Для того, чтобы оценить величину каждого эффекта, потребуется тест Тьюки, но прежде убедимся, что наши данные подходят под критерии дисперсионного анализа. Сначала, посмотрим на распределение остатков.

```
par(mfrow = c(1, 1))
qqnorm(m$resid)
qqline(m$resid)
```



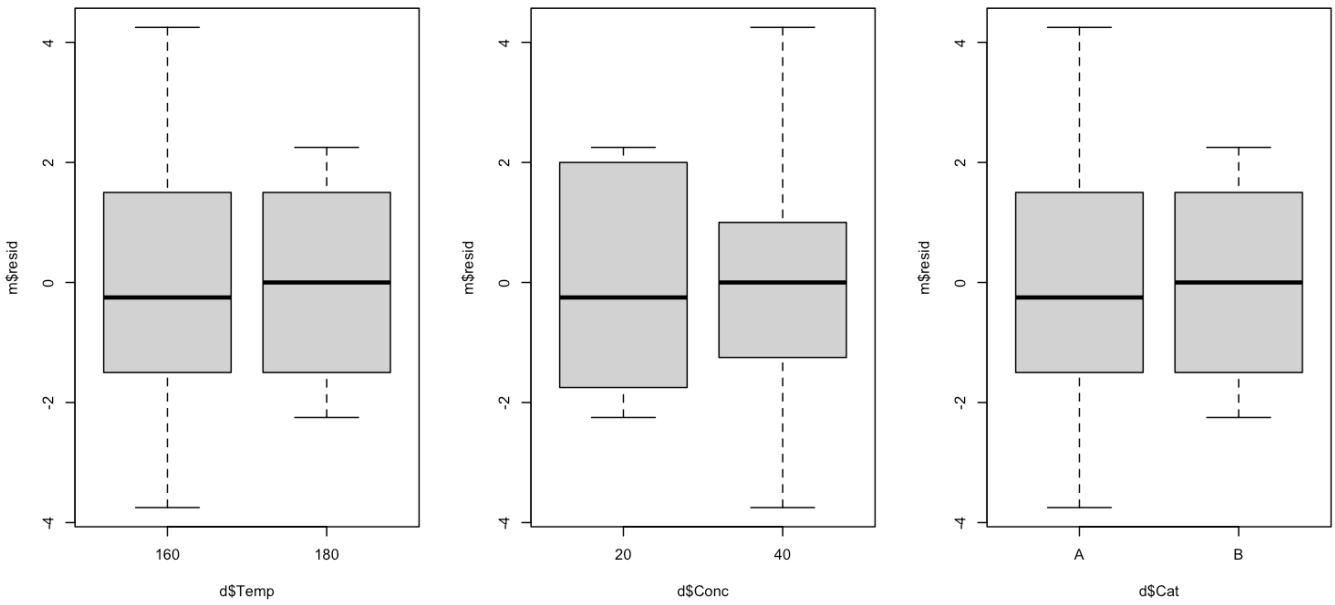

```
nres <- shapiro.test(m$resid)
show(nres)
```

```
    Shapiro-Wilk normality test

data:  m$resid
W = 0.96583, p-value = 0.7674
```

Ни график квантиль-квантиль ни тест Шапиро-Уилка не выявляют каких-то аномалий в поведении остатков. Величины выглядят случайными, выбросы отсутствуют и распределение хорошо описывается нормальным законом.

Построим диаграммы размаха остатков для каждого фактора:

```
par(mfrow = c(1, 3))
boxplot(m$resid ~ d$Temp)
boxplot(m$resid ~ d$Conc)
boxplot(m$resid ~ d$Cat)
```



Графики показывают некоторое различие в дисперсии значений, но оно довольно незначительно. Размеры “коробок” практически совпадают. Наибольшая разница наблюдается для вариации откликов, полученных на разных уровнях концентрации катализатора (средний график). Воспользуемся тестом Бартлетта для проверки этого случая:

```
bres <- bartlett.test(m$resid ~ d$Conc)
show(bres)
```

```
    Bartlett test of homogeneity of variances
```

```
data:  m$resid by d$Conc
Bartlett's K-squared = 0.22656, df = 1, p-value = 0.6341
```

Результаты теста показывают, что велика вероятность, что наблюдаемая разница является, на самом деле, случайной (p-значение = 0.634).

Таким образом, результаты дисперсионного анализа выглядят заслуживающими доверия и мы можем проанализировать величину значимых эффектов с помощью теста Тьюки. Начнем с температуры.

```
resTemp <- TukeyHSD(m, "Temp")
show(resTemp)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = y ~ Temp + Conc + Cat + Temp:Conc + Temp:Cat + Cat:Conc, data = d)

$Temp
        diff      lwr      upr p adj
180-160   23 19.96032 26.03968     0
```

Увеличение температуры со 160 до 180 градусов в нашем эксперименте привело к увеличению объема конечного продукта на 23 единицы в среднем. Мы можем ожидать, что при повторении эксперимента, мы получим прирост в количестве конечного продукта от 20 до 26 единиц, на что указывает доверительный интервал.

Результаты теста для концентрации катализатора показаны ниже.

```
resConc <- TukeyHSD(m, "Conc")
show(resConc)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = y ~ Temp + Conc + Cat + Temp:Conc + Temp:Cat + Cat:Conc, data = d)

$Conc
      diff       lwr       upr     p adj
40-20   -5 -8.039682 -1.960318 0.0047629
```

Очевидно, что имеется небольшой отрицательный эффект, увеличение концентрации привело к снижению выхода конечного продукта реакции в среднем на 5 единиц. При повторении эксперимента ожидаемое снижение варьируется от 8 до 2 единиц.

Наконец, сделаем тест Тьюки для взаимодействия между эффектом от температуры и типом катализатора. Обратите внимание на название эффекта, которое мы использовали при вызове функции `TukeyHSD()`.

```
resTempCat <- TukeyHSD(m, "Temp:Cat")
show(resTempCat)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = y ~ Temp + Conc + Cat + Temp:Conc + Temp:Cat + Cat:Conc, data = d)

$`Temp:Cat`
              diff       lwr       upr     p adj
180:A-160:A  13.0   7.06767  18.93233 0.0003571
160:B-160:A  -8.5 -14.43233  -2.56767 0.0069143
180:B-160:A  24.5  18.56767  30.43233 0.0000020
160:B-180:A -21.5 -27.43233 -15.56767 0.0000062
180:B-180:A  11.5   5.56767  17.43233 0.0008881
180:B-160:B  33.0  27.06767  38.93233 0.0000002
```

Таблица с результатами выглядит довольно объемной — она содержит все возможные комбинации. На самом деле, для оценки эффекта взаимодействия нам нужны только несколько строк, а именно: `180:A-160:A`, `180:B-160:B`, которые и показывают эффект от смены температуры при разных значениях типа катализатора. Так, в случае A наблюдаемый эффект составил 13 единиц, а ожидаемый — от 7 до 19 единиц. В случае же использования катализатора B наблюдаемый эффект составил 33 единицы, а ожидаемый — от 27 до 39 единиц. Т.е., использование катализатора B делает реакцию гораздо более чувствительной к смене температуры. Значения p, посчитанные для обеих строк показывают, что этот эффект не является случайным.

Еще одна строка, которая может оказаться полезной, это разница между `180:B` и `180:A`. В нашем случае разница составила 11 единиц и, судя по p-значению, она не является случайной. Т.е., не смотря на то, что сам по себе тип катализатора не влияет на конечный продукт, он усиливает эффект от температуры и выбор в пользу типа B выглядит разумным.

Таким образом, проведенный анализ позволяет сделать следующие выводы:

1. Температура имеет сильный положительный эффект на объем конечного продукта
2. Концентрация катализатора имеет небольшой негативный эффект
3. Тип катализатора сам по себе не влияет на результат, но делает реакцию более чувствительной к смене температуры

Величины эффектов очень важны для конечных решений. Так, очевидно, что температура имеет сильный эффект. С другой стороны, увеличение температуры повлечет за собой затраты на дополнительную энергию, что может нивелировать экономическую выгоду от этого эффекта. Зная величину эффекта, можно посчитать экономические последствия от его использования и принять окончательное решение.

2 Метод главных компонент

2.1 Отношения между переменными

В прошлой главе, помимо прочего, мы обсуждали представление данных в виде облака точек в пространстве переменных. Такое представление можно визуализировать с помощью графиков рассеяния, в которых оси соответствуют этим переменным. Мы также рассматривали примеры таких графиков для набора данных с концентрацией ионов в образцах воды, для удобства приводим их снова, см. рисунок 2.1.

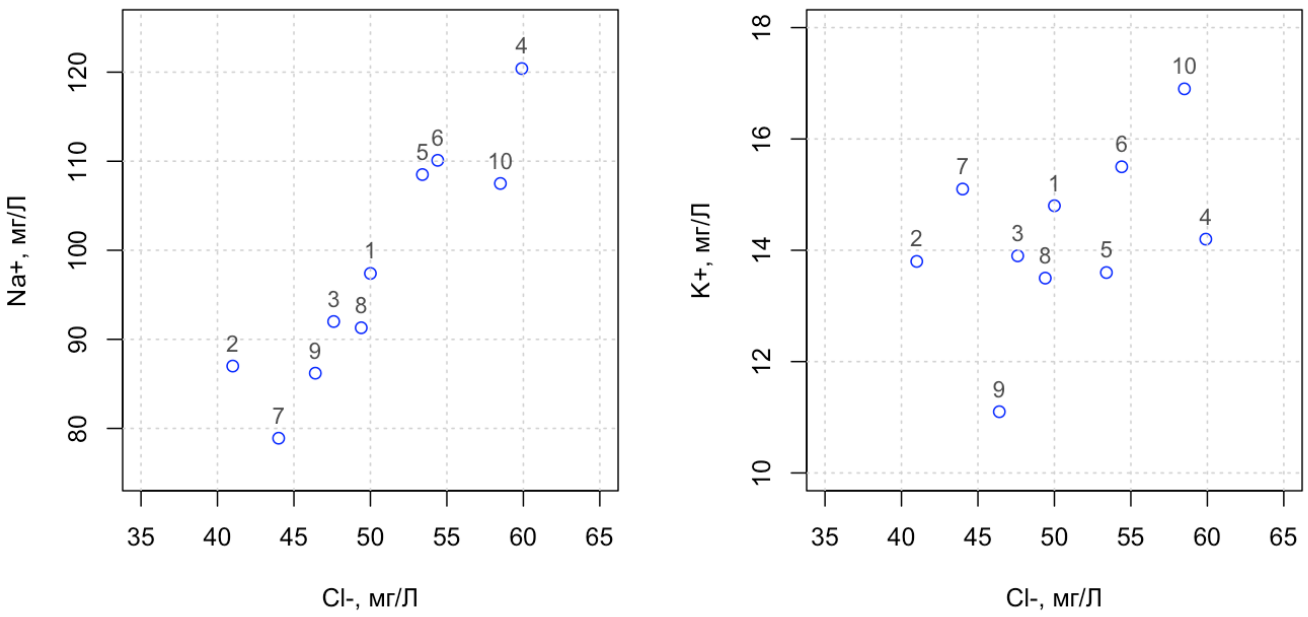


Рис. 2.1. Представление данных в пространстве переменных.

Часто точки, соответствующие индивидуальным измерениям, на таких графиках расположены не хаотично, а образуют некоторые структуры, например можно различить отдельные группы, выбросы или другие геометрические паттерны. Отдельный интерес представляют собой монотонные тренды, когда точки на графике располагаются вдоль воображаемой кривой, или прямой, как, например, на графике слева (рис. 2.1). Наличие такого тренда часто указывает на то, что переменные, представленные на графике,

связаны определенным образом и когда значение одной переменной меняется в большую, или меньшую сторону, это влечет за собой изменение значения второй переменной.

Такая связь между двумя переменными может быть обусловлена несколькими причинами, которые перечислены ниже.

1. Наличие прямой причинно-следственной зависимости между переменными, т.е. изменение одной переменной напрямую влияет на величину второй.
2. Наличие некоторого скрытого фактора, связанного с обеими переменными. Т.е. между двумя переменными нет прямой зависимости, но они обе зависят от некоторой третьей переменной.
3. Случайные зависимости, когда наблюдаемый тренд является случайным явлением. Это обычно характерно для выборок с относительно небольшим числом измерений.

Если с третьей причиной все понятно, то первые две часто путают. Приведем два графика рассеяния, которые сделаны на основе набора данных, собранных для случайно отобранных 32 человек — жителей Европейских стран. В этом наборе имеются различные характеристики этих людей, включая рост, вес, количество потребляемого в год пива и умение плавать. Именно эти четыре характеристики мы и взяли для построения графиков показанных на рисунке 2.2.

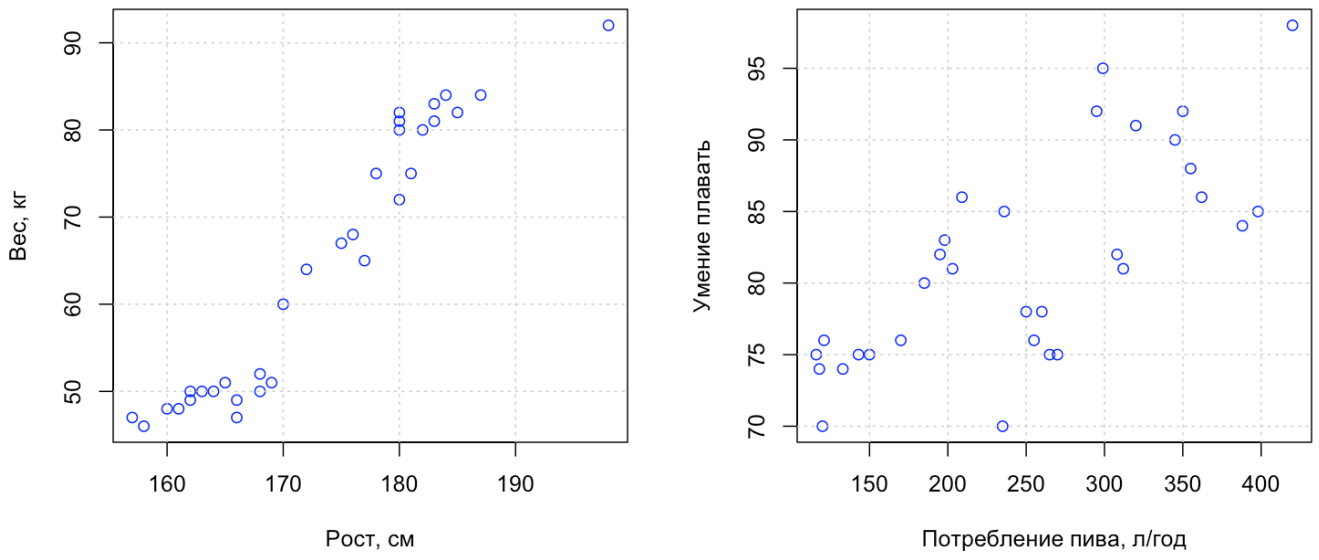


Рис. 2.2. Графики рассеяния для набора данных *Люди.*

График слева показывает зависимость между ростом и весом людей из выборки. Очевидно, что точки располагаются очень близко к некоторой воображаемой прямой, в таких случаях мы говорим, что имеется

линейная зависимость между этими переменными. Наши знания об анатомии человека позволяют утверждать, что это прямая зависимость, и что люди с высоким ростом в среднем будут весить больше людей со средним и маленьким ростом. Т.е., в данном случае имеется научно объяснимая причинно-следственная связь между этими двумя характеристиками.

График справа показывает зависимость умения плавать (в этом случае используется некий индекс от 0 до 100 основанный на том, как быстро человек проплывает дистанцию в 500 м.) от годового потребления пива (в литрах). Опять же, хорошо видно, что между этими двумя переменными также имеется линейная зависимость. Она не такая явная, как в случае роста и веса, но довольно хорошо различимая. Значит ли это, что для того, чтобы стать хорошим пловцом, нужно пить больше пива?

Думаю можно найти людей, которые бы обрадовались этому утверждению и охотно приняли бы его на веру. Хотя здравый смысл подсказывает, что наличие такой связи маловероятно, по крайней мере ее нельзя объяснить. Дело в том, что в этом случае мы имеем дело со второй причиной наличия связи между переменными — когда эта связь обусловлена тем, что обе переменные связаны с третьим скрытым фактором.

Часть этой выборки — мужчины из Скандинавии, которые с одной стороны не прочь выпить пива, а с другой — хорошо развиты физически и являются хорошими пловцами. Другая часть данных — это женщины, которые пьют гораздо меньше пива, но и (здесь мы говорим именно о данной выборке) физически менее развиты — большинство из женщин в нашей выборке имеют рост меньше 170 см и вес ниже 55 кг. Их можно различить в виде плотной группы точек в нижнем левом углу первого графика. Т.е. прямой зависимости между потреблением пива и умением плавать нет, как мы и предполагали. Более физически развитые люди плавают лучше и, так уж случилось в нашей выборке, пьют больше пива.

Очень часто таким скрытым фактором является время. Если набрать в поисковой системе запрос “spurious relationship” (этим термином и обозначают обычно подобные эффекты), то можно найти очень много забавных примеров.

Очевидно, что графики рассеяния являются отличным инструментом для выявления подобных зависимостей. Однако есть две проблемы:

- Во-первых, как оценить эту зависимость? Мы уже несколько раз употребляли слова “явная” или “сильная” зависимость, однако, как измерить ее “силу” и выразить ее в числовой форме?
- Во-вторых, что делать, если число переменных больше двух? Понятно, что можно просто перебрать все переменные попарно, но чем больше переменных, тем больше пар они могут образовывать. Так, для трех переменных потребуется изучить три графика, для четырех — шесть. Если же число переменных равно 20, то число возможных пар равно 190.

В этой главе мы расскажем как решить обе проблемы. Начнем с первой.

2.2 Ковариация и корреляция

Прежде всего нужно вспомнить, что работая с экспериментальными данными мы имеем дело со случайными значениями. Другими словами, даже если есть сильная зависимость между двумя переменными, их значения всегда будут содержать случайную составляющую (часто также называемую шумом). Для того, чтобы учесть это, воспользуемся очень простым подходом — будем сравнивать, как далеко измеренные значения находятся от среднего для обеих переменных. Вычисляемая таким образом статистика называется *ковариацией* (англ. *covariance*), ее формула показана ниже:

$$cov(x, y) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - m_x)(y_i - m_y)$$

Другими словами, для каждого измерения, i, мы вычисляем два расстояния — как далеко значение переменной x (например, рост конкретного человека) находится от среднего m_x, и как далеко значение y (например, вес этого человека) располагается от среднего m_y. После этого эти два расстояния перемножаются. Ковариация — это среднее значение произведения этих двух расстояний, полученного для всех измерений в выборке. Если посчитать ковариацию, используя в качестве x и y одну и туже переменную, то получится хорошо нам известная дисперсия.

Очевидно, что если значение одной из двух переменных меньше соответствующего среднего, то расстояние будет отрицательным. При этом, если это верно для обеих переменных, то произведение будет положительным. Это можно представить визуально на графике, показанном на рисунке 2.3.

Штриховые линии показывают расположение средних значений для x и y (среднее равно 6.5 для обеих переменных в этом случае). Точки, показанные в виде треугольников, имеют значения переменной x меньше среднего (соответственно расстояние будет отрицательным), а значения переменной y больше среднего. Т.е. произведение двух расстояний для этих точек будет меньше нуля. Рассуждая таким же образом, можно отметить, что точки в виде звездочек будут также иметь отрицательные расстояния, а точки, обозначенные в виде окружностей и квадратов, имеют положительные значения. Для удобства случаи, в которых произведение двух расстояний будет положительным, показаны красным цветом, а для отрицательных используется синий.

Т.е., если рассчитать ковариацию используя только точки показанные синим цветом, то ковариация будет отрицательной, а если сделать это только для красных точек — положительной. Если же посчитать ковариацию для всех точек на графике, то она будет равна нулю. Величина ковариации зависит от того, насколько далеко точки находятся от среднего значения.

Можно доказать математически, что если зафиксировать разброс точек, то ковариация будет максимальной, если все точки лежат на прямой с ненулевым угловым коэффициентом (тангенсом угла между прямой и осью x). Если угловой коэффициент положительный, то и ковариация будет положительной, в противном

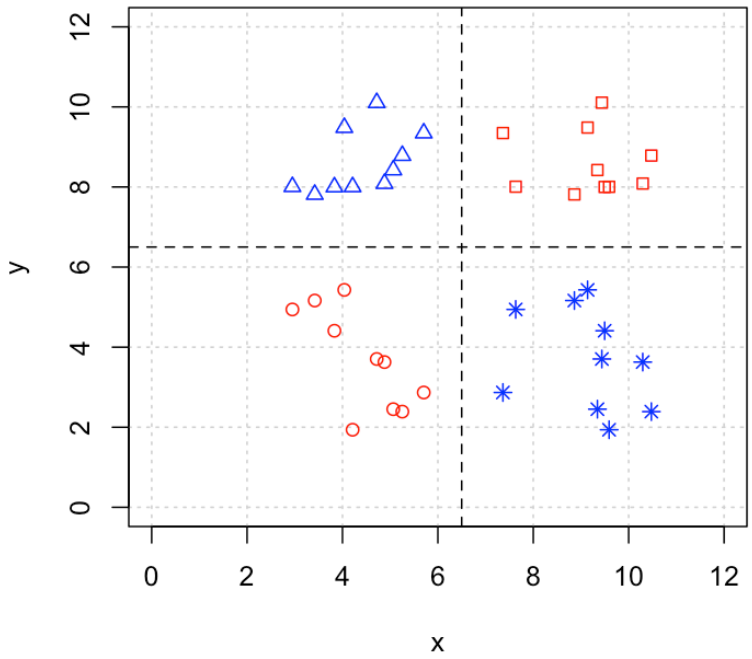


Рис. 2.3. График рассеяния для иллюстрации расчета ковариации.

случае ковариация будет отрицательной, означая, что большим значениям переменной x соответствуют маленькие значения переменной y и наоборот.

Покажем расчет ковариации на примере данных с концентрацией ионов в воде, как показано в таблице ниже.

$x : Cl-$	$y : Na+$	$x - m_x$	$y - m_y$	$(x - m_x)(y - m_y)$
50	97.4	-0.5	-0.5	0.2
41	87	-9.5	-10.9	103.4
47.6	92	-2.9	-5.9	17
59.9	120.4	9.4	22.5	212.1
53.4	108.5	2.9	10.6	31.1
54.4	110.1	3.9	12.2	47.9
44	78.9	-6.5	-19	122.9
49.4	91.3	-1.1	-6.6	7
46.4	86.2	-4.1	-11.7	47.6
58.5	107.5	8	9.6	76.9
504.6	**979.3**	**0**	**0**	**666.3**

Первые два столбца содержат исходные данные — концентрации хлорид-ионов и ионов натрия, которые мы для краткости обозначим, как x и y, соответственно. Следующие два столбца содержат значения расстояний этих значений от среднего, тогда как последний столбец содержит произведение этих расстояний. Последняя строка в таблице, где числа выделены полужирным начертанием, содержит сумму значений соответствующих столбцов.

Как можно видеть из таблицы, между выбранными двумя переменными имеется устойчивая положительная связь — для всех 10 измерений меньшему значению первой переменной соответствует меньшее значение второй, и наоборот. В результате все произведения двух расстояний положительны. Чтобы посчитать ковариацию, нужно взять сумму произведений двух расстояний (666.3) и разделить на 9 ($n - 1$), что даст нам:

$$cov(x, y) = \frac{1}{9} 666.3 \approx 74$$

Это довольно большое число, не так ли? Значит ли это, что в нашем случае ковариация между двумя переменными очень сильная? На этот вопрос нельзя ответить однозначно. Представим, что мы измерили концентрацию не в миллиграммах на литр, а в граммах на литр. Это означает, что все значения в таблице, включая расстояния от средних значений, будут меньше в 1000 раз. А произведение будет меньше в миллион раз, т.е. ковариация будет равна:

$$cov(x, y) = \frac{1}{9} 0.0006663 \approx 0.000074$$

Т.е. новое значение почти равно нулю. Но ведь мы ничего не поменяли с точки зрения отношений между переменными, мы просто использовали другие единицы измерения! В этом и состоит основной недостаток ковариации — ее значение и интерпретация зависит от того, в каких единицах представлены данные. Поговорим о том, как избавиться от этого недостатка.

2.2.1 Коэффициент корреляции Пирсона

Вообще говоря, мы уже знаем ответ на этот вопрос, в разделе 2.3.8. мы обсуждали, что данные можно привести к стандартному виду, который позволяет избавиться от единиц измерения и привести обе переменные к одной шкале. Это делается с помощью операции стандартизации — путем деления каждой переменной на ее стандартное отклонение. Т.е., нам нужно просто вычислить ковариацию для стандартизированных значений, которую мы обозначим как новую статистику, $r(x, y)$:

$$x' = \frac{x - m_x}{s_x}$$
$$y' = \frac{y - m_y}{s_y}$$
$$r(x, y) = cov(x', y') = \frac{1}{n-1} \sum_{i=1}^{n} x'_i y'_i$$

С другой стороны, стандартное отклонение является константой для всех значений, т.е. ее можно вынести за знак суммирования, а значит это выражение можно записать следующим образом:

$$r(x, y) = \frac{1}{n-1} \sum_{i=1}^{n} \frac{(x_i - m_x)}{s_x} \frac{(y_i - m_y)}{s_y} = \frac{1}{s_x s_y} \frac{1}{n-1} \sum_{i=1}^{n} (x_i - m_x)(y_i - m_y) = \frac{cov(x, y)}{s_x s_y}$$

Т.е., либо мы вычисляем ковариацию для стандартизированных значений наших переменных, либо мы делаем это для исходных переменных и стандартизируем саму ковариацию — и то, и другое приведет к одинаковому результату. Полученная статистика, $r(x, y)$ носит название *корреляции* или, точнее, *коэффициента корреляции Пирсона.*

Очевидно, что коэффициент корреляции безразмерный и не зависит от разброса значений (это, как раз, и компенсируется делением на стандартное отклонение). Можно доказать математически, что:

1. Значения коэффициента корреляции всегда варьируется между -1 и $+1$.
2. Это значение будет равно $+1$ только если точки лежат строго на одной прямой, т.е. $y = ax + b$. Если угловой коэффициент этой прямой, a, отрицательный, то корреляция будет равна -1.

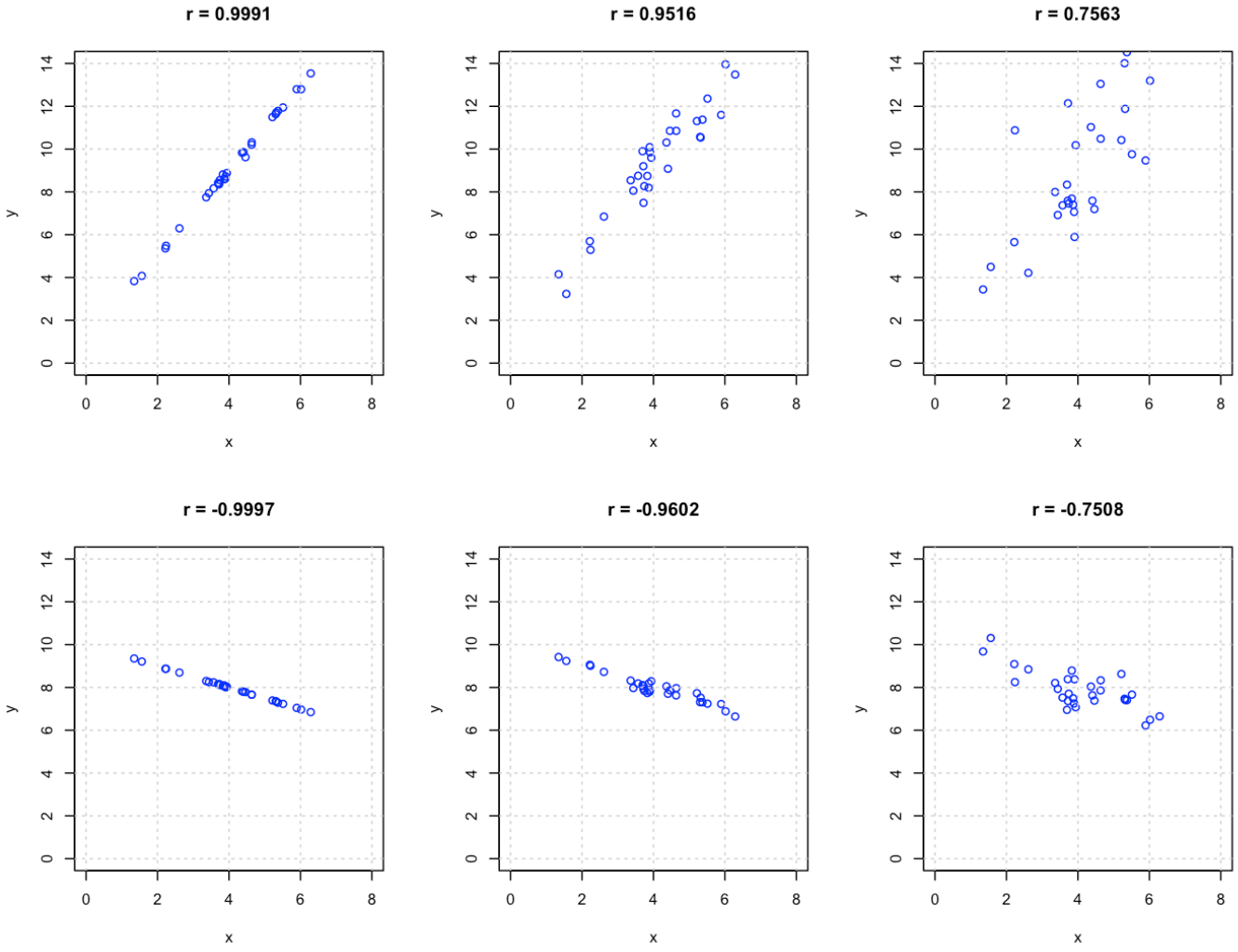


Рис. 2.4. Графики рассеяния для пар переменных с разной степенью линейной зависимости.

В остальном же корреляция имеет такой же смысл, как и ковариация — она показывает, насколько сильна линейная связь между двумя переменными (чем ближе абсолютное значение коэффициента к единице, тем сильнее), и направление этой связи (положительная или отрицательная корреляция).

Графики на рисунке 2.4 показывают диаграммы рассеяния для случаев с разной степенью линейной зависимости. Вычисленные коэффициенты корреляции показаны в заголовке графиков. Заметим, что, как мы отмечали ранее, чем ближе точки к некоторой прямой, тем выше коэффициент корреляции. При этом ориентация этой прямой на величину корреляции не влияет.

Вернемся к примерам показанным на рисунке 2.1. Так, корреляция посчитанная для данных показанных на первом графике составляет $r = 0.92$, что указывает на сильную линейную зависимость между этими двумя переменными, причем зависимость положительная. Другими словами, если в образце воды наблюдается высокая концентрация хлорид-ионов, то, с большой долей вероятности, концентрация ионов натрия также будет высока.

Если же посчитать корреляцию для данных, представленных на втором графике, то она будет гораздо меньше, $r = 0.44$. Тем не менее, мы получили положительное, отличное от нуля значение, которое также указывает на возможную связь между двумя переменными. Однако, с какого значения можно считать, что корреляция отсутствует? Понятно, что получить коэффициент равный нулю для реальных данных невозможно, т.е. нужен какой-то способ оценить, достаточно ли велико это значение.

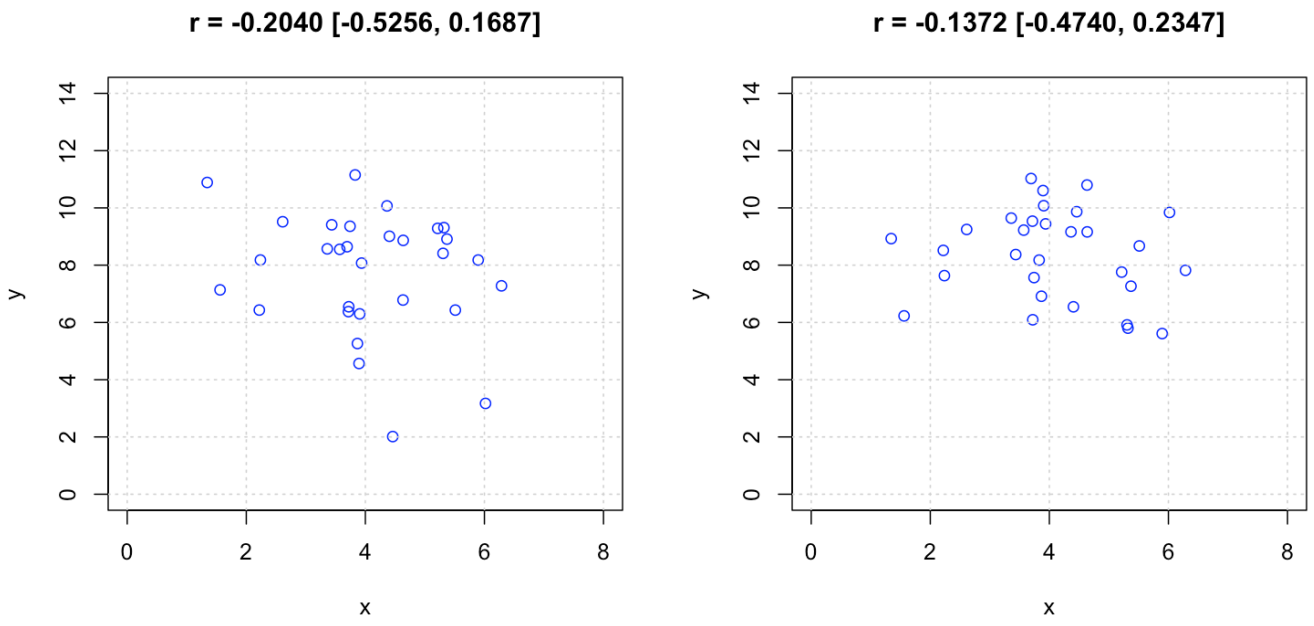


Рис. 2.5. Графики рассеяния для независимых пар переменных.

Для того, чтобы лучше проиллюстрировать проблему, давайте симулируем данные без корреляции, как

на рисунке 2.5. В этом случае и значения x, и значения y были получены с помощью обычного генератора нормально распределенных случайных чисел, никакой зависимости между ними нет. Однако, в силу случайности, мы можем видеть слабые признаки зависимости на графиках, особенно на том, что слева. Если посмотреть на вычисленные значения коэффициента корреляции (не обращайте пока внимания на числа в квадратных скобках), то можно видеть, что они отличаются от нуля. Для левого графика корреляция равна -0.2 что можно характеризовать как слабую отрицательную зависимость.

Для того, чтобы учесть случайные эффекты нужно использовать индуктивную статистику в виде теста на нулевое значение корреляции. Этот тест вернет p-значение и доверительный интервал для коэффициента — т.е. покажет, какую корреляцию мы можем ожидать, если повторим эксперимент. Именно этот интервал и показан на графиках в квадратных скобках. Как можно видеть, в обоих случаях ожидаемая корреляция может быть как отрицательной так и положительной. Другими словами, у нас недостаточно данных, чтобы надежно засвидетельствовать наличие зависимости между двумя переменными в этом случае.

Возвращаясь к примеру с ионами, вычисляя 95% доверительный интервал для первого примера, мы получим $[0.70, 0.98]$, т.е. даже если взять наименьшее возможное число в этом интервале, 0.7, коэффициент корреляции все равно довольно высокий. Такой же интервал, вычисленный для второго примера, даст $[-0.26, 0.84]$. В этом случае неопределенность очень высока и значение корреляции, которое мы можем получить, повторив наш эксперимент, может варьироваться от отрицательного -0.26 до положительного 0.84. Это говорит о том, что в данном случае у нас нет весомых оснований утверждать, что между этими двумя переменными есть положительная связь. Вполне возможно, что та слабая связь, которую мы наблюдаем в данном случае — лишь элемент случайности.

Как и с другими статистиками, неопределенность значения коэффициента корреляции зависит от нескольких факторов, в первую очередь, от размера выборки и величины самого коэффициента. Поэтому, если корреляция слабая, нужна выборка большого размера чтобы констатировать статистически значимое значение коэффициента.

2.2.2 Корреляционная матрица и ее тепловая карта

При работе с многомерными данными часто наблюдается эффект мультиколлинеарности — когда сразу несколько переменных (а иногда и десятки, и даже сотни) коррелируют между собой. При этом определить этот эффект и визуализировать его довольно трудно — как мы уже обсуждали ранее, графики рассеяния не очень подходят для многомерных данных. Наиболее эффективный способ решения этой проблемы мы рассмотрим в следующем разделе, а пока представим альтернативный подход, основанный на вычислении корреляционной матрицы.

Идея состоит в следующем. Предположим, у нас есть данные в виде матрицы $\mathbf{X}$, столбцы которой и представляют переменные. Скажем, для данных о содержании ионов в воде, эта матрица будет иметь 10 строк и 3 столбца. Для того, чтобы посчитать коэффициент корреляции между переменными, можно взять значения из соответствующих столбцов, стандартизовать их (т.е. для каждого столбца вычесть среднее и

поделить на стандартное отклонение), а затем умножить стандартизованные значения друг на друга. Но как сделать это:

1. Для всех пар столбцов в матрице данных?
2. Быстро и лаконично (с точки зрения, например, программного кода)?

Предположим, что нам удалось получить стандартизованные данные в виде матрицы **Z**, значения в каждом столбце этой матрицы — это стандартизованные значения из соответствующего столбца матрицы **X**:

$$z_{ij} = \frac{x_{ij} - m_{\mathbf{x}_j}}{s_{\mathbf{x}_\mathrm{j}}}$$

Тогда мы можем вычислить матрицу **R** с помощью простого скалярного произведения матрицы **Z** на саму себя:

$$\mathbf{R} = \frac{1}{n-1}\mathbf{Z}^\mathrm{T}\mathbf{Z}$$

Очевидно, что матрица **R** будет квадратной, число строк и столбцов в этой матрице будет равно числу столбцов в матрице **Z** — т.е., количеству исходных переменных. Каждый элемент матрицы **R** будет представлять собой корреляцию между соответствующими столбцами матрицы **X**, т.е., r_{23} будет содержать коэффициент корреляции между вторым и третьим столбцом матрицы **X**.

Таблица ниже показывает содержимое корреляционной матрицы, рассчитанной для данных о содержании ионов в воде. Как можно видеть, матрица является симметричной относительно главной диагонали, так как $r(x, y) = r(y, x)$. Значения по диагонали ожидаемо равны единице.

	Cl-	Na+	K+
Cl-	1.000	0.923	0.442
Na+	0.923	1.000	0.361
K+	0.442	0.361	1.000

Быстрый взгляд на таблицу позволяет оценить все необходимые значения. Так, видно, что содержимое ионов калия в воде очень слабо связано с двумя другими переменными, тогда как (мы определили это ранее) имеется сильная корреляция между содержимым ионов натрия и хлорид-ионов.

Однако, если бы число переменных в нашей таблице было не 3, а хотя бы 30, то проанализировать ее было бы довольно затруднительно. В этом случае нам поможет использование тепловых карт (англ. *heat maps*) — когда матрица показывается в виде прямоугольника с цветными сегментами. Число сегментов равно числу элементов матрицы, а их цвет зависит от значения элемента. Обычно используют различные градиенты, чтобы подчеркнуть разницу.

Рисунок 2.6 показывает представление корреляционной матрицы, полученной нами выше, с помощью такой карты. В качестве градиента используется цвета на основе красного для положительных значений, синего — для отрицательных, и желтого — для значений около нуля. Так, например, мы можем видеть, что корреляция между Cl- и K+ светло-оранжевая, т.е. находится в положительной части, но близко к желтому цвету. Одного взгляда на эту тепловую карту достаточно, чтобы увидеть, что Cl- и Na+ образуют некий кластер взаимозависимых переменных, тогда как K+ находится не в этом кластере.

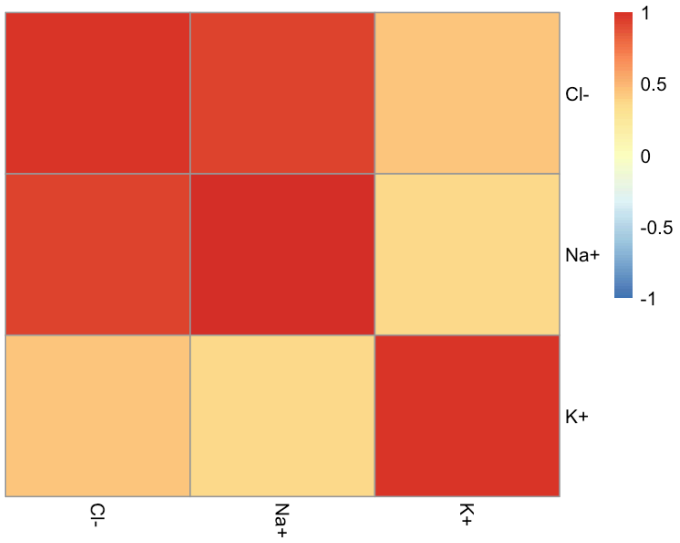


Рис. 2.6. Тепловая карта корреляционной матрицы полученной для данных описывающих концентрацию ионов в 10 пробах воды.

Рисунок 2.7 показывает тепловую карту корреляционной матрицы, полученной для набора данных Люди, который мы упоминали выше. В данном случае мы взяли все 12 переменных, т.е., корреляционная матрица имеет размерность 12×12. Кроме этого, мы сгруппировали переменные так, чтобы те, что имеют высокую взаимную корреляцию, оказались рядом.

Если посмотреть внимательно на тепловую карту, то можно заметить несколько интересных вещей. Во-первых, довольно большой блок переменных имеет высокую попарную корреляцию, а именно, *Swim*, *Weight*, *Height*, *Shoesize*, *Hairlength*, *Sex*. Туда же, с небольшой натяжкой, можно отнести и потребление пива, *Beer*. Другими словами, можно предположить наличие некоторого скрытого фактора, который коррелирует со всеми этими переменными. Как мы уже определили ранее, это, по сути, уровень физического развития человека, а точнее — тип его телосложения. Чем выше человек, тем больше он весит, имеет больший размер обуви, и, скорее всего, такой человек сильнее физически.

С большой долей вероятности он имеет мужской пол и, как следствие, короткие волосы. Дело в том, что эти две переменные (пол и длина волос) закодированы в наборе данных так, что они равны +1 для женщин и длинных волос и -1 для мужчин и коротких волос. Именно поэтому эти две переменные имеют отрицательную корреляцию с остальными переменными в данном блоке, это можно видеть по цвету.

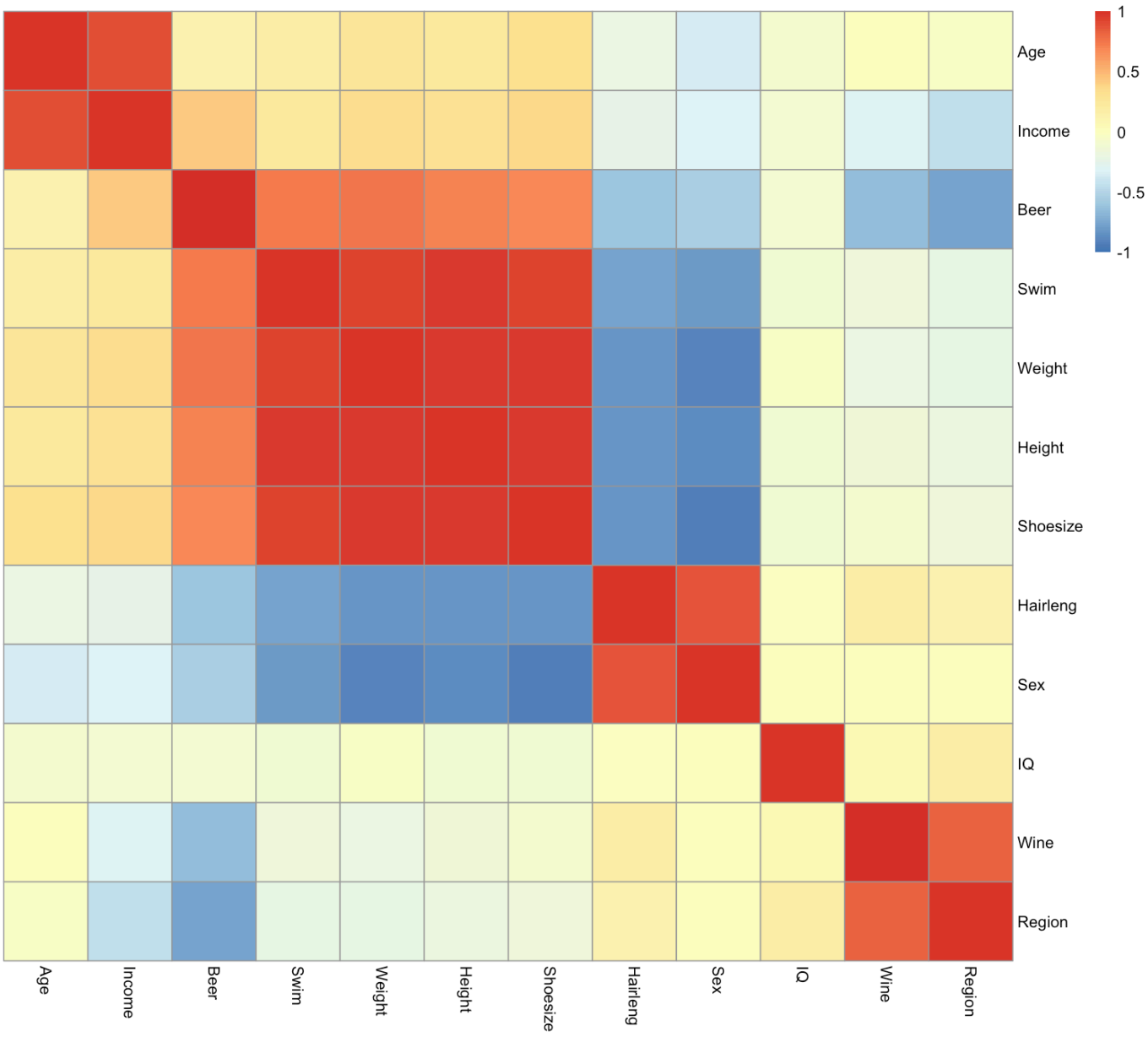


Рис. 2.7. Тепловая карта корреляционной матрицы для набора данных *Люди.*

Второй блок сильно связанных между собой переменных — это вино (*Wine*) и регион (*Region*). Опять же, мы знаем, что употребление вина в южных странах (например, средиземноморских) выше, по сравнению с северными странами. Т.е., по сути, эти две переменные описывают один и тот же фактор — регион.

Возраст (*Age*) и доход (*Income*) образуют третий блок и, наконец, коэффициент интеллекта (*IQ*) стоит обособлено и не связан ни с одной из других 11 переменных.

Помимо этого, мы можем заметить, что потребление пива и доход вносят свой вклад в два блока — так, потребление пива коррелирует как с переменными из первого блока (связанного с телосложением), так и с "региональным" блоком. Доход же, судя по всему, коррелирует и с возрастом, и с регионом.

Таким образом, тепловая карта показывает нам, что 12 переменных из нашего набора данных, по сути, описывают 4 скрытых характеристики. Или, другими словами, что разброс значений этих 12 переменных в первую очередь обусловлен тем, что люди в нашей выборке имеют разный пол, взяты из разных регионов, имеют разный возраст и разный IQ. Ну и конечно, случайная вариация тоже присутствует, как вы увидите чуть позже, она составляет примерно 6%, остальные 94% дисперсии связаны именно с этими четырьмя скрытыми факторами.

Такие скрытые в данных факторы мы будем называть *латентными переменными*, и в следующем разделе поговорим подробнее, как их найти и использовать.

2.3 Латентные переменные

Как мы уже обсуждали ранее в этой главе, очень часто переменные зависят от некоторого скрытого фактора. Многие характеристики людей, например, вес, умение плавать, делать гимнастические упражнения, объем потребляемой жидкости (не обязательно пива) и еды, зависят от их телосложения и уровня физического развития. Т.е., высокий человек с атлетическим сложением будет по всем этим параметрам отличаться от человека небольшого роста и несклонного заниматься спортом.

Если же попытаться проанализировать эти характеристики математически, то мы увидим, что между ними имеется сильная положительная корреляция, как мы видели в последнем примере предыдущего раздела. В этом случае можно говорить о наличии некоторого скрытого фактора, который и объясняет эти корреляции. Такой фактор мы будем называть *латентной переменной* (англ. *latent variable*).

Надо сказать, что латентная переменная необязательно связана именно с корреляцией, она может быть также связана и с другими структурными зависимостями в данных, например, группами. Таким образом, способы выявления латентных переменных очень сильно зависят от постановки задачи и преследуемых целей. Рассмотрим два простых примера, показанных на рисунке 2.8.

Верхняя часть рисунка содержит диаграммы рассеяния для двух разных наборов данных. И в том, и в другом случае мы имеем дело с 20 измерениями, которые получены для двух абстрактных переменных,

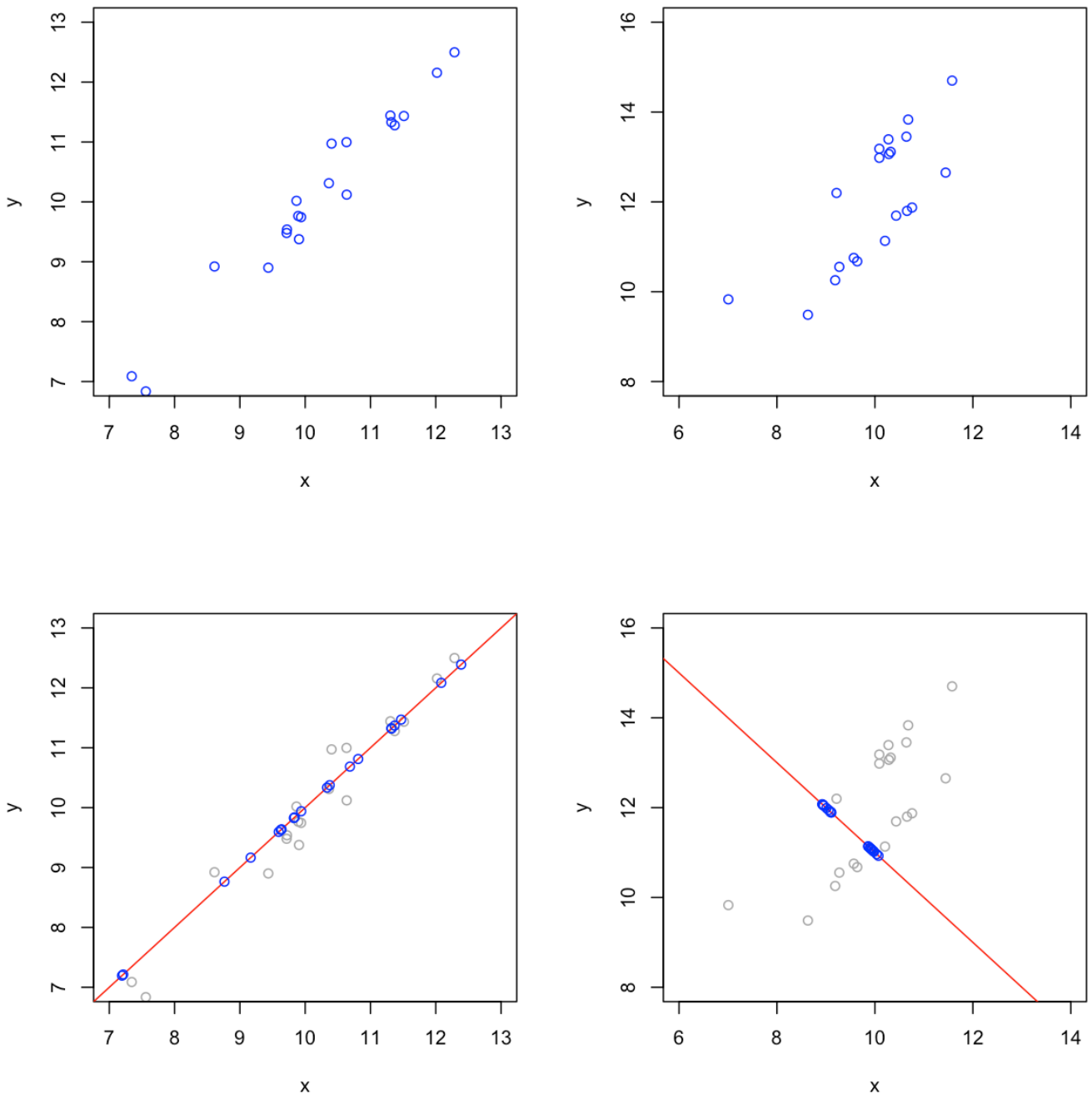


Рис. 2.8. Два разных подхода к определению латентных переменных.

x и y. Предположим, что перед нами стоит задача снизить размерность этих данных с двух до одной. Т.е. чтобы вместо 40 чисел наши измерения можно было бы описать 20. Как этого можно добиться?

Начнем с первого случая, он показан на левой части рисунка. Очевидно, что в данном случае между переменными имеется очень сильная линейная зависимость. Следовательно, будет логичным ориентировать нашу латентную переменную вдоль этой зависимости, т.е. в направлении максимального разброса точек на графике. Для этого нужно найти единичный вектор, соответствующий этому направлению, и спроецировать все точки на этот вектор, как мы рассматривали в разделе 2.2. Результат этого действия показан на левом нижнем графике рисунка 2.8.

Синие точки теперь лежат на новой переменной, заданной нашим вектором, она показана в виде красной линии. Так как переменная одна, то каждое измерение имеет только одну координату, т.е. с задачей уменьшения размерности мы справились. Про исходную двумерную систему координат можно теперь "забыть" и работать в одномерной системе координат, соответствующей красной линии.

Можно заметить, что то, как мы расположили нашу латентную переменную, позволило хорошо учесть взаимное расположение точек друг относительно друга (какие точки располагаются близко, а какие — далеко). Кроме этого, мы учли почти всю дисперсию — синие и серые точки на новом графике расположены очень близко друг к другу и взаимное расстояние между синими точками почти такое же (немного меньше на самом деле), как и расстояние между исходными (серыми) точками. Т.е., мы уменьшили размерность данных в два раза, почти ничего не потеряв.

Рассмотрим теперь второй пример, показанный на графиках справа. Очевидно, что помимо линейной зависимости в данных имеется и другая структура — наличие двух хорошо различимых групп точек. Предположим, что нашей целью в этом случае как раз является сохранение этой структуры при уменьшении размерности. Т.е., в новой одномерной системе координат точки из разных групп должны хорошо различаться.

В этом случае ориентация латентной переменной будет не вдоль максимального разброса точек, как в предыдущем примере, а вдоль направления, которое позволяет максимально различить эти две группы. Такая переменная и проекции исходных точек на нее показаны на графике ниже. Очевидно, что в данном случае синие точки очень сильно отличаются от исходных, серых. Т.е. проецируя точки на новую переменную мы теряем большую часть вариации этих точек. Но дело в том, что вариация, которую мы теряем — это вариация точек внутри отдельных групп. А вариация, которую сохраняем, как раз связана с разницей между группами — это именно та полезная структура данных, в которой мы заинтересованы. И выбор латентной переменной позволяет нам ее сохранить — разница между группами для спроецированных точек довольно хорошо различима.

Собственно в этом и заключается концепция использования латентных переменных — во-первых, сократить размерность данных, а во-вторых, избавиться от ненужной (нерелевантной) вариации данных и сохранить ее полезную часть. Т.е., путем проецирования точек на латентные переменные мы разделяем

данные на две части: структуру (полезную информацию) и шум (нерелевантную для нашей задачи информацию).

Зачем нужно снижать размерность? Для этого имеется много причин. Во-первых (это не самое главное, но все же), это даст нам возможность визуализировать данные с помощью удобных графиков рассеяния. Если исходное число переменных, скажем, 12, то число таких графиков для всех возможных пар переменных равно 66. Если же число переменных можно снизить до, скажем, 4, не потеряв при этом взаимосвязей между отдельными измерениями, то максимальное число графиков рассеяния, которые нужно изучить чтобы эту взаимосвязь обнаружить будет равно лишь 6.

Во-вторых, уменьшая число переменных мы избавляемся от проклятья размерности. Предположим, мы хотим построить регрессионную модель, чтобы прогнозировать значение одной переменной по измеренными значениям других (к примеру, градуировку для спектральных данных, используя закон Ламберта-Бера). Для того, чтобы провести линию в двумерном пространстве (т.е., у нас есть две переменных) — нужно две точки. Такая модель в трехмерном пространстве будет представлять собой плоскость, а для ее построения нужно уже три точки. Если вы работаете со спектральными данными, то число переменных будет равно сотням а то и тысячам. Т.е., для построения модели вам понадобится, как минимум, столько же измерений.

С другой стороны, в спектральных данных очень высока взаимная корреляция между переменными — такой эффект называют *мультиколлинеарностью*. Это позволяет снизить исходную размерность с сотен переменных до нескольких латентных, сохранив до 95% исходной дисперсии.

Настало время перейти к конкретике и познакомить вас с одним из методов, который использует концепцию латентных переменных — методом главных компонент.

2.4 Нахождение главных компонент

Метод главных компонент (англ. *Principal Component Analysis*, *PCA*) — это довольно простой метод представления данных в виде линейной комбинации латентных переменных. Его особенностями является то, как выбираются эти переменные, в частности:

1. Латентные переменные ориентированы вдоль направления максимального разброса данных.
2. Все латентные переменные взаимно ортогональны.

Такие переменные называют *главными компонентами* (англ. *principal components*), от чего и произошло, собственно, название самого метода.

Для начала разберемся, как определяются главные компоненты, на простом двумерном примере. Будем использовать данные о содержании ионов в воде, уже хорошо нам знакомые. Исходные данные показаны на рисунке 2.9a (график слева).

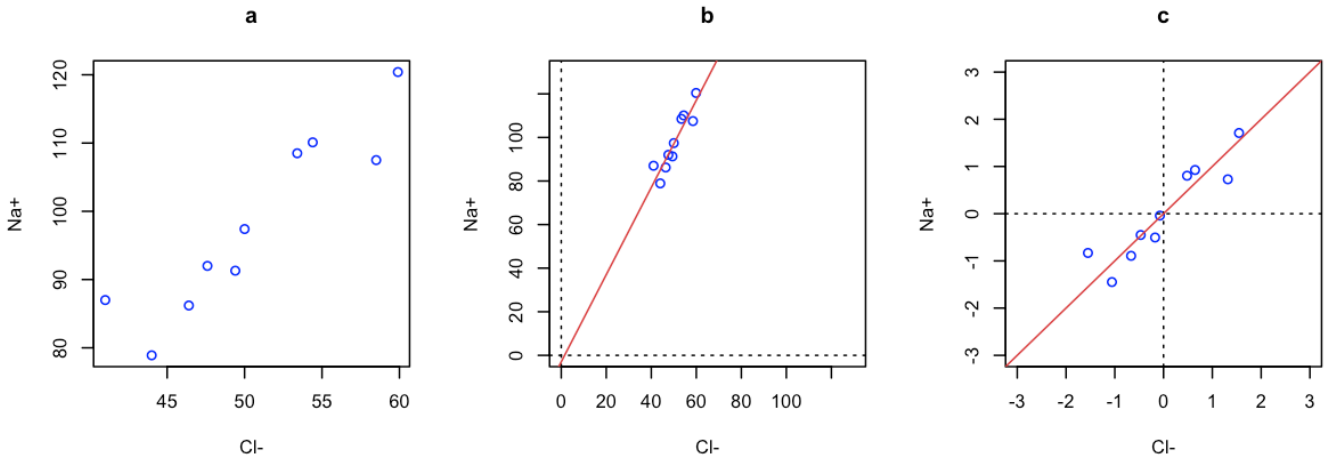


Рис. 2.9. Определение направления максимального разброса данных (левый график: исходные данные, посередине: те же данные, но в новом масштабе, справа: стандартизированные данные).

В главе 2 мы обсуждали, что направления в пространстве переменных задаются векторами и все векторы проходят (точнее начинаются) в начале координат. Рисунок 2.9b (график посередине) показывает те же данные, что изображены на левом графике, но в масштабе, где также представлено начало координат, плюс, оси имеют одинаковый разброс (от 0 до 120 мг/л). Красная линия на этом графике показывает направление вдоль максимального разброса точек. К сожалению, это направление не может быть задано вектором, так как оно не проходит через начало координат (хотя и расположено довольно близко к нему).

Решается эта проблема просто — центрированием, а очень часто и стандартизацией данных, которую мы уже обсуждали ранее. В этом случае центр облака точек, представляющих наши данные, будет находиться в начале координат, а это нам и нужно.

Рисунок 2.9с (график справа) показывает уже стандартизованные данные. Как мы видим, теперь красная линия, ориентированная вдоль максимального разброса точек, проходит через начало координат, а значит ее направление можно задать единичным вектором. В хемометрике операцию стандартизации обычно называют *автошкалированием* (англ. *autoscaling*).

Поговорим теперь о том, как найти направление максимального разброса данных. Идея очень проста — это такое направление, для которого дисперсия (т.е. среднеквадратичное расстояние) точек, спроецированных на него, будет максимально возможным. Такую дисперсию называют *объясненной дисперсией* (англ. *explained variance*) — это та часть общей дисперсии данных, которую это направление (точнее связанная с ним компонента) объясняет.

Чтобы проиллюстрировать эту концепцию, посмотрим на четыре графика, показанных на рисунке 2.10. Визуально довольно трудно определить, какой из графиков показывает наиболее оптимальное направление будущей главной компоненты. Но если посчитать дисперсию точек, спроецированных на нее (синие точки на графиках), то легко видеть, что наилучшим образом это направление задано на последнем графике. В этом случае объясненная дисперсия равна 0.2137 и, как мы можем видеть, это максимальное значение из четырех представленных на рисунке.

> *Идею объясненной дисперсии можно показать на простом примере, который мы использовали для объяснения скалярного произведения матриц во второй главе. Для этого мы его немного изменим. Предположим, что, как и в оригинальном примере, вы хотите смешать яйца с рисом и запечь их в духовке. Кроме этих двух ингредиентов вы также добавите соль, перец и, может быть, капельку молока для вкуса. Все компоненты, как и само блюдо, можно представить в пространстве, связанном с основными питательными веществами — белками, жирами и углеводами (БЖУ). Яйца и рис в этом случае будут главными компонентами. Это векторы в пространстве БЖУ. Само блюдо будет является точкой в этом пространстве, ее координаты можно определить с помощью линейной комбинации этих компонент. При этом, чтобы определить точное содержание питательных веществ в блюде, недостаточно учесть только две главных компоненты, нужно еще посчитать и те небольшие добавки (соль, перец и молоко). Но, так как их питательная ценность мала по сравнению с основными ингредиентами, то их вкладом можно пренебречь. Т.е. примерное содержание БЖУ блюда, определяемое содержанием яиц и риса, и будет являться аналогом объясненной дисперсии.*

Посчитанные значения для объясненной дисперсии приведены в заголовке графиков. В скобках указаны относительные значения, посчитанные как процент от дисперсии исходных точек. Другими словами, в

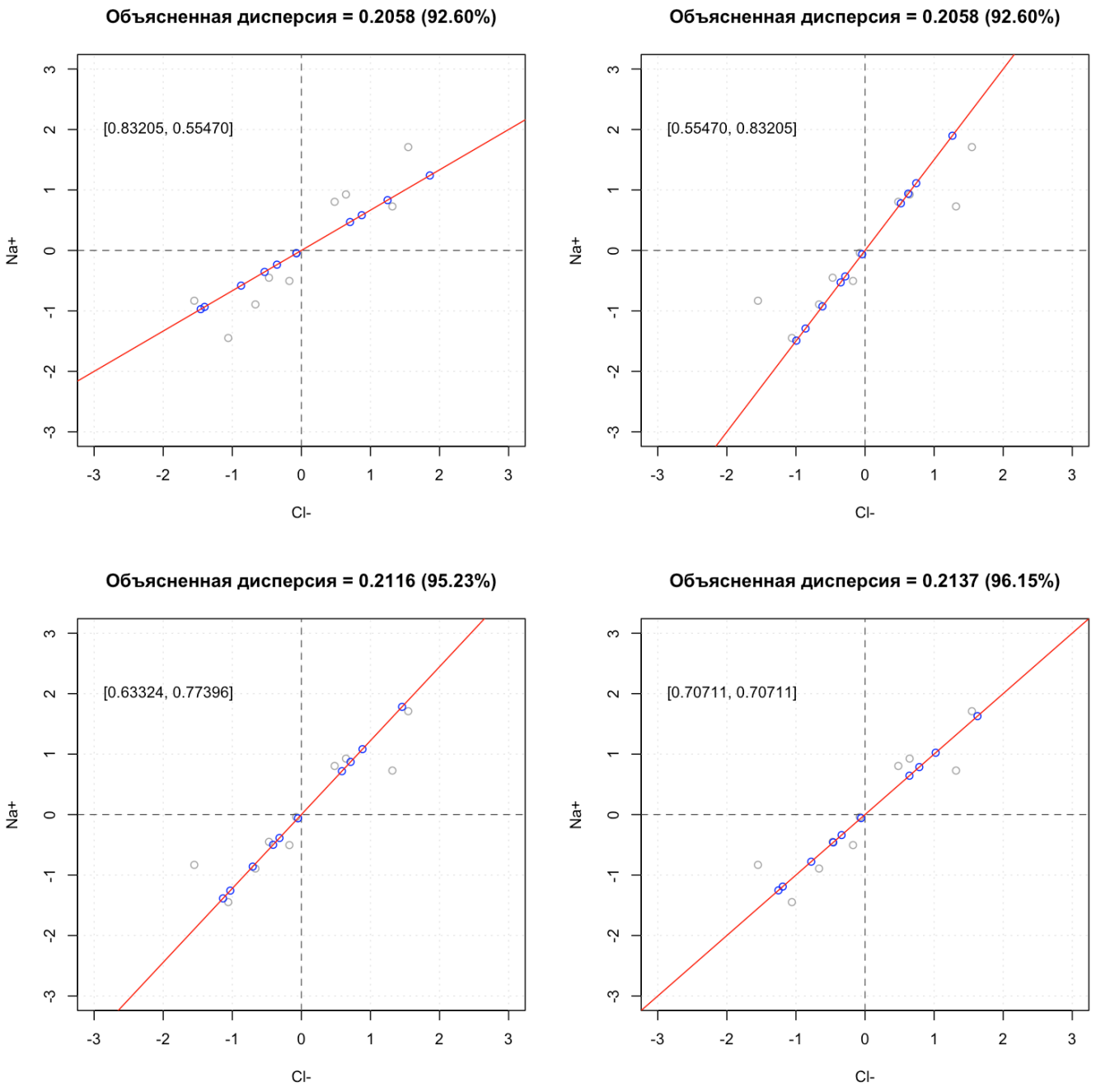


Рис. 2.10. Несколько вариантов ориентации главной компоненты и дисперсия, которую она объясняет.

последнем случае, при проекции точек на главную компоненту теряется чуть меньше 4% от дисперсии исходных данных. Так как данные центрированы, то дисперсия считается очень просто — как среднее значение квадратов расстояний от точек до начала координат. Т.е., нужно просто возвести все элементы матрицы, которая задает координаты точек, в квадрат, просуммировать их и разделить на число точек минус один.

Числа в квадратных скобках, показанные на графиках слева вверху, это координаты соответствующих векторов. Векторы, задающие направления главных компонент называются *нагрузками* (англ. *loadings*), и, как правило, обозначаются как $\mathbf{p}$. Это всегда единичные векторы (чтобы проще было вычислять проекции с помощью скалярного произведения).

Если компонент больше одной, то векторы объединяют вместе, так, чтобы они образовывали столбцы матрицы нагрузок, которую обычно обозначают $\mathbf{P}$.

2.4.1 Алгоритм NIPALS

Есть несколько способов расчета координат векторов нагрузок. Алгоритм *NIPALS* (Nonlinear Iterative Partial Least Squares) позволяет определить эти координаты с помощью нескольких итераций. Рассмотрим его работу сначала для одной главной компоненты, а чуть позже покажем, как посчитать и остальные. Основной алгоритм следующий:

1. Выбираем столбец (переменную) матрицы с наибольшей дисперсией — это будет нашим нулевым приближением для вектора нагрузок. Если мы работаем с двумя переменными и выбран первый столбец, то вектор будет иметь координаты $[1, 0]^T$ (т.е. он совпадает с осью X), а если выбран второй столбец, то $[0, 1]^T$ (т.е. он совпадает с осью Y). Координаты проекции точек на этот вектор в этом случае будут равны значениям выбранной переменной, обозначим вектор с этими начальными значениями как $\mathbf{t}$.

2. Считаем ковариацию между значениями вектора с проекциями, $\mathbf{t}$, и столбцами исходных данных. В результате получится столько значений, сколько у нас столбцов в таблице (т.е. по одному значению для каждой исходной переменной).

3. Нормируем эти значения так, чтобы длина вектора равнялась единице. Полученные в итоге значения и будут новыми координатами нашего вектора.

4. Проецируем все точки на направление, заданное новыми координатами вектора $\mathbf{p}$. Как мы уже обсуждали в главе 2, для того, чтобы спроецировать точки на направление, заданное единичным вектором, нужно просто умножить матрицу с данными $\mathbf{X}$ на столбец с координатами этого вектора, $\mathbf{p}$: $\mathbf{t} = \mathbf{X}\mathbf{p}$.

Ниже показан код на R который считает координаты вектора для главной компоненты для любой матрицы **X**. Предполагается, что эта матрица уже отцентрирована, т.е. среднее значение для каждого столбца равно нулю:

```
# 1. Считаем стандартное отклонение для столбцов и
#    находим номер столбца с наибольшим значением
sd <- apply(X, 2, sd)
ind <- which.max(sd)

# 2. Выбираем начальные значения для вектора проекций, t
t <- X[, ind, drop = FALSE]

# 3. Считаем ковариацию между значениями исходных переменных и вектором
#    с проекциями, сохраняя полученные значения в вектор нагрузок, p
p <- cov(X, t)

# 4. Нормируем вектор нагрузок
p <- p / sqrt(sum(p^2))

# 5. Проецируем точки на этот вектор
t <- X %*% p
```

Этот код делает только одну итерацию. Чтобы получить правильное значение, таких итераций (шаги с 3 по 5) нужно несколько, пока не будет достигнут некий критерий сходимости — например, разница между координатами вектора на предыдущем шаге и на нынешнем не будет меньше некоторого порогового значения.

Рисунок 2.11 показывает первые четыре итерации такого алгоритма, примененного для стандартизированных данных (содержание ионов в воде), ранее используемых нами в этом разделе.

Обратите внимание, что разница между координатами вектора и объясненной дисперсией для итераций 3 и 4 практически неразличима. Т.е. продолжать дальше уже не имеет смысла, нужно сохранить полученные результаты и приступить к определению следующей компоненты.

2.4.2 Нахождение младших компонент

Самая первая главная компонента обычно считается старшей, так как она всегда объясняет наибольшую часть дисперсии данных. Для того, чтобы найти вторую и (если размерность исходных данных больше двух) следующие компоненты нужно вспомнить, что все компоненты должны быть взаимно ортогональны.

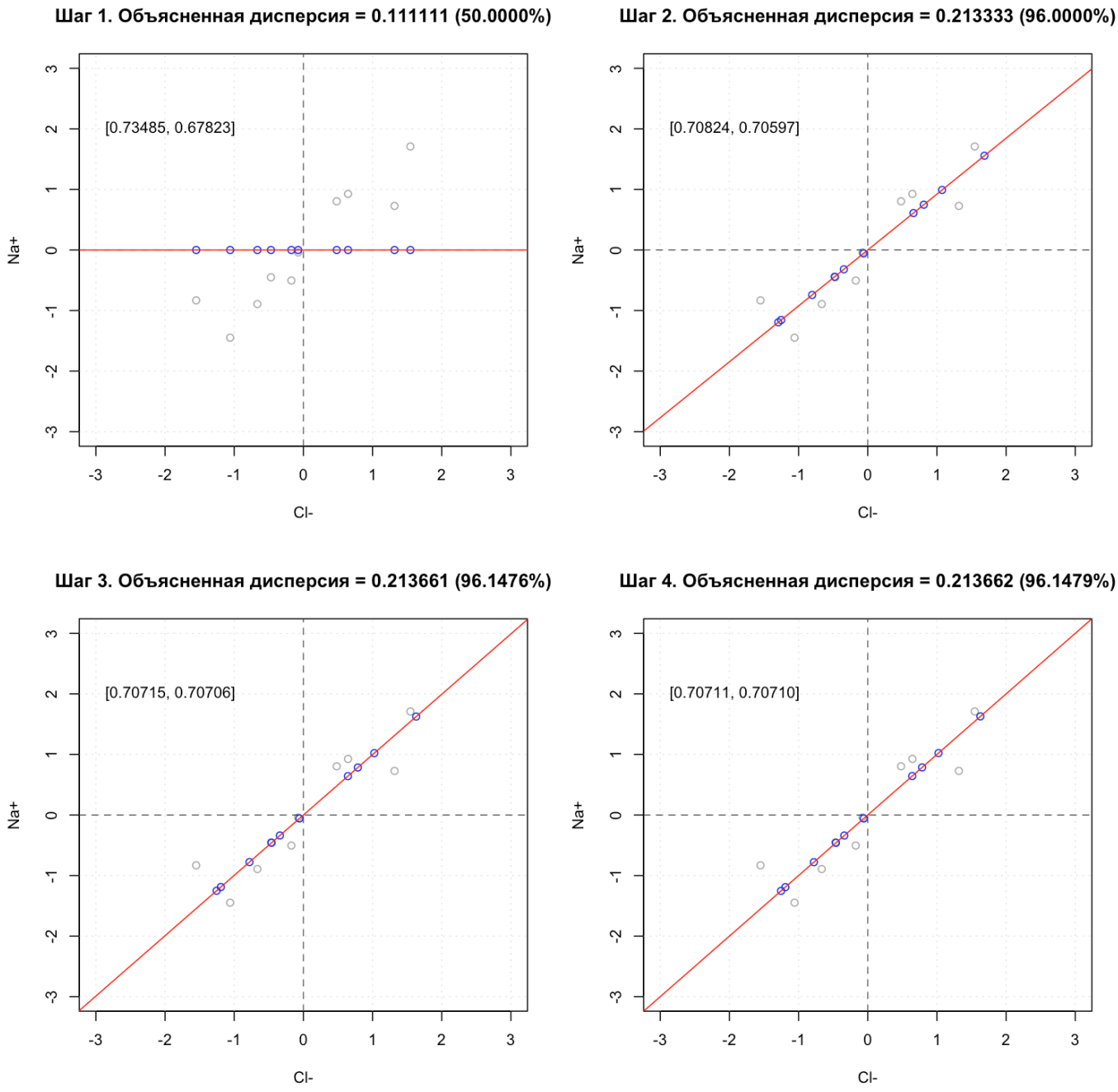


Рис. 2.11. Первые четыре итерации алгоритма NIPALS для стандартизированных данных по содержанию ионов в воде.

Это значит, что дисперсия, которую объясняет первая компонента, никак не должна быть связана с направлением второй компоненты. Чтобы этого добиться ее просто нужно убрать из исходных данных.

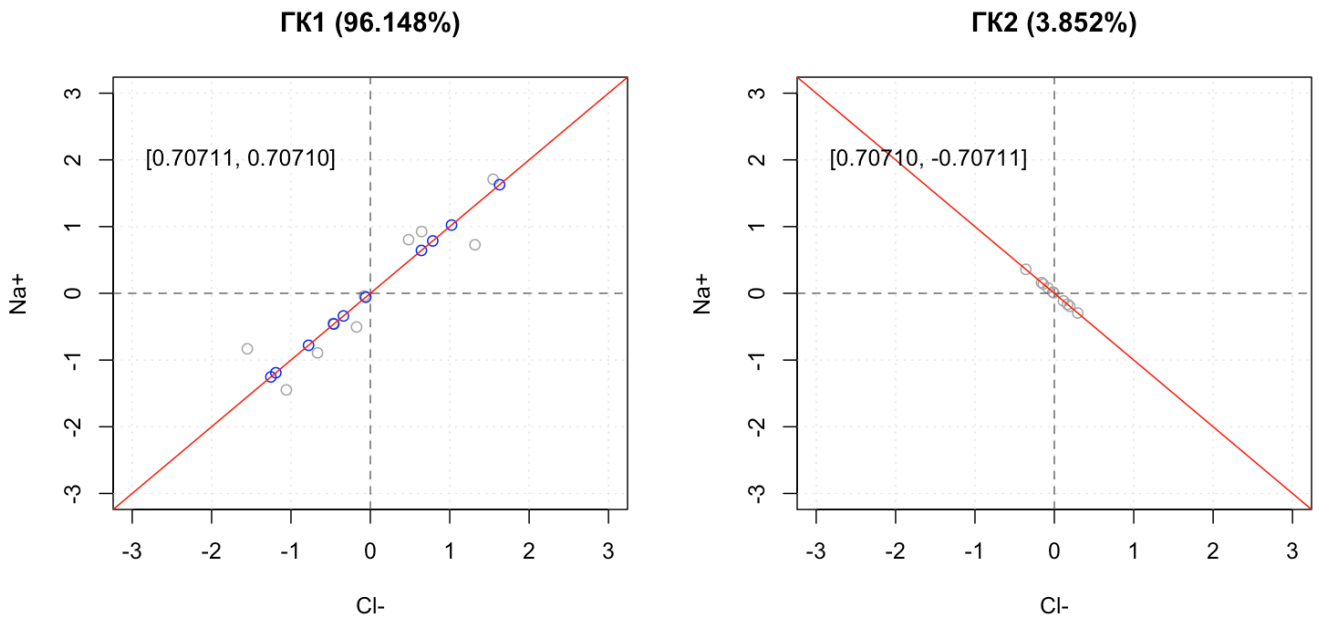


Рис. 2.12. Нахождение главных компонент для данных по содержанию ионов в воде. Слева — нахождение первой компоненты. Справа — нахождение второй главной компоненты.

Рисунок 2.12 показывает как это выглядит. Сначала мы определяем координаты вектора нагрузок для первой компоненты (график слева) и проецируем точки на нее. Координаты этих проекций можно посчитать как для компоненты, так и для исходной системы координат. Для этого нужно сделать следующую операцию:

$$\mathbf{X}' = \mathbf{X}\mathbf{P}\mathbf{P}^{\mathrm{T}} = \mathbf{T}\mathbf{P}^{\mathrm{T}}$$

Матрица $\mathbf{X}'$ как раз и содержит координаты проекций (синих точек) в исходной системе координат, т.е. ее размер (число строк и столбцов) точно такой же как и размер нашей исходной матрицы данных, $\mathbf{X}$. Этот этап показан на двух рисунках ранее и также на левом графике рисунка 2.12.

Теперь чтобы вычесть то, что мы уже объяснили с помощью первой главной компоненты, нужно просто вычислить разницу между $\mathbf{X}'$ и $\mathbf{X}$:

$$\mathbf{E} = \mathbf{X} - \mathbf{X}'$$

После этого мы снова запускаем алгоритм NIPALS, но уже для матрицы $\mathbf{E}$. Этот этап показан на правом графике рисунка 2.12. Как мы можем видеть, серые точки сильно отличаются от тех, что показаны на первом

графике, так как мы вычли из их исходных координат ту часть, которую объясняет первая компонента. Другими словами, этот график показывает координаты точек не из **X** а из **E**. И, соответственно, вторую компоненту, которая расположена вдоль максимального разброса этих точек.

Если нужно посчитать больше компонент, то мы просто вычитаем из матрицы **E** проекции точек на вторую компоненту и продолжаем до тех пор, пока все компоненты не будут найдены.

Очевидно, что для нашего случая мы не можем определить число компонент больше двух, так как исходное пространство переменных двумерное. В случае реально многомерных данных максимальное число компонент, которое можно посчитать равно наименьшему из двух чисел — числу индивидуальных измерений минус один, либо числу исходных переменных.

После того, как все компоненты будут найдены, их объединяют в общую матрицу нагрузок, **P**, так, чтобы каждый вектор занимал отдельный столбец в этой матрице. Т.е. ее размерность будет равна $J \times A$, где J — число исходных переменных, а A — число вычисляемых главных компонент. Соответственно, МГК разложение можно записать в виде следующего соотношения:

$$\mathbf{X} = \mathbf{T}\mathbf{P}^{\mathrm{T}} + \mathbf{E}$$

Здесь **X** — это исходные данные после центрирования (и шкалирования, если оно было необходимо); **P** — матрица нагрузок, которая содержит единичные векторы, задающие направления главных компонент; **T** — матрица проекций исходных данных на главные компоненты, т.е., координаты синих точек на наших графиках в пространстве ГК (положение этих точек на красных линиях).

Произведение $\mathbf{T}\mathbf{P}^{\mathrm{T}}$ позволяет определить координаты проекций (синих точек), но уже в исходном пространстве переменных, соответственно, матрица **E** содержит в себе разность координат между синими и серыми точками. Эту матрицу называют матрицей *остатков* (англ. *residuals*) и значения в ней становятся меньше с увеличением числа главных компонент.

Блок кода ниже содержит функцию, которая для заданной матрицы данных вычисляет матрицы счетов и нагрузок для нужного числа компонент. Попробуйте реализовать ее и использовать для простых (например, двумерных) данных.

```
nipals <- function(X, A = 1) {

   # центрируем и шкалируем данные
   X <- scale(X, center = TRUE, scale = TRUE)

   # подготавливаем пустые матрицы для счетов и нагрузок
   scores <- matrix(0, nrow = nrow(X), ncol = A)
```

```
    loadings <- matrix(0, nrow = ncol(X), ncol = A)

    # цикл для вычисления каждой компоненты
    for (a in 1:A) {

       # находим столбец с наибольшей дисперсией
       col.ind <- which.max(apply(X, 2, sd))

       # выбираем значения этого столбца как первое приближение вектора проекций
       t <- X[, col.ind, drop = FALSE]

       # цикл для вычисления компоненты (30 итераций)
       for (i in 1:30) {
          p <- cov(X, t)
          p <- p / sqrt(sum(p^2))
          t <- X %*% p
       }

       # вычитаем объясненную дисперсию из матрицы данных
       X <- X - tcrossprod(t, p)

       # сохраняем полученные значения t и p в виде столбцов матриц
       scores[, a] <- t
       loadings[, a] <- p
    }

    # возвращаем матрицы нагрузок и счетов
    return(list(scores = scores, loadings = loadings))
}
```

Поговорим теперь о том, что нужно делать со всеми этими матрицами.

2.5 Анализ графиков счетов и нагрузок

Итак, метод главных компонент позволяет найти направления, вдоль которого данные имеют наибольший разброс и построить новую систему координат с осями, расположенными вдоль этих направлений. Эти направления задаются единичными векторами нагрузок и носят названия *главных компонент*.

Главные компоненты представляют собой пространство латентных переменных, расположенное внутри

пространства исходных переменных. За счет того, что они ориентированы так, чтобы “захватить” как можно большую часть дисперсии данных, несколько главных компонент обычно достаточно, чтобы описать большую часть вариации исходных данных. Эффективность такого “захвата” зависит от того, насколько сильно скоррелированы исходные переменные. Например, для спектральных данных, очень часто можно снизить размерность с сотен или даже тысяч исходных переменных (длин волн) до нескольких главных компонент и при этом объяснить более 90% исходной вариации.

Если спроецировать точки с исходными данными на пространство главных компонент, то мы получим новый набор точек, лежащих внутри ГК пространства. Если, например, имеется две компоненты, то ГК-пространство будет представлять собой плоскость и точки будут лежать на этой плоскости. Координаты точек внутри такой плоскости называются *счетами* (англ. *scores*).

Рисунок 2.13 иллюстрирует МГК разложение трехмерных данных с помощью двух главных компонент. Для этого примера мы взяли данные из набора *Люди*, выбрав три переменных — рост, коэффициент интеллекта и вес.

График “a” (слева вверху) показывает исходные данные в пространстве переменных после того, как они были центрированы и шкалированы. Как можно видеть, данные хорошо описываются двумя главными компонентами, направление которых показано в виде красных стрелок. Эти стрелки — единичные векторы нагрузок, координаты которых вместе будут составлять матрицу нагрузок **P**. Так как мы имеем три исходных переменных и две главных компоненты, то эта матрица будет иметь три строки и два столбца.

График “b” (справа вверху) показывает пространство главных компонент “натянутое” на векторы нагрузок. Так как компонент две, то пространство будет представлять собой плоскость, показанную на графике желтым цветом. Эта плоскость на самом деле бесконечна, ее границы на графике показаны для лучшего визуального восприятия.

График “c” (слева внизу) показывает исходные данные (синие точки) и их проекции на пространство главных компонент (красные точки). Все красные точки лежат строго на плоскости. Очевидно, что имеется небольшая разница между положением синих и красных точек. Эта разница (в виде разницы исходных координат) и описывается матрицей остатков, **E**.

График “d” (справа внизу) показывает только проекции в виде красных точек. Эти проекции можно рассматривать в двух системах координат. Если посмотреть на них в системе координат, которая задается главными компонентами, то каждая красная точка имеет две координаты — ее положение вдоль ГК1 и вдоль ГК2. Эти координаты и называются *счетами* (англ. *scores*). Счета задаются матрицей счетов **T**, число строк в ней равно числу измерений, а число столбцов — числу главных компонент.

Собственно, рисунок 2.13 дает графическое представление МГК разложения, которое мы обсудили в предыдущем разделе.

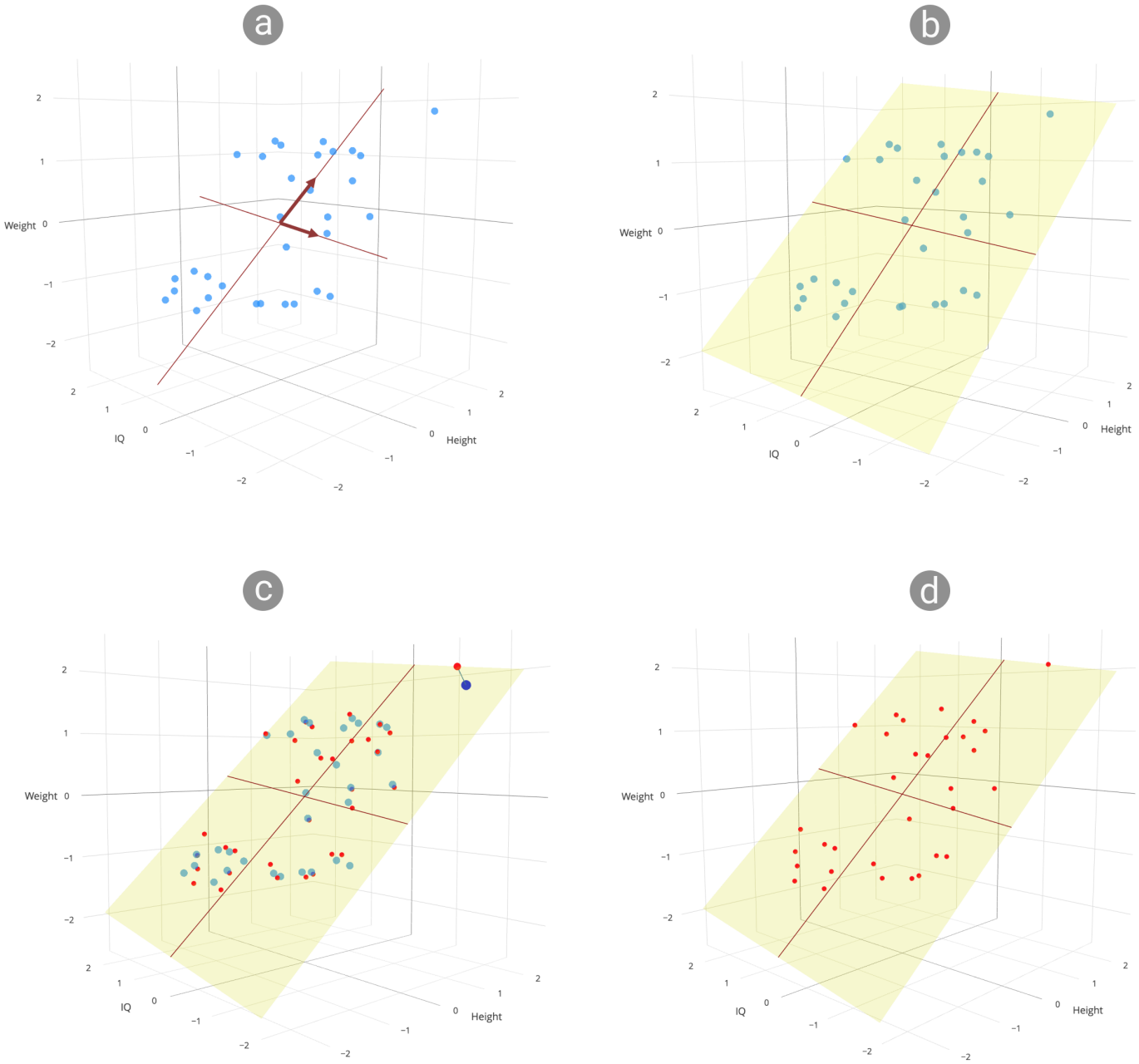


Рис. 2.13. Представление данных в пространстве главных компонент.

2.5.1 График нагрузок

Таблица ниже показывает значения матрицы нагрузок **P**, для примера, показанного на рисунке 2.13. Как мы можем видеть, каждая компонента представлена в виде вектора с тремя координатами. Можно легко проверить, что длина векторов равна единице (кстати, как?).

	Comp 1	Comp 2
Height	-0.705	-0.034
IQ	0.113	-0.992
Weight	-0.700	-0.125

Рассмотрим значения нагрузок для первой компоненты. Очевидно, что наибольшее влияние на ориентацию ГК1 оказывают переменные *Вес* (*Weight*) и *Рост* (*Height*), значения их нагрузок почти равны. Тогда как значение для коэффициента интеллекта (*IQ*) в несколько раз меньше. Если вспомнить, что нагрузки по сути представляют собой нормированные значения ковариации между этими переменными и счетами (координатами проекций точек расположенных вдоль этой компоненты), то мы можем сделать следующие выводы:

1. Имеется сильная ковариация между ростом людей из выборки и счетами для ГК1.
2. Имеется сильная ковариация между весом людей из выборки и счетами для ГК1.

Это возможно только если рост и вес также скоррелированы между собой. Собственно ГК1 — это комбинация роста и веса, “физический” размер людей из этого набора.

Так как IQ практически не влияет на ориентацию ГК1, то мы можем сделать вывод о том, что коэффициент интеллекта не связан с ростом, или весом людей. Собственно, мы это знали и так, МГК лишь выявил эти закономерности, которые мы и ожидали увидеть. Однако, хорошая новость в том, что мы не использовали наши знания для построения МГК модели. Т.е. если в наших данных имеются подобные закономерности, о которых мы не подозреваем, то МГК сможет их выявить.

Если посмотреть на значения для ГК2, то можно заметить, что ситуация практически обратная — вес и рост имеют практически нулевые нагрузки, тогда как значение нагрузки для IQ равно почти единице. Это соответствует нашим знаниям об МГК, а именно тому, что все главные компоненты ортогональны, т.е. если ГК1 объясняет дисперсию данных, связанную с вариацией веса и роста, то ГК2 объясняет дисперсию не связанную с этими двумя переменными — в нашем случае, коэффициент интеллекта.

Такая интерпретация, основанная на непосредственном сравнении значений нагрузок, может быть оправдана в случае небольшого числа исходных переменных, как наш. Однако, если переменных много, то это не самый эффективный способ. Улучшить ситуацию можно с помощью представления нагрузок в виде графика. При этом есть два варианта, либо строить график дисперсии для двух выбранных компонент.

Либо показывать значения нагрузок для одной компоненты в виде столбчатой диаграммы, или линейного графика.

Начнем с графика дисперсии, показанного на рисунке 2.14.

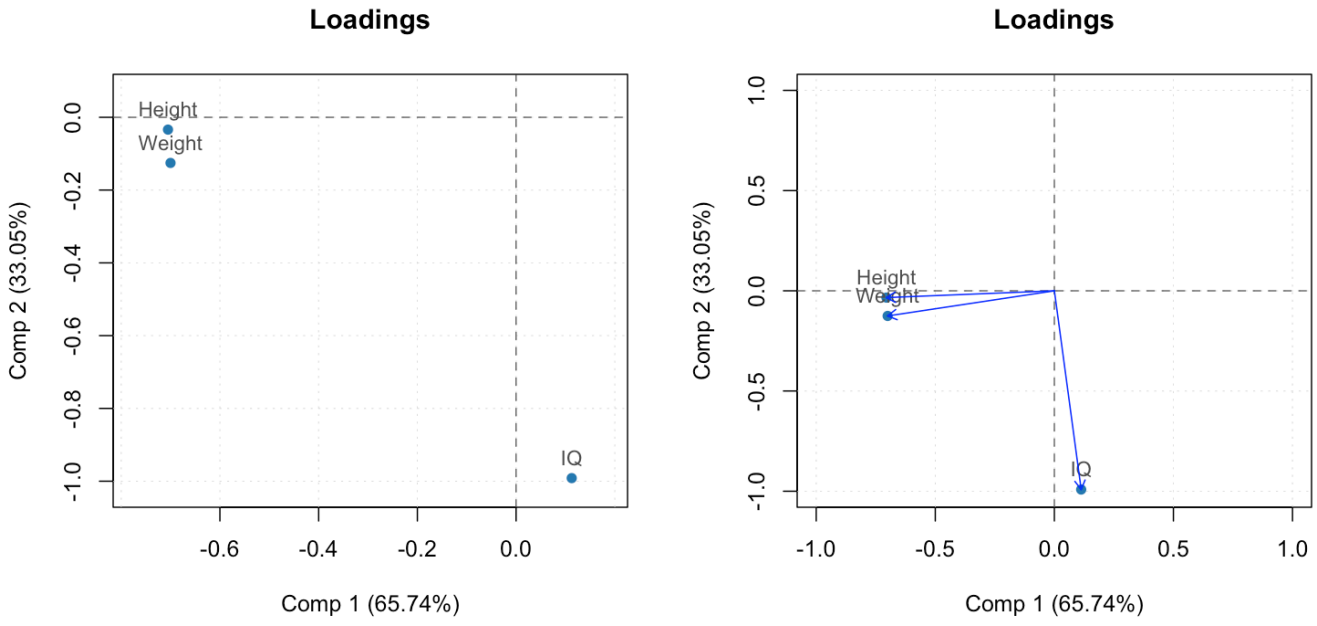


Рис. 2.14. График нагрузок.

Слева показан исходный график, тогда как график справа показывает то же самое, но разброс значений по обеим осям нормирован к [-1, 1]. Кроме этого, на втором графике переменные показаны также в виде векторов, а не только точек.

Есть несколько общих правил анализа графика нагрузок, а именно:

1. Нужно рассматривать только переменные, которые имеют большое значение для ориентации ГК, показанных на графике. Другими словами, точки, соответствующие этим переменным, должны лежать достаточно далеко от начала координат.
2. Если точки находятся далеко от начала координат в одном и том же направлении (т.е. находятся близко друг к другу), то соответствующие переменные имеют одинаковое влияние на ориентацию той ГК, к которой эти точки расположены ближе всего, и эти переменные скоррелированы между собой. В этом случае мы можем сказать, что угол между соответствующими векторами очень мал, как, в нашем случае, для веса и роста.
3. Если точки находятся далеко от начала координат в противоположных направлениях, но лежат почти на одной прямой (т.е. угол между соответствующими векторами близок к 180 градусам), то

переменные, соответствующие этим точкам имеют отрицательную корреляцию. В нашем примере такие переменные отсутствуют, но чуть позже мы покажем другой пример, где такая ситуация также имеет место быть.

4. Наконец, если угол между векторами, которые соответствуют переменным на графике, близок к 90 градусам, как в случае роста и коэффициента интеллекта на рисунке 2.14, то такие переменные имеют очень небольшую корреляцию, либо она и вовсе отсутствует.

Для того, чтобы изучить влияния переменных на отдельные компоненты можно воспользоваться столбчатой диаграммой или линейным графиком, как показано на рисунке 2.15. Линейный график имеет смысл использовать, если порядок переменных важен, например, в случае спектральных данных.

2.5.2 График счетов

Таблица ниже показывает значения счетов из матрицы **T**. Так как в этом случае число строк довольно велико (32), мы ограничились только первыми пятью измерениями.

	Comp 1	Comp 2
Lars	-3.153	0.922
Peter	-1.525	-1.410
Rasmus	-1.436	-1.154
Lene	1.276	0.423
Mette	0.378	0.465

Счета представляются графически точно таким же образом, как и нагрузки — либо в виде диаграммы рассеяния для выбранных двух компонент, либо в виде столбчатой диаграммы, или линейного графика для одной компоненты. Рисунок 2.16 демонстрирует два графика счетов в качестве примера.

Графики счетов и нагрузок, представленные в виде диаграммы рассеяния лучше всего рассматривать вместе. Дело в том, что направления, которые задаются расположением переменных на графике нагрузок соответствуют напрямую направлениям изменения значений этих переменных на графике счетов. Лучше всего это можно проиллюстрировать с помощью цветовых градиентов, как показано на рисунке 2.17.

Как можно видеть, направление цветового градиента, показывающего изменение роста на графике счетов, совпадает с направлением, которое задает переменная *Рост* на графике нагрузок. Так, наиболее высокие люди расположены на левой стороне графика, в области отрицательных счетов. Как можно заметить, величина нагрузки для переменной *Рост* также отрицательна (-0.705). Соответственно, люди с небольшим ростом находятся на противоположной стороне.

Графики на рисунке 2.18 идентичны предыдущим за одним исключением — цветовой градиент на графике счетов теперь показывает вариацию значений коэффициента интеллекта. Опять же легко заметить, что

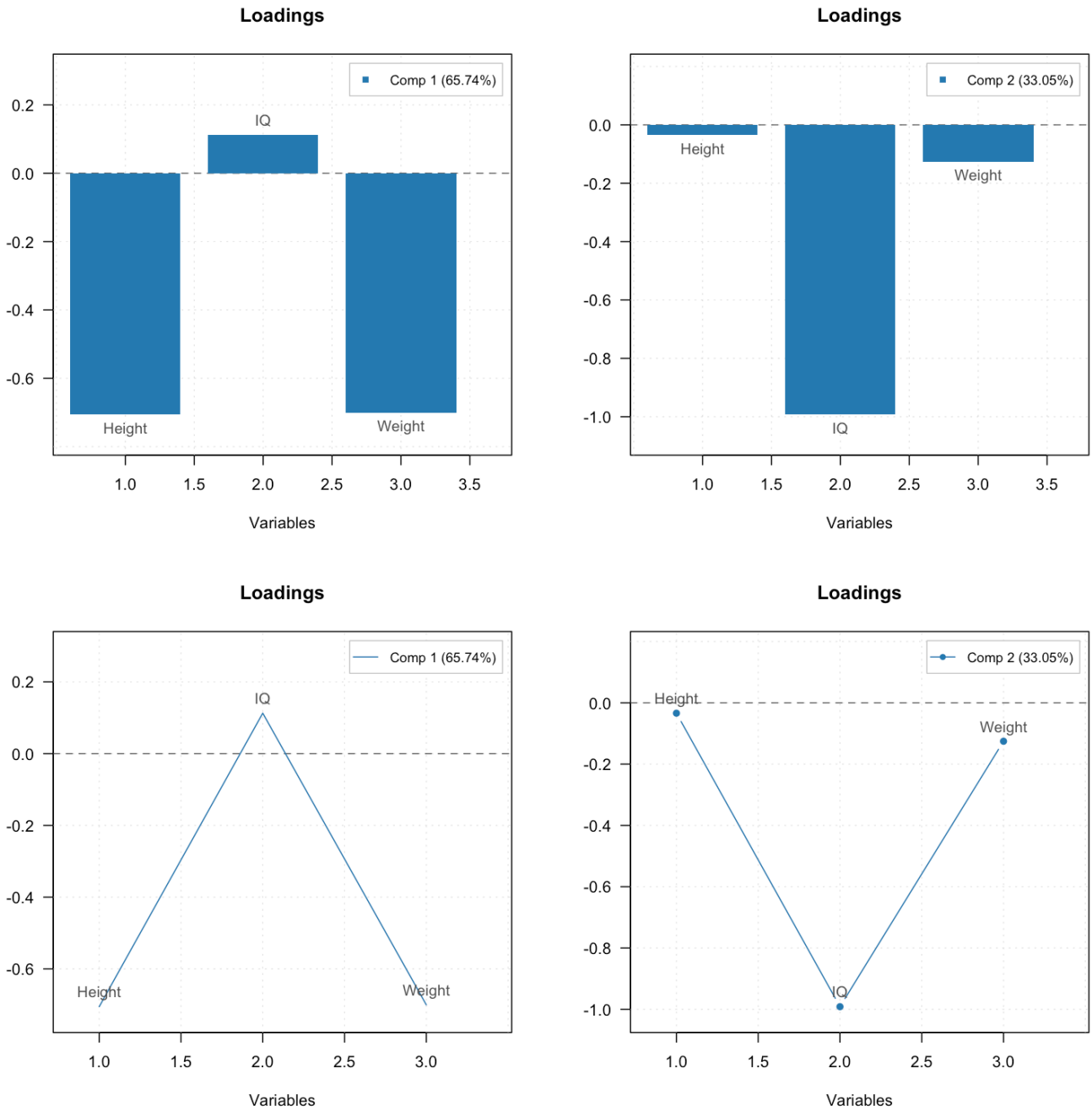


Рис. 2.15. Графики нагрузок в виде столбчатой диаграммы и линейных графиков.

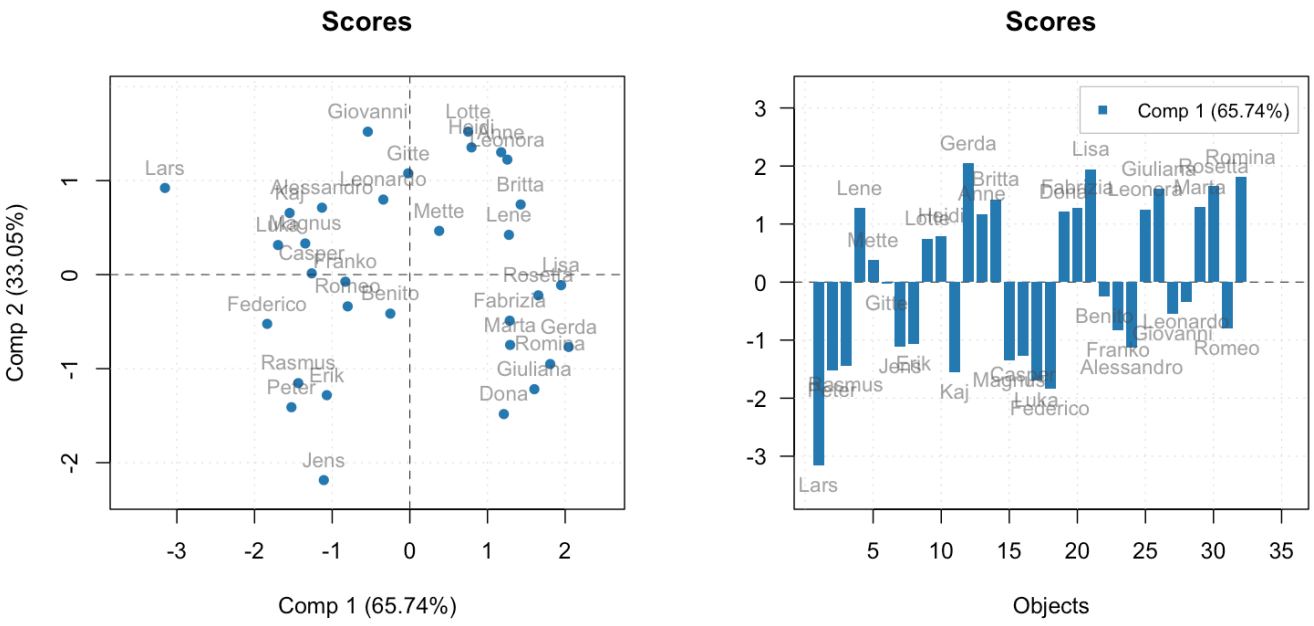


Рис. 2.16. Графики счетов в виде диаграммы рассеяния (слева) и столбчатой диаграммы (справа).

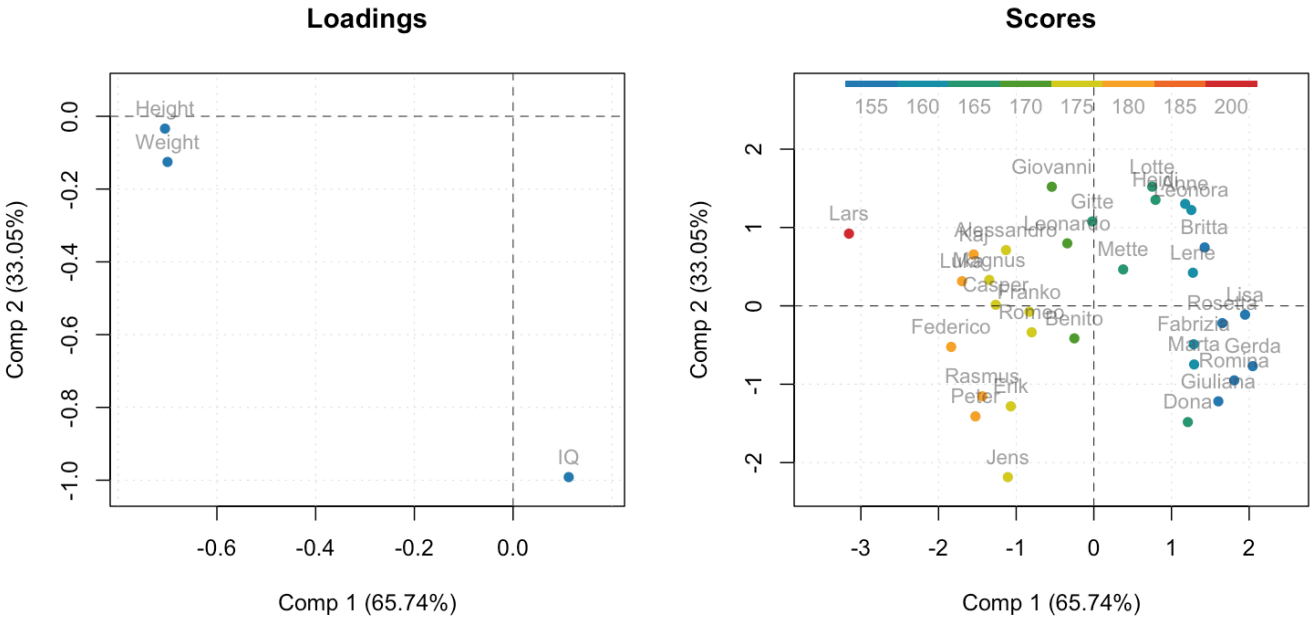


Рис. 2.17. Графики нагрузок (слева) и счетов (справа) для ГК1 и ГК2. Точки на графике счетов раскрашены в соответствии со значениями роста индивидуумов.

направление цветового градиента на графике счетов совпадает с направлением, на которое указывает переменная *IQ* на графике нагрузок.

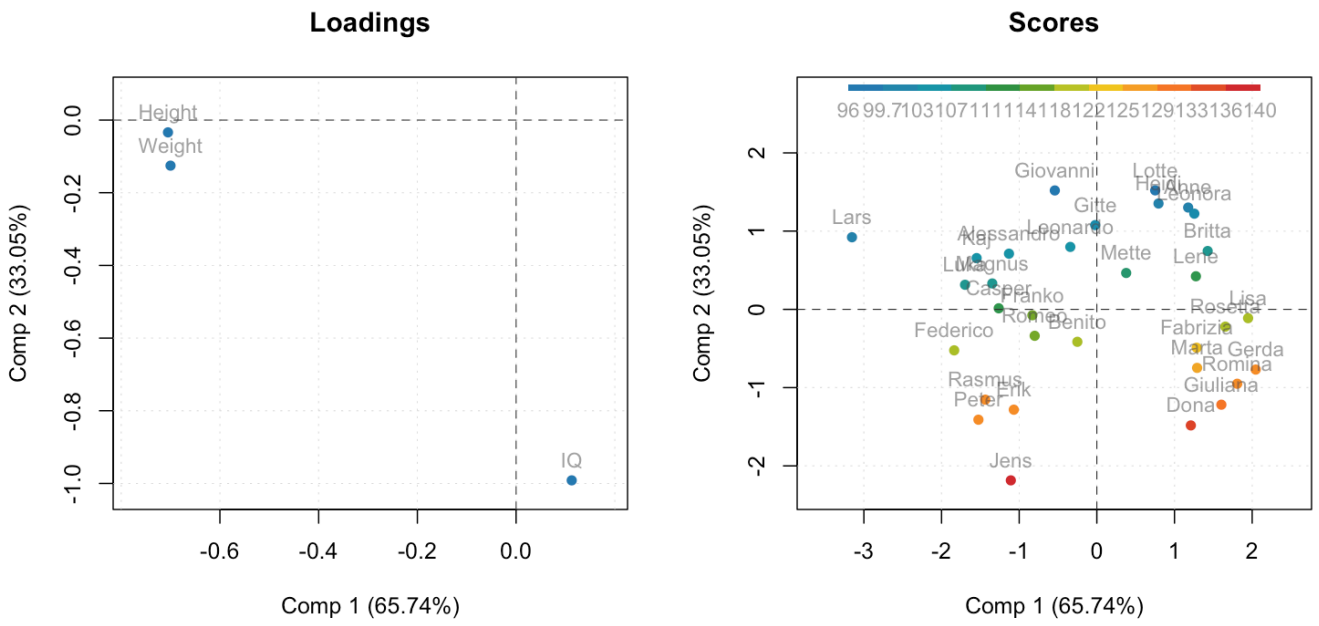


Рис. 2.18. Графики нагрузок (слева) и счетов (справа) для ГК1 и ГК2. Точки на графике счетов раскрашены в соответствии со значениями индекса интеллекта.

Собственно, эти два примера иллюстрируют почему, если направления, заданные двумя переменными на графике нагрузок, перпендикулярны, то такие переменные не коррелируют между собой. Цветовые градиенты показывают наглядно, что и у людей с маленьким ростом, и у людей с большим ростом вариация интеллекта примерно одинаковая.

2.5.3 Сдвоенный график (biplot)

Если число наблюдений и переменных не очень большое, то вместо параллельного изучения графиков счетов и нагрузок, их можно объединить в один график — *biplot*. Нагрузки на этом графике обычно показываются в виде векторов, или линий, а счета — в виде точек. И счета и нагрузки нормируются так, чтобы их значения лежали в пределах от -1 до 1.

Рисунок 2.19 показывает такой график для нашего примера.

2.5.4 Графики счетов и нагрузок для набора данных Люди

Проведем быстрый анализ графиков счетов и нагрузок, сделанных для полного набора данных *Люди*. Перед этим рекомендуем прочесть еще раз раздел 3.2.2, где мы анализировали эти данные с помощью тепловой карты матрицы корреляции и выявили наличие четырех кластеров.

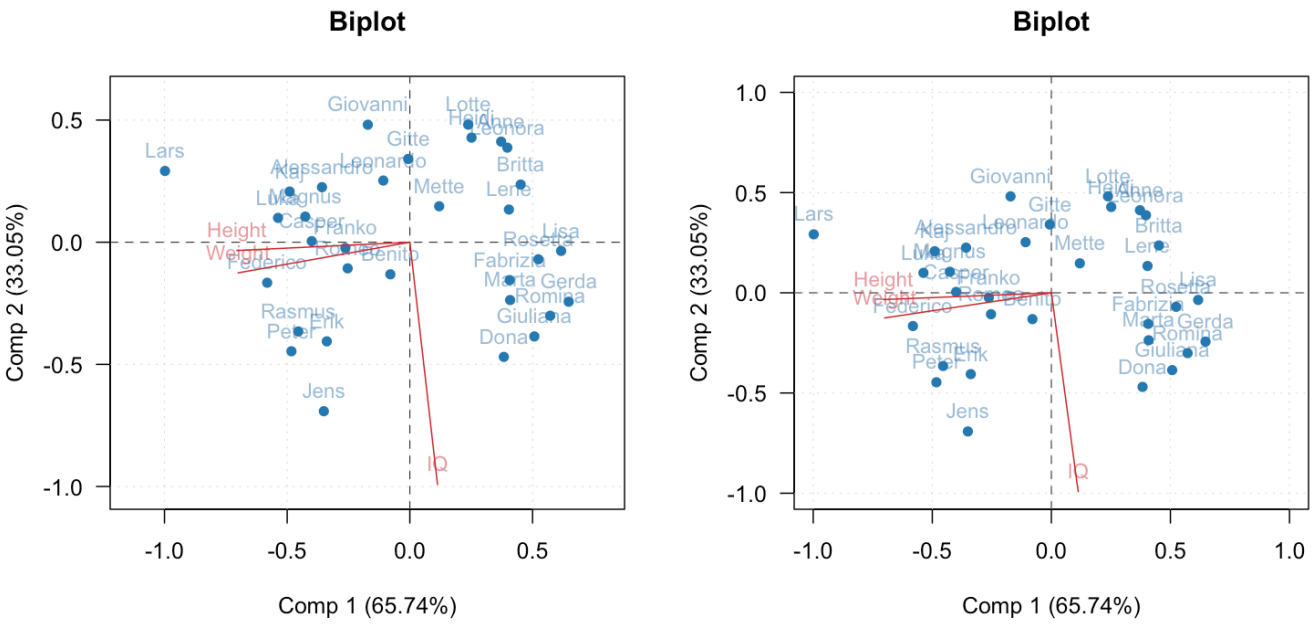


Рис. 2.19. Сдвоенный график счетов и нагрузок (biplot)

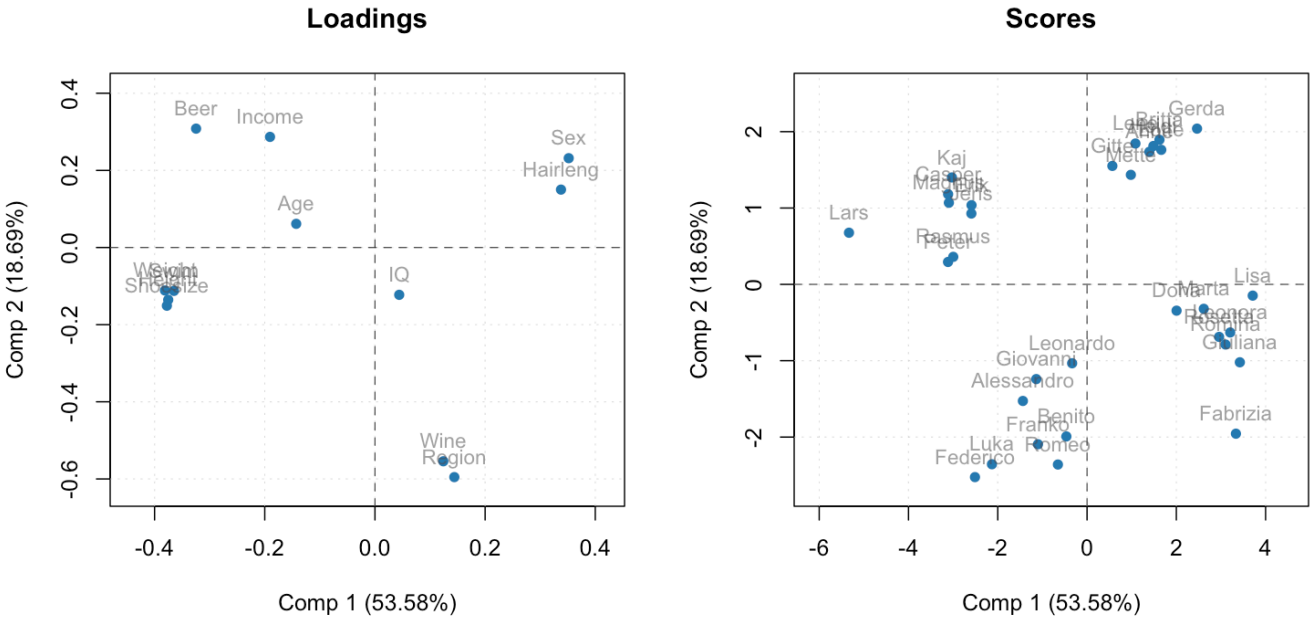


Рис. 2.20. Графики нагрузок (слева) и счетов (справа) для первых двух компонент МГК разложения набора данных *Люди*.

Рисунок 2.20 показывает графики нагрузок (слева) и счетов (справа) для ГК1 и ГК2. Очевидно, что на ориентацию ГК1 влияет большое число переменных, которые собраны в два кластера. Кластер слева состоит из таких переменных, как *вес*, *рост*, *размер обуви*, и *умение плавать*. Кластер справа состоит из переменных *пол* и *длина волос*.

Хорошо видно, что все переменные в первом кластере расположены близко друг к другу на графике нагрузок и далеко от начала координат. Т.е., между этими переменными имеется положительная корреляция. Пол и длина волос имеют отрицательную корреляцию с переменными из первого кластера. Другими словами, ГК1 описывает то же, что и самый большой кластер, найденный нами на тепловой карте корреляционной матрицы, а именно — дисперсию данных, связанную с тем, что в нашей выборке имеются люди мужского и женского пола. В первую очередь это дает большой разброс роста и веса этих людей и связанных с ними характеристик (размер обуви и умение плавать). Вариация значений этих переменных составляет примерно 53% от всей вариации данных (что неудивительно, так как здесь задействована половина всех переменных).

Можно также заметить, что пол и длина волос, хоть и расположены близко к друг другу, но не идентичны, несмотря на то, что обе переменных являются фиктивными и имеют бинарные значения (в нашем случае -1 и $+1$). Очевидно, это связано с тем, что некоторые мужчины в выборке имеют длинные волосы, а некоторые женщины — короткие.

Если посмотреть на график счетов, то можно заметить, что все люди разделены на четыре кластера. При этом ГК1 делит людей по полу (это можно видеть прочитав подписи к точкам). Человек с именем Lars при этом сильно выделяется среди остальных. Это связано с тем, что он является самым высоким (192 см), имеет наибольший вес (92 кг) и размер обуви (48). Он также является самым сильным пловцом (98).

Переменные *Region* (регион) и *Wine* (потребление вина) имеют наибольшее влияние на ориентацию второй главной компоненты, ГК2. Помимо этого, мы можем заметить, что потребление пива и доход, также важны для ее ориентации. При этом потребление пива, во-первых, имеет небольшую отрицательную корреляцию с потреблением вина (угол между соответствующими векторами на графике счетов хоть и не равен 180 градусам, но довольно близок к ним), а во-вторых, имеет влияние на ориентацию обеих компонент, ГК1 и ГК2. Т.е., и пол человека и то, где он живет, влияет на то, сколько пива потребляет этот человек.

Что неудивительно, если вспомнить, что в нашей выборке собраны люди из скандинавских стран (где пиво является традиционным алкогольным напитком), и из средиземноморских (где люди предпочитают вино). Кроме этого мы знаем, что уровень и стоимость жизни, как и доход людей в скандинавских странах в среднем выше.

График счетов подтверждает наше наблюдение, показывая, что в двух кластерах сверху находятся люди с именами, характерными для юга европы, тогда как люди из кластеров внизу графика счетов имеют преимущественно скандинавские имена. Можно также заметить, что Fabrizia, находится немого дальше

других в направлении, заданном переменной Вино. Если посмотреть на исходные данные, то мы увидим, что она выпивает 245 литров вина в год, что на 20% выше следующего по величине значения.

Рисунок 2.21 показывает аналогичные графики для ГК1-ГК3 (вверху) и ГК1-ГК4 (внизу). Из графиков очевидно, что ГК3, отвечает за дисперсию данных связанную с возрастом людей из выборки. При этом имеется только один параметр, которые коррелирует с возрастом — это среднегодовой доход. Как можно заметить, корреляция не самая сильная, точки находятся на некотором расстоянии друг от друга, но имеет место быть. При этом график счетов позволяет сделать вывод о том, что Lars имеет наибольший доход, а Leonardo является самым возрастным представителем выборки.

Наконец, ГК4, по сути совпадает с переменной IQ, что еще раз подтверждает отсутствие корреляции между коэффициентом интеллекта и остальными 11 характеристиками. Изучение графика счетов позволяет определить, что Jens обладает наибольшим коэффициентом (140).

Изучение графиков счетов и нагрузок начиная с ГК5, не позволяет дать какую-то осмысленную интерпретацию поведения точек на графиках, что обычно означает, что эти компоненты описывают случайную вариацию в данных.

Этот небольшой пример показывает, что МГК может эффективно выявлять скрытые закономерности в данных, такие как, например, наличие групп, трендов, экстремальных значений, с помощью анализа относительно небольшого числа графиков.

2.6 Объясненная и остаточная дисперсия

Мы уже обсуждали ранее объясненную и остаточную дисперсию, рассмотрим теперь эту тему более подробно. Для начала вспомним, что по определению дисперсия — среднеквадратичное расстояние от точек с данными до среднего значения:

$$s^2(x) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - m_x)^2$$

Формула выше показывает расчет дисперсии для одномерного случая. Если же переменных две или больше, то необходимо просто считать Евклидово расстояние, а точнее его квадрат. В случае с методом главных компонент вычисления упрощаются, так как данные центрированы, т.е. среднее значение каждой переменной равно нулю. Т.е., квадрат расстояния от точки, координаты которой находятся в строке i, до начала координат рассчитывается как сумма квадратов значений из соответствующей строки матрицы данных $\mathbf{X}$:

$$d_i^2 = x_{i1}^2 + x_{i2}^2 + ... + x_{im}^2$$

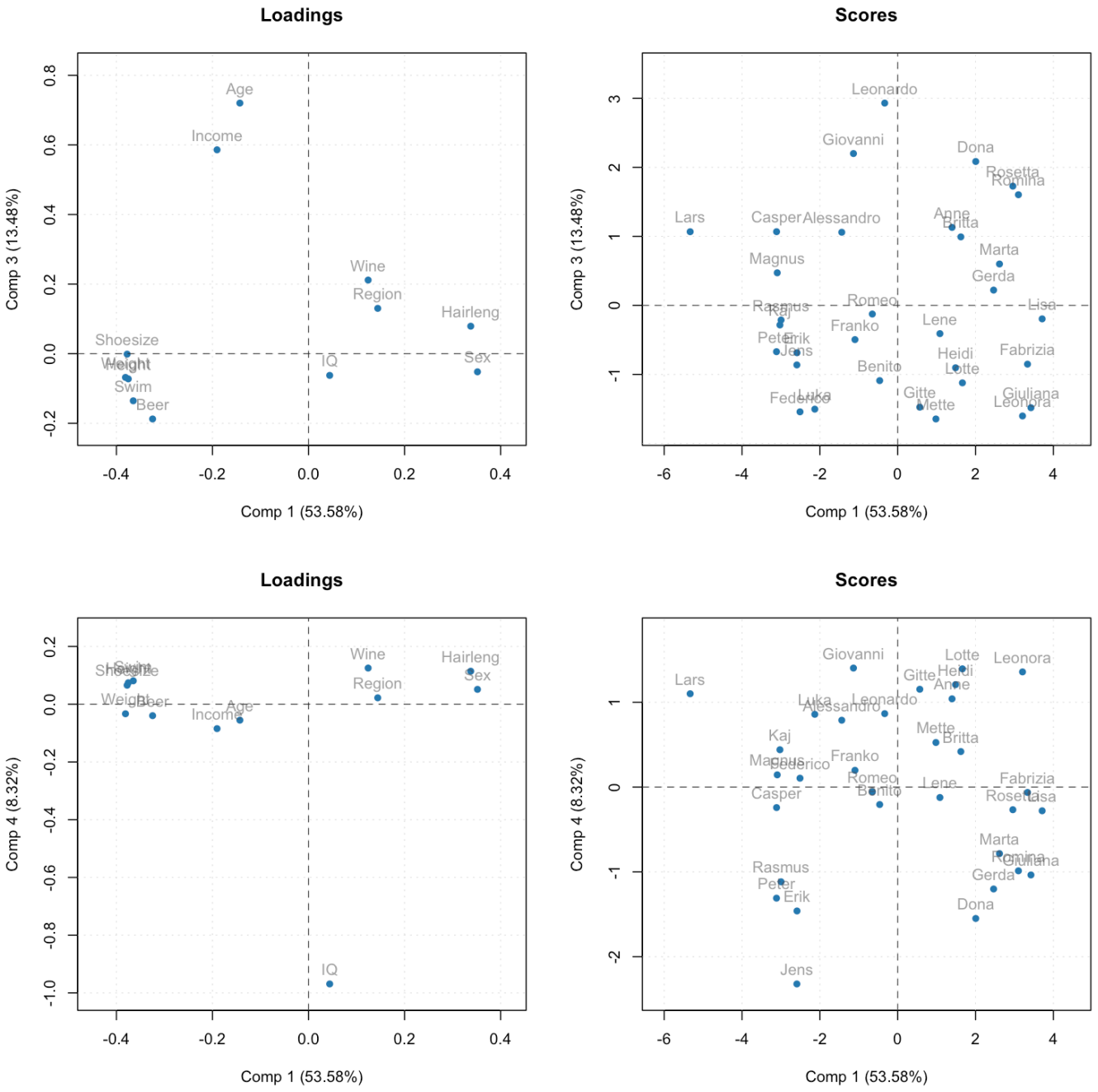


Рис. 2.21. Графики нагрузок и счетов для ГК1-ГК3 (сверху) и ГК1-ГК4 (снизу).

Здесь x_{ij} это значение из строки i и столбца j матрицы с данными $\mathbf{X}$. Здесь мы подразумеваем, что матрица $\mathbf{X}$ содержит уже центрированные значения, т.е. среднее значение каждого столбца матрицы равно нулю.

В этом случае полную дисперсию данных можно посчитать как:

$$s^2_{tot} = \frac{1}{n-1}\sum_{i=1}^{n} d_i^2 = \frac{1}{n-1} SS_{tot}$$

С помощью SS_{tot} мы обозначили сумму квадратов расстояний от точек до начала координат (SS — это сокращение для *sum of squares*, суммы квадратов).

Рассмотрим теперь простой пример. Пусть у нас имеются четыре наблюдения для двух переменных. Их центрированные значения заданы следующей матрицей:

$$\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 3 & 1 \\ -3 & -1 \\ -1 & -2 \end{bmatrix}$$

Квадрат расстояния от каждой точки до начала координат можно посчитать как:

$$\mathbf{f} = \begin{bmatrix} 1^2 + 2^2 \\ 3^2 + 1^2 \\ (-3)^2 + (-1)^2 \\ (-1)^2 + (-2)^2 \end{bmatrix} = \begin{bmatrix} 5 \\ 10 \\ 10 \\ 5 \end{bmatrix}$$

Здесь $\mathbf{f} = [f_1, f_2, ..., f_n]^\mathrm{T} = [d_1^2, ..., d_n^2]^\mathrm{T}$, это вектор с квадратами расстояний от исходных точек до начала координат, т.е. $f_1 = d_1^2$. Сумма квадратов расстояний получится путем суммирования этих значений:

$$SS_{tot} = 5 + 10 + 10 + 5 = 30$$

Заметим, что сумму квадратов можно получить и проще, просто просуммировав все значения матрицы $\mathbf{X}$:

$$SS_{tot} = 1^2 + 2^2 + 3^2 + 1^2 + (-3)^2 + (-1)^2 + (-1)^2 + (-2)^2 = 30$$

К примеру, в R это можно сделать с помощью одной строки кода:

```
SStot <- sum(X^2)
```

Предположим, что нам нужно спроецировать эти точки на главную компоненту ГК1, которая задается матрицей нагрузок $\mathbf{P} = [0.8, 0.6]^\mathrm{T}$. Для начала вычислим матрицу счетов $\mathbf{T}$:

$$\mathbf{T} = \mathbf{XP} = \begin{bmatrix} 1 & 2 \\ 3 & 1 \\ -3 & -1 \\ -1 & -2 \end{bmatrix} \times \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix} = \begin{bmatrix} 0.8 + 1.2 \\ 2.4 + 0.6 \\ -(2.4) + (-0.6) \\ (-0.8) + (-1.2) \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ -3 \\ -2 \end{bmatrix}$$

Чтобы найти координаты проекций в исходном пространстве переменных, нужно умножить матрицу счетов на транспонированную матрицу нагрузок:

$$\mathbf{TP}^\mathrm{T} = \begin{bmatrix} 2 \\ 3 \\ -3 \\ -2 \end{bmatrix} \times \begin{bmatrix} 0.8 & 0.6 \end{bmatrix} = \begin{bmatrix} 1.6 & 1.2 \\ 2.4 & 1.8 \\ -2.4 & -1.8 \\ -1.6 & -1.2 \end{bmatrix}$$

Рисунок 2.22 иллюстрирует положение исходных точек (обозначены синим цветом), главной компоненты и проекций исходных точек на нее (показаны красным цветом). Соответствующие расстояния и координаты показаны только для первой точки O_1 (которая соответствует первой строке нашей матрицы данных).

Очевидно, что исходные (синие) точки находятся дальше от начала координат, чем их проекции (красные). Т.е., проецируя точки на главную компоненты мы часть дисперсии теряем, но какую часть? Вычислим сумму квадратов расстояний от красных точек до начала координат. Для этого нужно просуммировать квадраты значений матрицы $\mathbf{TP}^\mathrm{T}$:

$$SS_{exp} = 1.6^2 + 1.2^2 + 2.4^2 + 1.8^2 + (-2.4)^2 + (-1.8)^2 + (-1.6)^2 + (-1.2)^2 = 26$$

Как мы и ожидали, дисперсия красных точек (проекций) немного меньше, а точнее она составляет $26/30 \times 100\% = 86.7\%$ от дисперсии исходных данных. Таким образом мы можем сказать, что в данном случае первая главная компонента объясняет 86.7% дисперсии исходных данных. Эта величина называется *объясненной* дисперсией (англ. *explained variance*).

На самом деле, чтобы посчитать эти расстояния, нам достаточно было использовать координаты вдоль ГК1, т.е. счета, t_i. Как можно видеть, сумма квадратов счетов будет также равна 26: $2^2 + 3^2 + (-3)^2 + (-2)^2 = 4 + 9 + 9 + 4 = 26$.

Посчитаем теперь расстояния от красных точек, лежащих на главной компоненте, до исходных (синих) точек с данными. Для начала нужно вычесть координаты двух наборов точек:

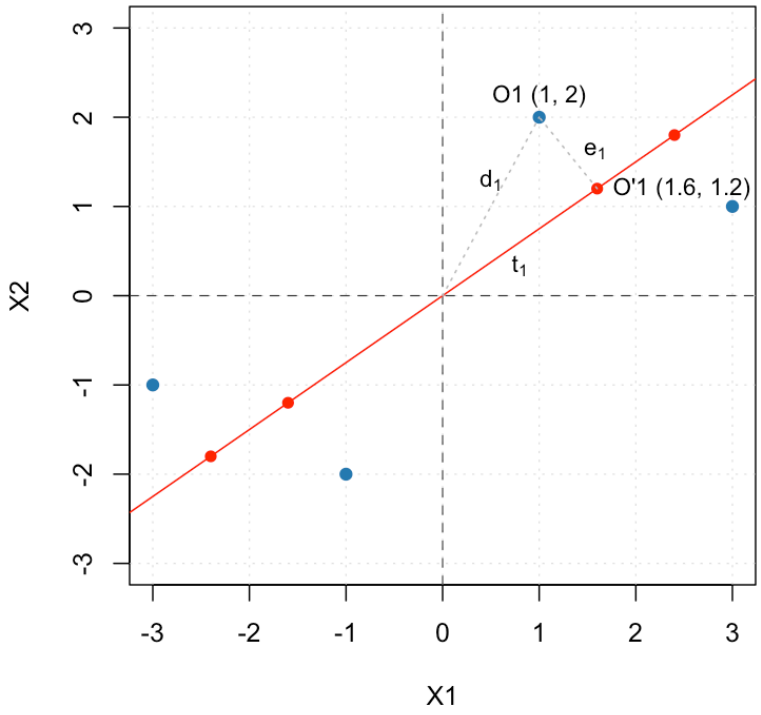


Рис. 2.22. Исходные точки из центрированного набора данных и их проекции на ГК1.

$$\mathbf{E} = \mathbf{X} - \mathbf{T}\mathbf{P}^{\mathrm{T}} = \begin{bmatrix} 1 - 1.6 & 2 - 1.2 \\ 3 - 2.4 & 1 - 1.8 \\ (-3) - (-2.4) & (-1) - (-1.8) \\ (-1) - (-1.6) & (-2) - (-1.2) \end{bmatrix} = \begin{bmatrix} -0.6 & 0.8 \\ 0.6 & -0.8 \\ -0.6 & 0.8 \\ 0.6 & 0.8 \end{bmatrix}$$

Теперь просуммируем квадраты разности координат в каждой строке, чтобы получить квадрат расстояния от исходной точки до ее проекции, как показано ниже. Это расстояние обозначено на графике буквой e, а для его квадрата в этом пособии будем использовать букву $q = e^2$. Соответственно, вектор значений $[q_1, ..., q_n]$ с квадратами расстояний для каждой точки из нашей выборки будем обозначать как $\mathbf{q}$.

Выбранные нами случайные данные имеют забавный эффект — все расстояния q_i равны единице, что, с другой стороны, делает вычисления проще. Это можно заметить и на графике — все красные точки находятся на равном расстоянии от синих.

$$\mathbf{q} = \begin{bmatrix} (-0.6)^2 & 0.8^2 \\ 0.6^2 & (-0.8)^2 \\ (-0.6)^2 & 0.8^2 \\ 0.6^2 & 0.8^2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Сумма этих квадратов даст нам возможность вычислить остаточную дисперсию (англ. *residuals variance*). Так как мы вычисляем дисперсию относительную (относительно дисперсии исходных данных), то нам не нужно делить на число степеней свободы, а достаточно только вычислить сумму квадратов расстояний:

$$SS_{res} = 1 + 1 + 1 + 1 = 4$$

Вспомним, что сумма квадратов расстояний от начала координат до исходных точек, $SS_{tot} = 40$, а сумма квадратов расстояний до их проекций, $SS_{exp} = 36$, т.е. $SS_{res} = SS_{tot} - SS_{exp}$. Если вспомнить, что эти три расстояния образуют стороны прямоугольного треугольника, то результат вполне закономерен.

Т.е. остаточная дисперсия в нашем случае равна $4/30 \times 100\% = 13.3$, или ее можно вычислить через объясненную дисперсию, рассчитанную нами ранее: $100\% - 86.7\% = 13.3\%$.

2.6.1 Графики объясненной и остаточной дисперсии

Объясненную дисперсию можно вычислить для каждой индивидуальной компоненты (т.е., узнать какой процент дисперсии именно эта компонента объясняет), а можно аккумулировать, чтобы показать какой процент дисперсии объясняют несколько главных компонент вместе (например первые две или три компоненты). В последнем случае такую дисперсию называют полной (англ. *total*), или кумулятивной (англ. *cumulative*). Также можно поступить и с остаточной дисперсией.

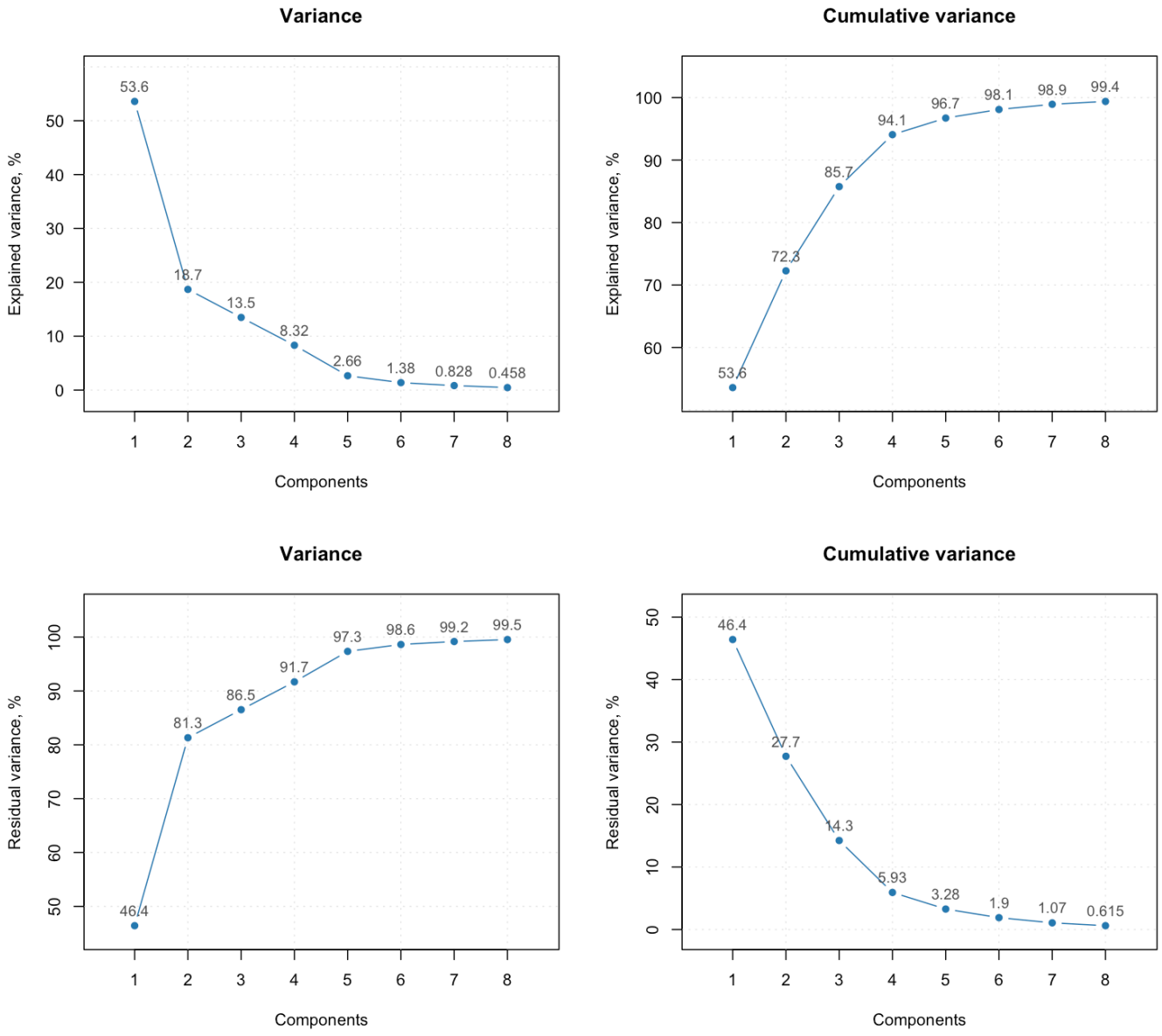


Рис. 2.23. Графики объясненной и остаточной дисперсии для данных *Люди*. Графики слева показывают индивидуальные значения для каждой компоненты, графики справа — кумулятивные значения.

Вычисленные таким образом значения удобно представить с помощью столбчатой диаграммы или линейного графика. Рисунок 2.23 показывает четыре графика, сделанных для МГК разложения полного набора данных *Люди* для 8 компонент. Графики слева показывают индивидуальную дисперсию, а справа — кумулятивную. Верхние графики показывают дисперсию объясненную, а нижние — остаточную.

Если посмотреть на график полной (кумулятивной) остаточной дисперсии, который показан снизу справа, то можно заметить, что он похож на руку человека, согнутую в локте. При этом "локоть" находится в области четвертой главной компоненты.

Это довольно субъективный, но иногда полезный способ определения оптимального числа компонент, которые описывают систематическую вариацию. Его так и называют — *правило локтя* (англ. *elbow rule*). В данном случае это правило сработало верно, но обычно лучшим способом определения оптимального числа компонент является изучение и интерпретация графиков счетов и нагрузок, а также графика расстояний, о котором пойдет речь в следующем разделе.

2.7 Расстояния и скрытые структуры данных

Объясненная и остаточная дисперсия дают нам представление насколько хорошо МГК модель описывает поведение данных. Однако, это усредненное представление, некая общая характеристика для всей выборки. В этом разделе мы рассмотрим, как можно использовать этот подход для того, чтобы оценить насколько хорошо модель описывает отдельное измерение, или их группу.

Для начала, давайте еще раз посмотрим на график, который мы использовали в предыдущем разделе для визуализации объясненной и остаточной дисперсии. Для удобства приводим его еще раз, см. рисунок 2.24.

Если рассмотреть отношение между каждой отдельной точкой, которая соответствует исходным данным, и МГК моделью (в этом случае она представлена лишь одной компонентой), то это отношение задается двумя расстояниями. Здесь и далее мы будем говорить о квадратах расстояний, но, для удобства изложения, слово "квадрат" будем иногда пропускать.

Итак, первое расстояние, о котором идет речь, — это ортогональное расстояние от точки до модели. Оно обозначено на графике как e_1, а его квадрат, как мы обсуждали в предыдущем разделе, обозначим как $q_1 = e_1^2$. По сути, расстояние q дает нам меру того, насколько хорошо данное измерение соответствует общему тренду, задаваемому главными компонентами. Чем больше это расстояние, тем хуже модель описывает поведение данной точки.

Второе расстояние, — это расстояние между проекцией точки на пространство главных компонент и началом координат. На графике это расстояние задано как t_1, в нашем простом случае это величина счета для данной проекции. Чуть позже мы обсудим, как правильно считать это расстояние для случая двух, или более компонент в модели, а пока введем обозначения для квадрата этого расстояния: $h_1 = t_1^2$. Будем называть эту величину расстоянием в пространстве главных компонент (англ. *score distance*).

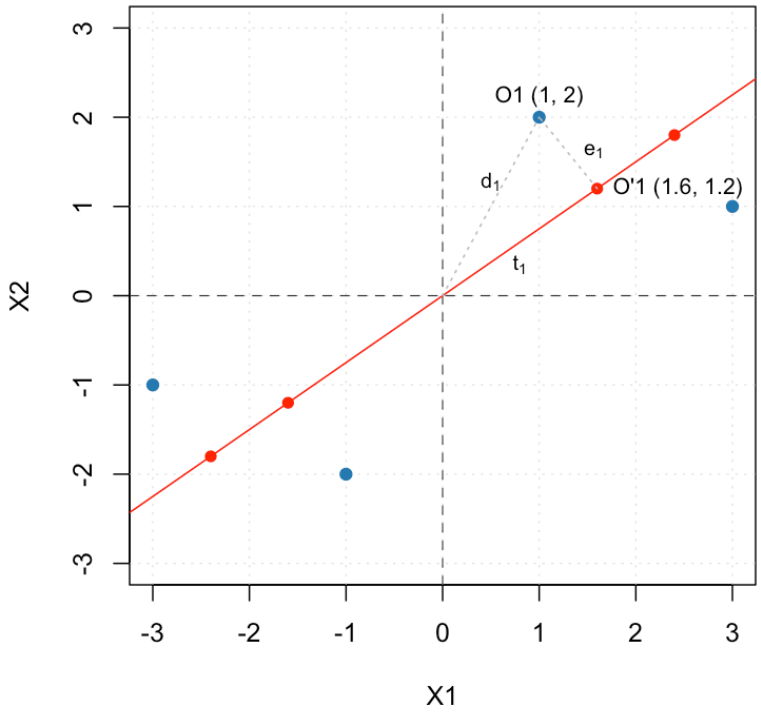


Рис. 2.24. Исходные точки из центрированного набора данных и их проекции на ГК1.

Расстояние в пространстве счетов служит индикатором экстремальности измерения — насколько далеко оно находится от остальных. Для того, чтобы понять смысл этих двух расстояний, приведем простой пример. Пусть переменная x — это возраст людей, а переменная y — их ежегодный доход. При анализе данных *Люди*, мы заметили, что эти две переменные скоррелированы, т.е. чем старше человек, тем, в среднем, выше его доход. Это укладывается в наше общее представление о современном обществе, так как с возрастом, человек, в среднем, имеет больше опыта и знаний и ценится выше как сотрудник.

В этом случае, если мы применим МГК анализ к таким данным, то ГК1 будет описывать общий тренд — соотношение между возрастом и доходом людей в выборке. Пусть в нашей выборке возраст людей варьируется от 18 до 65 лет, а доход — от 2000 до 8000 евро. Рассмотрим теперь четыре случая:

- A. Человек со средним возрастом и средним доходом.
- B. Молодой человек с высоким доходом.
- C. Человек с большим возрастом и высоким доходом.
- D. Человек с возрастом больше 65 лет, но с маленьким доходом.

График слева на рисунке 2.25 иллюстрирует этот пример и упомянутые выше четыре случая. Красная линия задает направление главной компоненты, но в этом случае, для наглядности, мы не стали центрировать и шкалировать данные.

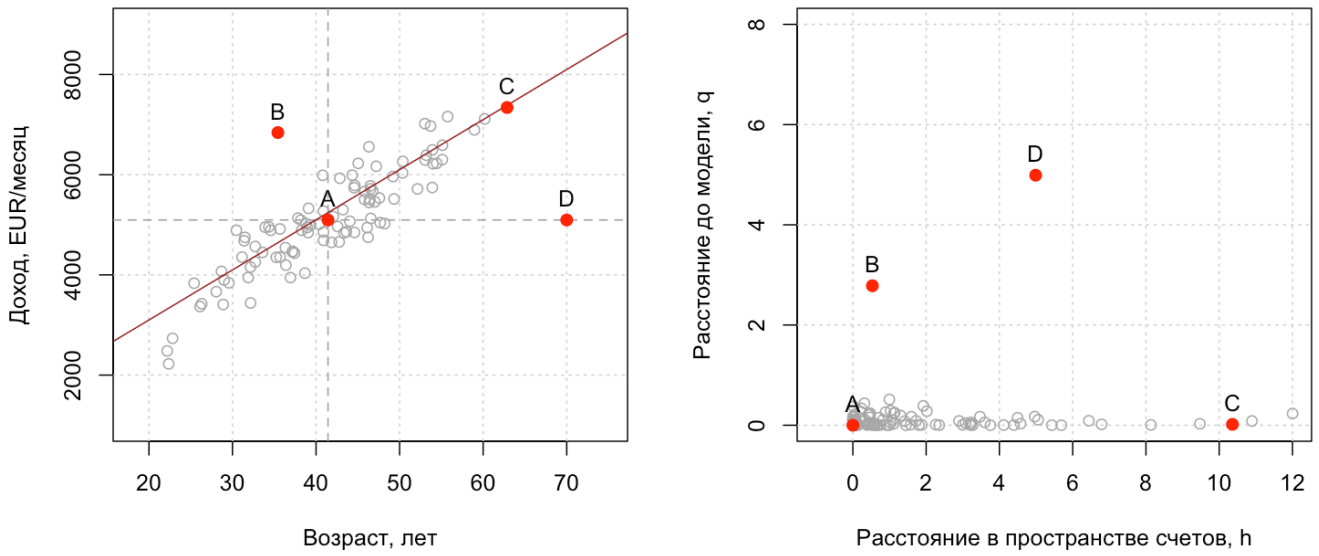


Рис. 2.25. Исходные данные для примера с возрастом и доходом (слева) и соответствующий график расстояний (справа).

Попробуйте оценить эти два расстояния для всех четырех случаев. Очевидно, что в случае A оба

расстояния будут практически равны нулю. Т.е. этот случай отлично вписывается в общий тренд и не является экстремальным.

В случае B, если рассматривать возраст, или доход по отдельности, то ничего необычного в этих значениях нет. Но вот соотношение дохода и возраста не соответствует общему тренду, поэтому ортогональное расстояние будет большим, а расстояние для проекции этой точки на ГК — почти таким же, как и для случая A. Обычно такие *out of trend* наблюдения называют *выбросами* (англ. *outlier*). Чаще всего выброс — это наблюдение из другой генеральной совокупности, т.е. своего рода лимон среди яблок.

Случай C хорошо объясняется моделью, соотношение между возрастом и доходом для этого случае совпадает с общим трендом. Но, так как обе величины довольно велики, точка, которая соответствует этому случаю, сильно удалена от начала координат в направлении ГК1. Т.е. h-расстояние будет одним из самых больших в выборке, в то время как q-расстояние почти равно нулю. Такой случай обычно называется *экстремальным* (англ. *extreme*).

Наконец, случай D является и экстремальным и не вписывается в общий тренд. Соответственно, оба расстояния для этого случая будут велики.

Этот простой пример, по сути, иллюстрирует основную идею использования расстояний — оценка того, насколько хорошо то, или иное измерение “вписывается” в общий тренд. Однако, что делать, если данные по настоящему многомерные и число компонент больше единицы? Вообще-то, это не проблема, так как мы можем считать расстояния между точками для любой размерности. Т.е. независимо от числа переменных и числа компонент, для любого наблюдения мы можем посчитать:

1. Расстояние q, как сумму квадратов соответствующей строки матрицы $\mathbf{E}$
2. Расстояние h, как сумму квадратов соответствующей строки матрицы $\mathbf{T}$

После этого мы можем иллюстрировать эти два расстояния с помощью диаграммы рассеяния, как это показано на графике справа (рис. 2.25). Тут стоит сказать, что для построения этого графика мы, все таки, шкалировали исходные данные, иначе расстояния были бы очень большими из-за большой дисперсии доходов. График расстояний смог очень хорошо выявить все несоответствия в данных, которые мы только что обсудили. А самое главное — такой график можно построить для любой МГК модели, независимо от размерности исходных данных и числа главных компонент.

Однако, есть два нюанса:

1. В пространстве главных компонент использование расстояния Евклида не совсем правильно. Если вы помните, первая компонента всегда имеет самую большую дисперсию, вторая компонента имеет дисперсию ниже, и так далее. Иногда дисперсия счетов первой и второй компонент может различаться в несколько раз. Однако, при вычислении расстояния Евклида вклад каждой компоненты в общее расстояние одинаков. Как вычислить расстояние с поправкой на разницу в дисперсиях?

2. Очевидно, что эти два расстояния связаны между собой. Чем больше компонент в МГК модели, тем меньше расстояние q, и тем больше расстояние h. Каким образом учесть это соотношение?

Поговорим о том, как решаются эти две проблемы.

2.7.1 Расстояние Махаланобиса

Как мы уже обсуждали, расстояние Евклида подразумевает, что все переменные независимы и имеют равный вклад. Что же делать, если ни первое, ни второе условия не выполняются? В этом случае нужно воспользоваться обобщением расстояния Евклида, которое носит название расстояния Махаланобиса, в честь известного индийского математика.

В общем случае, квадрат расстояния Махаланобиса для двух точек, $\mathbf{x} = x_1, x_2, ..., x_J$ и $\mathbf{y} = y_1, y_2, ..., y_J$ в J-мерном пространстве вычисляется как:

$$d^2(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})\mathrm{S}^{-1}(\mathbf{x} - \mathbf{y})^{\mathrm{T}}$$

Здесь, S — это ковариационная матрица размерности $J \times J$. Чтобы понять смысл этого выражения, попробуем немного упростить задачу.

Во-первых, для начала, ограничимся двумерным случаем. Пусть у нас есть выборка точек в двумерном пространстве и нам нужно вычислить расстояние от точки i с координатами $[x_{i1}, x_{i2}]$ до центра (т.е. координаты второй точки равны $[0, 0]$). Тогда формулу можно упростить до следующего соотношения:

$$d_i^2 = \begin{bmatrix} x_{i1} & x_{i2} \end{bmatrix} \begin{bmatrix} s_{x_1}^2 & cov(x_1, x_2) \\ cov(x_2, x_1) & s_{x_2}^2 \end{bmatrix}^{-1} \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix}$$

Здесь, $s_{x_1}^2$ и $s_{x_2}^2$ — это дисперсия переменных x_1 и x_2 соответственно, а $cov(x_1, x_2)$ — ковариация между ними. Если переменные x_1 и x_2 независимы, т.е. ковариация между ними равна нулю, то выражение принимает еще более простой вид:

$$d_i^2 = \begin{bmatrix} x_{i1} & x_{i2} \end{bmatrix} \begin{bmatrix} \frac{1}{s_{x_1}^2} & 0 \\ 0 & \frac{1}{s_{x_2}^2} \end{bmatrix} \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix} = \frac{x_{i1}^2}{s_{x_1}^2} + \frac{x_{i2}^2}{s_{x_2}^2}$$

Т.е., если корреляция между переменными отсутствует, все, что нам нужно, —- это стандартизация переменных, требуется лишь разделить каждую переменную на стандартное отклонение, или квадрат этой переменной на дисперсию.

Обобщая теперь полученную формулу на пространство любой размерности с J переменными, получаем:

$$d_i^2 = \sum_{j=1}^{J} \frac{x_{ij}^2}{s_{x_j}^2}$$

Нужно еще раз повторить, что это упрощенное выражение, которое работает только если между переменными нет корреляции/ковариации. К счастью, это именно тот случай, когда речь идет о МГК счетах, как мы помним, главные компоненты выбираются так, чтобы они были взаимно ортогональны и корреляция между счетами равна нулю.

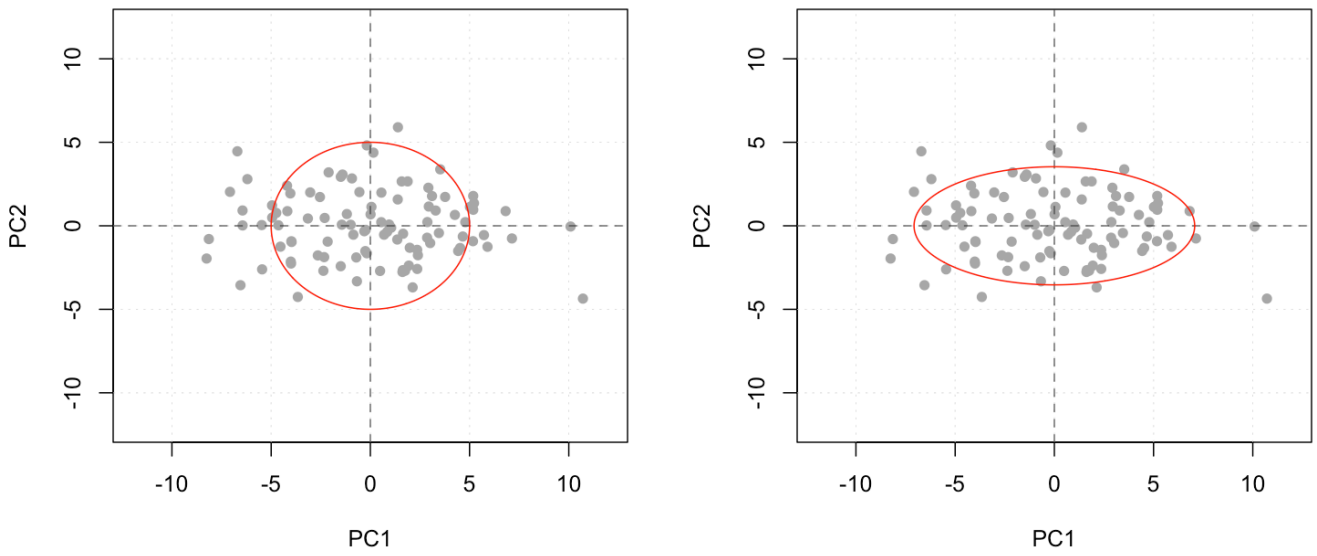


Рис. 2.26. Иллюстрация расстояния Евклида (слева) и расстояния Махаланобиса (справа). Точки на окружности и на эллипсе равноудалены от начала координат.

Т.е. для того, чтобы правильно вычислить расстояние в пространстве счетов, h, нам предварительно нужно посчитать дисперсию счетов для каждой главной компоненты и шкалировать значения счетов на дисперсию. Дисперсия счетов в МГК называется *собственным значением* (англ. *eigenvalue*) компоненты и обычно обозначется греческим символом λ. Таким образом, формулу для вычисления расстояния h для точки i можно записать как:

$$h_i = \sum_{a=1}^{A} \frac{t_{ia}^2}{\lambda_a}$$

здесь t_{ia} - значение счета для точки i и компоненты a (другими словами, элемент матрицы счетов $\mathbf{T}$ из строки i и столбца a), а λ_a — сумма квадратов счетов из столбца a разделенное на число наблюдений минус один.

Чтобы лучше понять смысл расстояния Махаланобиса, приведем простой пример. Графики на рисунке 2.26 показывают счета для ГК1 и ГК2 разложения некоторых случайных данных. Как можно видеть, данные образуют облако точек, причем дисперсия вдоль ГК1 почти в два раза больше дисперсии вдоль ГК2.

Если взять расстояние Евклида, равное 5, от начала координат в разных направлениях, то мы получим набор точек, лежащих на окружности с таким радиусом, как показано на графике слева. Т.е. расстояние Евклида не зависит от направления, так как ГК1 и ГК2 в этом случае вносят равный вклад в это расстояние.

Однако, если посчитать такое же расстояние Махаланобиса, то точки будут лежать на эллипсе, вытянутом вдоль ГК2, как показано на графике справа. В этом случае вклады ГК1 и ГК2 будут разными и небольшое расстояние вдоль ГК2 будет соответствовать более длинной дистанции вдоль ГК1. Так как дисперсия ГК1 в два раза больше дисперсии ГК2, то большая полуось эллипса будет в $\sqrt{2} \approx 1.4$ раза больше 5, а малая — в $\sqrt{2} \approx 1.4$ раза меньше.

2.7.2 График расстояний

Теперь, когда мы знаем, как правильно считать расстояния, приведем R код, который позволяет это сделать. Здесь мы используем функцию `nipals` код которой мы реализовали в конце раздела 3.4. Для начала напомним, как выглядит эта функция:

```
nipals <- function(X, A = 1) {

   # центрируем и шкалируем данные
   X <- scale(X, center = TRUE, scale = TRUE)

   # подготавливаем пустые матрицы для счетов и нагрузок
   scores <- matrix(0, nrow = nrow(X), ncol = A)
   loadings <- matrix(0, nrow = ncol(X), ncol = A)

   # цикл для вычисления каждой компоненты
   for (a in 1:A) {

      # находим столбец с наибольшей дисперсией
      col.ind <- which.max(apply(X, 2, sd))

      # выбираем значения этого столбца как первое приближение вектора проекций
      t <- X[, col.ind, drop = FALSE]

      # цикл для вычисления компоненты (30 итераций)
```

```
      for (i in 1:30) {
         p <- cov(X, t)
         p <- p / sqrt(sum(p^2))
         t <- X %*% p
      }

      # вычитаем объясненную дисперсию из матрицы данных
      X <- X - tcrossprod(t, p)

      # сохраняем полученные значения t и p в виде столбцов матриц
      scores[, a] <- t
      loadings[, a] <- p
   }

   # возвращаем матрицы нагрузок и счетов
   return(list(scores = scores, loadings = loadings))
}
```

А теперь приведем код для расчета расстояний и построении графика:

```
# загружаем данные Люди из пакета mdatools
library(mdatools)
data(people)
X <- people

# задаем число компонент
A <- 2

# центрируем и шкалируем матрицу с данными
X <- scale(X, center = TRUE, scale = TRUE)

# вычисляем счета и нагрузки используя функцию nipals
m <- nipals(X, A = A)

# вычисляем матрицу E
E <- X - tcrossprod(m$scores, m$loadings)
```

```
# вычисляем расстояние до модели
q <- rowSums(E^2)

# вычисляем дисперсию счетов
lambda <- colSums(m$scores^2) / (nrow(X) - 1)

# шкалируем квадраты счетов и вычисляем расстояние h
h <- rowSums(scale(m$scores^2, center = FALSE, scale = lambda))

# строим график расстояний
plot(h, q)
```

Часто величины расстояний h и q предварительно нормируют, разделив их на средние значения, h_0 и q_0 соответственно. Рисунок 2.27 показывает графики расстояний для МГК разложения данных *Люди*, сделанных с 2 (график слева) и 4 (график справа) компонентами.

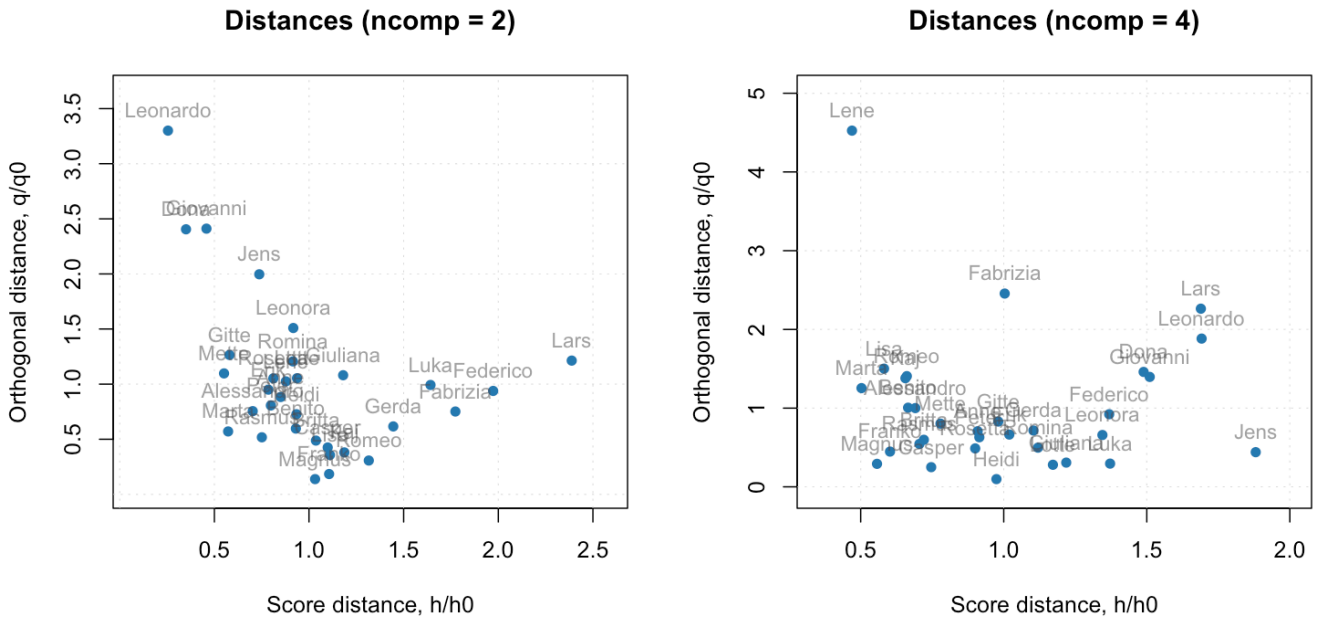


Рис. 2.27. Графики расстояния, построенные для МГК декомпозиции набора данных *Люди* с 2 компонентами (слева) и с 4 компонентами (справа).

Первый график показывает, что объект Lars обладает наиболее экстремальным значением, в то время как Leonardo имеет наибольшее расстояние до модели. Нужно вспомнить, что первые две компоненты объясняют вариацию данных, связанную с полом и регионом проживания. Т.е. вариация, связанная с возрастом, доходом и IQ осталась необъясненной и, соответственно, она вносит вклад в величину

расстояния до модели.

Собственно это и объясняет почему Leonardo находится дальше других — во-первых, он самый возрастной в выборке, его возраст 55 лет (на 5 лет старше следующего по возрасту, Giovanni). Во-вторых, у Leonardo длинные волосы и третий по величине доход. Все это в совокупности определяет относительно большую величину расстояния до модели.

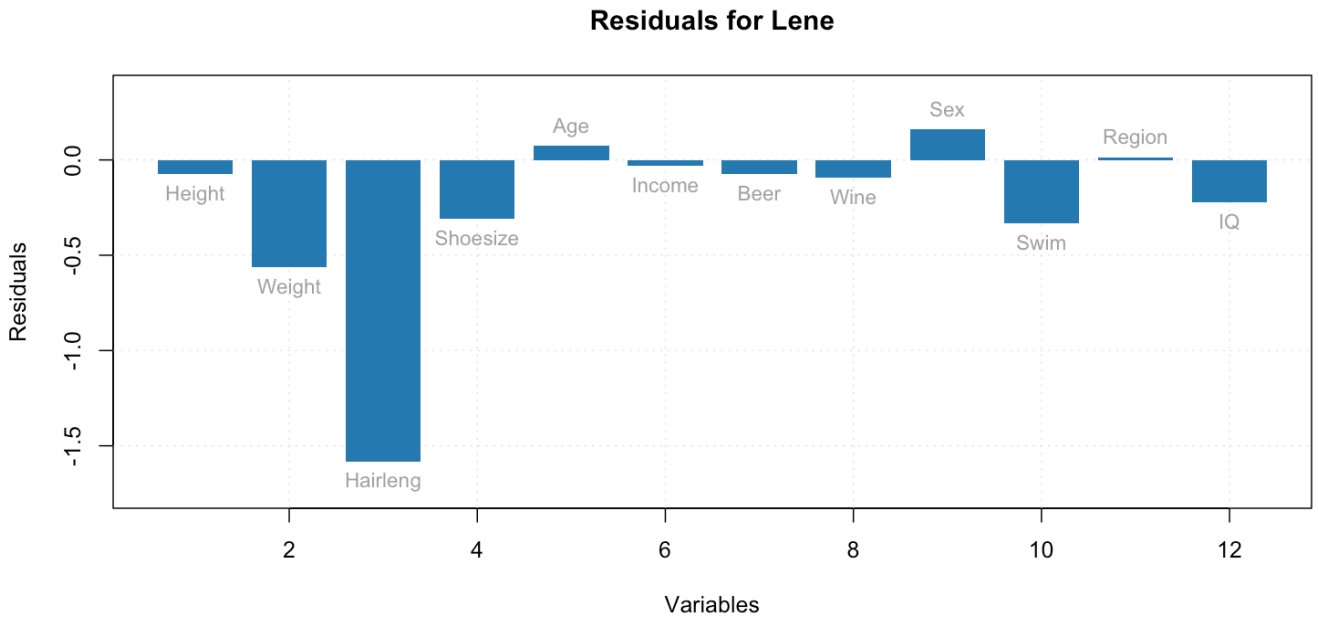


Рис. 2.28. График значений матрицы остатков, E, для строки, соответствующей человеку с именем Lene.

Заметим, что если использовать 4 компоненты, то Leonardo уже неплохо вписывается в тренд, а наибольшее расстояние до модели имеет Lene. Для того, чтобы определить причину этого, посмотрим на значения матрицы остатков **E** для Lene, показанную в виде столбчатой диаграммы на рисунке 2.28.

Как можно видеть из этого графика, наибольшее расхождение вносит переменная *Hairlength* (длина волос), так как Lene — единственная женщина в данной выборке с короткими волосами. Кроме этого, ее вес (47 кг при росте 166 см) также сильно отличается от общего тренда в меньшую сторону. Эти два фактора в совокупности и дают относительно большое расстояние до модели.

2.7.3 Критические значения для расстояний

Как определить допустимые значения для расстояний q и h? С какой величины можно считать наблюдение экстремальным, или выбросом? Очевидно, что оба расстояния — это статистики, которые описываются некоторым распределением. Если мы знаем вид этого распределения и сможем оценить его параметры, то это даст возможность определить так называемые критические значения.

Оба расстояния вычисляются как сумма квадратов значений координат, или их разницы. Если считать, что эти значения — случайные величины, которые распределены нормально, то сумма квадратов таких величин хорошо описывается распределением хи-квадрат. Но только в том случае, если эти значения стандартизованы (т.е. имеют нулевое среднее и единичную дисперсию).

Распределение хи-квадрат имеет только один параметр — число степеней свободы (англ. *degrees of freedom*). Если значения координат независимы, то число степеней свободы равно числу координат (т.е. числу переменных в случае q и числу компонент для h) минус единица. Однако, если данные не полностью случайны, то наличие внутренней структуры существенно снижает число степеней свободы. Причем его точное значение зависит от самих данных.

Таким образом, если найти, как шкалировать величины q и h и определить число степеней свободы, то это даст нам возможность использовать распределение хи-квадрат. В простейшем случае число степеней свободы определяется следующим образом:

$$N_q = \text{int}\left(\frac{2q_0^2}{s_q^2}\right)$$

$$N_h = \text{int}\left(\frac{2h_0^2}{s_h^2}\right)$$

Здесь q_0 h_0 — средние значения расстояний, а s_q^2 и s_h^2 их дисперсия. В этом случае расстояния, шкалированные следующим образом:

$$N_q \frac{q}{q_0}$$

$$N_h \frac{h}{h_0}$$

хорошо описываются распределением хи-квадрат с N_q и N_h числом степеней свободы, соответственно. А значит, критическое значение для каждого расстояния можно определить с помощью обратной функции распределения:

$$\chi^{-2}(1-\alpha, N)$$

Здесь α — это уровень значимости, он определяет процент наблюдений, которые будут определены как экстремальные.

Проблема в том, что расстояний два, соответственно для того, чтобы наблюдение не считалось экстремальным, необходимо выполнение двух условий одновременно:

$$N_q \frac{q}{q_0} < \chi^{-2}(1-\alpha, N_q)$$

$$N_h \frac{h}{h_0} < \chi^{-2}(1-\alpha, N_h)$$

Однако, в этом случае мы сталкиваемся с проблемой множественного сравнения. При уровне значимости $\alpha = 0.05$, 95% всех наблюдений (теоретически) будут иметь расстояния q меньше критического. И 95% всех наблюдений будут иметь расстояния h меньше критического. Однако, только 90% ($0.95 \times 0.95 \approx 0.90$) наблюдений будут иметь оба расстояния меньше критического. Т.е. 10% наблюдений будет определено как экстремальные, вместо ожидаемых 5%.

Для того, чтобы решить эту проблему, критическое значение следует считать для комбинации этих двух расстояний:

$$f = N_h \frac{h}{h_0} + N_q \frac{q}{q_0}$$

Эта величина, называемая полным расстоянием (англ. *full distance*), также описывается распределением хи-квадрат с $N_f = N_q + N_h$ степенями свободы. Такое значение геометрически можно представить в виде линии на графике нормированных расстояний q/q_0 от h/h_0. Наклон такой линии будет равен отношению степеней свободы, $-N_h/N_q$, а пересечение с осью y (т.е. значение q/q_0 когда $h = 0$) равно f_{crit}/N_q, где f_{crit} — критическое значение для полного расстояния, вычисляемое как:

$$f_{crit} = \chi^{-2}(1-\alpha, N_q + N_h)$$

Блок кода ниже показывает, как сделать это на практике. Здесь мы подразумеваем, что вы уже вычислили расстояния q и h, воспользовавшись предыдущим блоком.

```
# задаем уровень значимости
alpha <- 0.05

# вычисляем среднее значение
q0 <- mean(q)
h0 <- mean(h)

# вычисляем число степеней свободы
Nq <- round( (2 * q0^2) / var(q) )
Nh <- round( (2 * h0^2) / var(h) )
```

```
# вычисляем критическое расстояния для f
f.crit <- qchisq(1 - alpha, Nq + Nh)

# вычисляем какие наблюдения имеют полное расстояние больше критического
f <- Nq * q / q0 + Nh * h / h0
ext.ind <- which(f >= f.crit)

# показываем график расстояний для всех наблюдений
plot(h/h0, q/q0, xlim = c(0, 4), ylim = c(0, 8))

# вычисляем параметры границы как линии (наклон и пересечение с осью y)
b <- -Nh/Nq
a <- f.crit / Nq

# рисуем линию
abline(a = a, b = b, lty = 2)

# показываем наблюдения, для которых f >= f.crit красным цветом
points(h[ext.ind]/h0, q[ext.ind]/q0, col = "red", pch = 16)
```

Рисунок 2.29 показывает два графика расстояний, идентичных изображенным на рисунке 2.27, но с двумя границами. Первая, расположенная ближе к точкам, — это граница для экстремальных наблюдений, вычисление которой мы рассмотрели выше. Вторая — это граница для определения выбросов — наблюдений, которые сильно отличаются от остальных и, следовательно, требуют специального внимания. Нужно найти причину такого поведения и, если это на самом деле выбросы, удалить их из выборки и пересчитать МГК модель заново уже без них.

Так, график сделанный для модели с двумя главными компонентами (слева), показывает только одно наблюдение, определенное, как экстремальное — уже известный нам Lars. Тогда как в случае с четырьмя компонентами таких наблюдений три — Lars, Leonardo и Jens. Теоретически, число экстремальных наблюдений должно быть между 1 и 2 (мы используем уровень значимости, $\alpha = 0.05$, и, так как общее число наблюдений $n = 32$, теоретически, $32 \times 0.05 = 1.6$ наблюдений будет определено как экстремальные). Но это работает только если выборка достаточно большая, в случае с небольшими выборками реальные значения могут отличаться от теоретических.

Нужно отметить, что изложенный выше способ построения графиков расстояний и определения критических значений — не единственный. Так, например, если известно, что в данных имеются выбросы, то можно воспользоваться робастным способом, основанным на использовании непараметрических

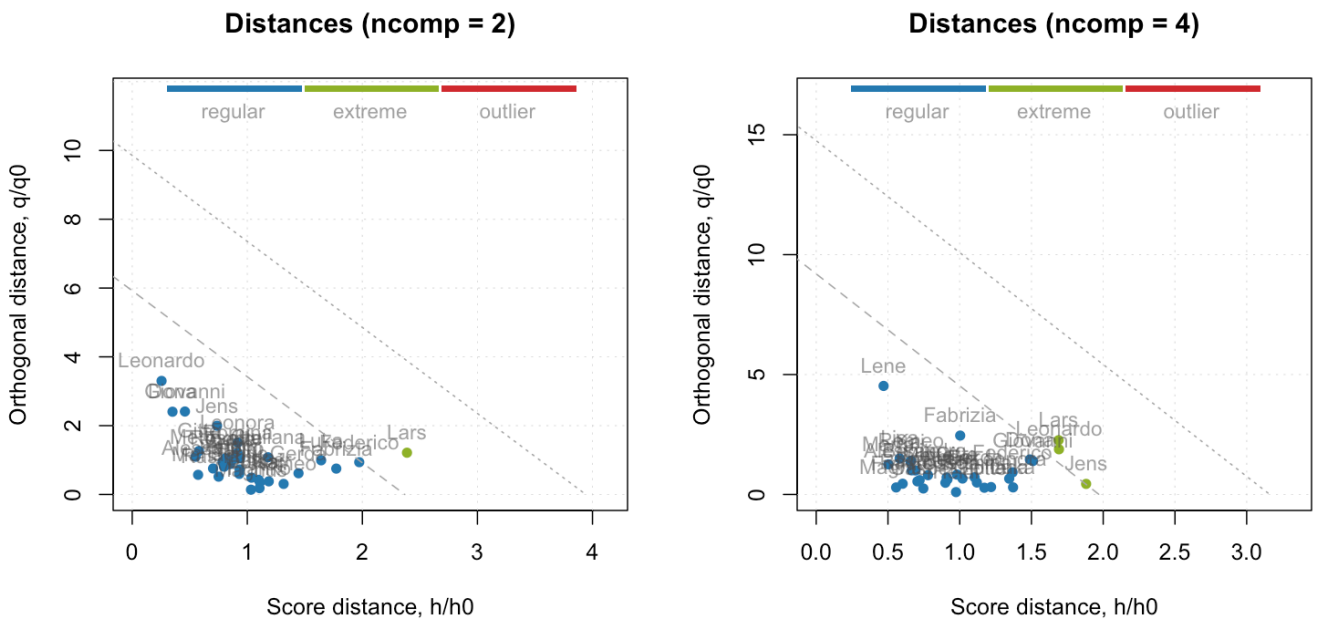


Рис. 2.29. График расстояний с границами для экстремальных образцов и выбросов

оценок центра и разброса значений расстояний.

Существуют также методы, которые задают критические значения отдельно для каждого расстояния. В этом случае граница области “нормальных” наблюдений на графике расстояний будет представлять собой четырехугольник, а не треугольник, как в нашем случае. Обычно, метод запрограммирован в ПО, которое используется для анализа, поэтому возможность писать код для анализа самостоятельно, позволяет применять самые последние наработки в этой области и экспериментировать, предоставляя исследователю гораздо больше возможностей.

2.8 Другие способы вычисления МГК

Алгоритм NIPALS, подробно рассмотренный нами выше, является не единственным, с помощью которого можно разложить данные на линейную комбинацию взаимно ортогональных главных компонент. Более того, он не является и самым эффективным, особенно если нужно получить не 1-2 компоненты, а больше. Наиболее распространен другой алгоритм, который называется разложением по сингулярным значениям (англ. *singular values decomposition, SVD*), или просто *сингулярное разложение.*

Сингулярное разложение представляет исходную матрицу **X** (как и в случае с NIPALS обычно предварительно центрированную, так чтобы среднее значение каждого столбца этой матрицы было равно нулю) в виде произведения трех матриц:

$$\mathbf{X} = \mathbf{UDV}^T$$

Столбцы матриц **U** и **V** содержат, соответственно, левые и правые сингулярные вектора, а матрица **D** — это диагональная матрица, которая содержат сингулярные значения на главной диагонали. Так вот, между результатами МГК и результатами сингулярного разложения есть прямая зависимость. Если сделать МГК (например, с помощью алгоритма NIPALS) для максимально возможного числа компонент, то матрица остатков **E** будет иметь нулевые значения:

$$\mathbf{X} = \mathbf{TP}^T$$

Тогда матрица нагрузок **P** будет идентична матрице с правыми сингулярными векторами **V**, а матрица счетов **T** будет равна произведению матрицы с левыми сингулярными векторами **U** на диагональную матрицу **D**.

Мы можем легко показать это с помощью простого кода на R. Сингулярное разложение реализовано в R с помощью метода `svd()`. Этот метод возвращает результаты в виде списка с тремя элементами. Код ниже показывает, как сгенерировать матрицу **X** с 5 строками и 3 столбцами со случайными значениями из нормального распределения, и сделать ее сингулярное разложение. Строка `set.seed(42)` используется для того, чтобы генератор случайных чисел возвращал всегда одни и те же значения, чтобы когда вы выполните этот код у себя на компьютере, ваш результат совпал с показанным ниже.

```
# генерируем матрицу X со случайными значениями
set.seed(42)
X <- people[1:5, 1:3]

# центрируем и шкалируем столбцы матрицы
X <- scale(X, center = TRUE, scale = TRUE)

# делаем сингулярное разложение матрицы
m1 <- svd(X)
```

Давайте посмотрим на результаты разложения:

```
# показать матрицу с левыми сингулярными векторами
show(m1$u)
```

```
          [,1]        [,2]        [,3]
```

```
[1,] -0.5765778 -0.18699149 -0.65677058
[2,] -0.2499999  0.07352438  0.46446118
[3,] -0.2218810  0.09677098  0.50906090
[4,]  0.4685781  0.69759561 -0.30605693
[5,]  0.5798806 -0.68089947 -0.01069456
```

```r
# показать матрицу с правыми сингулярными векторами
show(m1$v)
```

```
           [,1]       [,2]        [,3]
[1,] -0.6429102 -0.2670163 -0.71789185
[2,] -0.6313075 -0.3460101  0.69406613
[3,]  0.4337248 -0.8994328 -0.05388406
```

```r
# показать сингулярные значения
show(m1$d)
```

```
[1] 2.9895741 1.6927752 0.4438005
```

Как можно видеть, всего получилось три сингулярных вектора (левых и правых) и три сингулярных значения, что и было ожидаемо, так как размерность матрицы 5×3 и мы не можем разложить ее на более чем три компоненты.

Теперь применим нашу функцию `nipals` чтобы найти счета и нагрузки МГК разложения нашей матрицы с данными и покажем их значения:

```r
# делаем МГК разложение с помощью NIPALS
m2 <- nipals(X, 3)

# показать матрицу нагрузок
show(m2$loadings)
```

```
           [,1]      [,2]        [,3]
[1,]  0.6429102 0.2670163  0.71789185
[2,]  0.6313075 0.3460101 -0.69406613
[3,] -0.4337248 0.8994328  0.05388406
```

```
# показать матрицу счетов
show(m2$scores)
```

```
           [,1]       [,2]         [,3]
[1,]  1.7237220  0.3165346  0.291475090
[2,]  0.7473931 -0.1244602 -0.206128088
[3,]  0.6633298 -0.1638115 -0.225921464
[4,] -1.4008488 -1.1808726  0.135828210
[5,] -1.7335961  1.1526098  0.004746252
```

Обратите внимание, что значения матрицы нагрузок **P** идентичны значениям матрицы с правыми сингулярными векторами **V** с точностью до знака. И та и другая матрица задают ориентацию главных компонент в пространстве. Разные знаки означают, что компоненты имеют разное направление, но ориентация, или положение компонент в пространстве — идентичны.

Т.е. для нахождения МГК нагрузок можно просто применить SVD, матрица **V**, полученная в результате и будет матрицей МГК нагрузок. Если нужно сделать не полное разложение, а использовать только несколько первых компонент, нужно просто выбрать нужные столбцы из матрицы **V** (например, первые два), а остальные — отбросить.

Далее, найдем соответствие между счетами **T** и левыми сингулярными векторами **U**. Как мы уже упомянули выше, левые векторы, которые находит SVD, имеют единичную длинну. А их оригинальная длина, которая соответствует разбросу данных, это как раз те самые сингулярные значения из матрицы **D**. С другой стороны, матрица счетов **T** содержит координаты левых сингулярных векторов оригинального размера. Другими словами, если мы вычислим длину векторов, которые заданы столбцами матрицы **T**, мы должны получить сингулярные значения.

Из главы 2 мы знаем, что длину вектора можно вычислить, взяв квадратный корень из суммы квадратов его координат (по правилу Пифагора). Проверим это:

```
# показать длинну векторов, которые заданы сначениями МГК счетов
show(sqrt(colSums(m2$scores^2)))
```

```
[1] 2.9895741 1.6927752 0.4438005
```

Как вы видите, мы получили в точности сингулярные значения. Но это дает нам возможность получить счета — нужно просто взять левые сингулярные вектора из матрицы **U** и шкалировать их, умножив их координаты на сингулярные значения. Т.е.

$$\mathbf{T} = \mathbf{UD}$$

Проверим это на практике:

```
show(m1$u %*% diag(m1$d))
```

```
           [,1]       [,2]         [,3]
[1,] -1.7237220 -0.3165346 -0.291475090
[2,] -0.7473931  0.1244602  0.206128088
[3,] -0.6633298  0.1638115  0.225921464
[4,]  1.4008488  1.1808726 -0.135828210
[5,]  1.7335961 -1.1526098 -0.004746252
```

Если сравнить полученные значения со значениями из матрицы счетов, то мы получим точное соответствие (с точностью до знака). Таким образом, МГК в R можно легко реализовать через встроенный метод сингулярного разложения. Код ниже показывает, как это сделать по тому же принципу, что и метод `nipals` реализованный нами ранее.

```
pcasvd <- function(X, A = 1) {

   # центрируем и шкалируем данные
   X <- scale(X, center = TRUE, scale = TRUE)

   # вычисляем полное сингулярное разложение
   m <- svd(X)

   # выбираем первые A правых векторов в качестве матриц нагрузок
   loadings <- m$v[, 1:A]

   # выбираем первые A левых векторов и шкалирем их на собственные значение
   # чтобы получить матрицу счетов
   scores <- m$u[, 1:A] %*% diag(m$d[1:A], A, A)

   # возвращаем матрицы нагрузок и счетов
   return(list(scores = scores, loadings = loadings))
}
```

Как можно видеть, новая реализация получилась гораздо проще и не требует цикла. Следует лишь заметить, что шкалирование не всегда обязательно и его имеет смысл сделать опциональным. Оставляем это как небольшое упражнение для вас.

2.9 МГК на практике

Если собрать все блоки кода, приведенные в этой главе, то вы без труда сможете провести МГК декомпозицию любых данных, получить основные результаты (матрицы счетов, нагрузок и остатков, а также векторы расстояний и значения объясненной и остаточной дисперсии для каждой компоненты), и визуализировать их с помощью соответствующих графиков. Однако, на практике есть много нюансов, которые потребуют от вас писать дополнительный код. Если вы не против — то это будет хорошим решением и позволит лучше изучить все нюансы описанного метода, а также "прокачать" свои навыки в программировании. Однако, когда вы понимаете, как работает метод, то можно воспользоваться уже готовыми решениями и сэкономить время для непосредственно анализа и описания результатов.

В R есть несколько пакетов, позволяющих применять МГК и другие хемометрические методы для ваших данных, например, пакет *chemometrics* австрийских ученых Peter Filzmoser and Kurt Varmuza. Большинство примеров в этой книге сделаны с помощью другого пакета, *mdatools*, который разработан одним из авторов этой книги. Пакет имеет подробное руководство для пользователя, доступное по адресу https://mda.tools/docs/. В этом разделе мы очень коротко коснемся того, как сделать МГК анализ с помощью этого пакета.

Для создания МГК модели необходимо воспользоваться функцией `pca()`. Она имеет один обязательный аргумент — фрейм, или матрицу с данными. Однако, вы можете использовать дополнительные аргументы, чтобы задать такие параметры модели, как число компонент, необходимость шкалирования данных, уровень значимости для определения экстремальных образцов и выбросов, и многое другое. Пример кода ниже показывает, как построить МГК модель для набора данных *Люди* с четырьмя главными компонентами и шкалированием.

```
library(mdatools)

# загрузить данные People в окружение
data(people)

m <- pca(people, 4, scale = TRUE)
```

Набор данных *Люди* поставляется вместе с пакетом, поэтому его можно легко загрузить с помощью функции `data()`. Чтобы посмотреть основные параметры модели, а именно собственные значения и объясненную дисперсию компонент, нужно воспользоваться стандартным методом `summary()`:

```
summary(m)
```

Объект `m` — это список, который содержит все результаты вычисления. Так, например, `m$loadings` содержит матрицу нагрузок, `m$eigenvals` — собственные значения компонент, а `m$Qlim` — матрицу с параметрами для вычисления критических значений для ортогональных расстояний.

Проекции точек на модель называются результатами и собраны вместе в другом списке `m$res`. Внутри этого списка могут быть два объекта: `m$res$cal` — результаты МГК разложения для калибровочного (тренировочного набора) и `m$res$test` — результаты проекции тестового набора, если он был использован. Например `m$res$cal$scores` содержит матрицу счетов для объектов из калибровочного набора, а `m$res$cal$expvar` — вектор с объясненной дисперсией.

Для того, чтобы применить МГК модель к новому набору данных нужно воспользоваться методом `predict()`:

```
res <- predict(m, Xnew)
```

В этом случае объект `res` будет содержать в себе все результаты и по структуре идентичен объектам `m$res$cal` и `m$res$test`.

Для того, чтобы визуализировать результаты с помощью графиков, можно воспользоваться большим набором различных функций. Эти функции можно использовать как с объектом, содержащим МГК модель, так и с объектами, содержащими результаты МГК разложения. Вот так, например, можно показать график счетов для модели:

```
# график счетов для ГК1 и ГК2
plotScores(m)

# график счетов для ГК1 и ГК3
plotScores(m, c(1, 3))

# график счетов для ГК1 в виде столбчатой диаграммы
plotScores(m, 1, type = "h")
```

Графические функции обладают рядом специальных возможностей, например, можно легко группировать точки и линии с помощью цветовых градиентов, строить доверительные эллипсы и границы кластеров точек и многое другое. Полный список функций и аргументов можно найти в руководстве пользователя, или в справке (`?pca`).

3 Предварительная обработка данных

3.1 Искажения в аналитических данных

За несколько десятков лет активного использования хемометрических алгоритмов сформировалась методология, подсказывающая какие этапы нужно пройти для успешного моделирования данных. Причем эта методология довольно универсальна и не зависит от типа данных. К примеру, анализ БИК (ближний инфракрасный) и ЯМР (ядерно-магнитный резонанс) спектров подразумевает один и тот же набор шагов (правда, используемые подходы на некоторых из них могут отличаться). Предварительная обработка вместе с визуализацией данных — это, зачастую, один из первых шагов, который может очень сильно повлиять на конечный результат.

Предпосылкой успешного хемометрического анализа инструментальных данных является получение воспроизводимых сигналов для всех измерений в серии. Этого можно добиться экспериментально с использованием стандартизированной процедуры пробоподготовки и измерений. Например, особое внимание для растворенных в воде образцов должно быть уделено постоянству pH, иначе в результатах измерений будет содержаться дополнительный “шум” от непостоянства pH. Это может отрицательно сказаться на качестве последующего многомерного моделирования.

Для получения воспроизводимого результата нужно, чтобы условия измерения аналитического сигнала были стандартизированы как можно сильнее. Но это, к сожалению, не всегда возможно. К примеру, интенсивность сигналов в Рамановском спектре при работе с выносными оптоволоконными зондами сильно зависит от положения зонда. Сдвиг даже на долю миллиметра может повлиять на их интенсивность. Вышеописанные проблемы можно решить либо используя дорогие устройства, с точным позиционированием, или контролем pH растворов, либо исправлять эти эффекты математически с помощью методов предварительной обработки, о которых речь пойдет в этой главе.

Кроме того, важным является выбор параметров для регистрации сигналов. Так, в случае ЯМР спектроскопии, применение импульсных программ с подавлением сигнала растворителя может повысить чувствительность измерений в несколько раз по сравнению со стандартным измерением (рисунок 3.1). Известны и более сложные импульсные программы одновременного подавления нескольких сигналов в смешанных растворителях, что используется, в частности, для ЯМР анализа алкогольных напитков.

В процессе хемометрического моделирования после регистрации спектров первым шагом является предварительная обработка данных, которая обычно состоит из нескольких шагов (например, коррекция базовой линии, фазы и рассеяния света, устранения шумов, а также ряд простейших математических операций). Кроме того, перед непосредственным построением хемометрической модели обязателен разведочный анализ, с которым Вы познакомились в главе 2. Только после этого исследователи строят многомерные модели для качественного, или количественного анализа. Важным этапом является расчет метрологических характеристик методики в целом. Обычно выполняется расчет тех же характеристик, что и для методик, основанных на одномерном анализе данных (чувствительность, селективность, предел обнаружения и др.). В случае многомерного моделирования алгоритм их расчета отличается от стандартного подхода, и в настоящем учебнике не описывается.

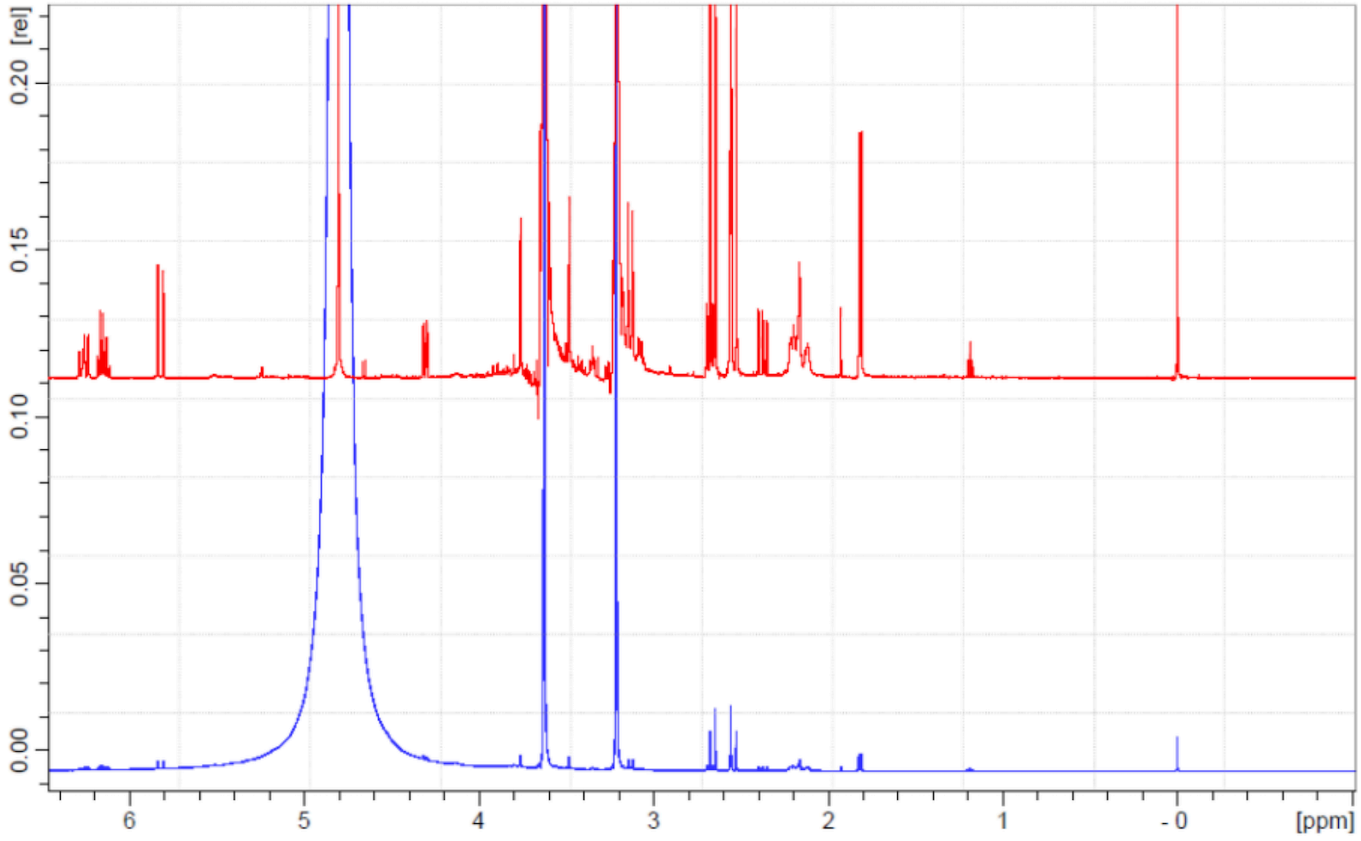


Рис. 3.1. ЯМР 1Н спектр водного экстракта препарата Алоэ веры с (А) и без использования (Б) импульсной программы подавления сигнала воды.

Предварительная обработка данных, о которой речь пойдет в данной главе, является очень важным этапом многомерного анализа. Она напрямую влияет на результаты разведочного анализа, результаты работы классификационных и регрессионных моделей. В общем случае предварительная обработка направлена на устранение нежелательных искажений в экспериментальных данных.

Стратегия выбора оптимального подхода к предварительной обработке из имеющегося арсенала

доступных алгоритмов зависит от многих факторов и должна базироваться на совокупности знаний о решаемой химической проблеме, свойствах набора данных и планируемом для использования методе хемометрического моделирования.

Предварительная обработка данных обычно состоит из нескольких этапов, каждый из которых направлен исправление одного конкретного артефакта. В качестве примера, на рисунке 3.2 показана типичная последовательность предварительной обработки для ИК данных. Обратите внимание, что здесь представлен только один из многих возможных сценариев предварительной обработки ИК спектров – изменение порядка этапов предварительной обработки может повлиять как окончательный вид спектров, так и результаты хемометрического моделирования.

Рис. 3.2. Возможная стратегия предварительной обработки ИК спектроскопических данных. Отдельные этапы могут быть пропущены, или изменены в зависимости от свойств данных и цели исследования.

Цель данной главы познакомить читателя с наиболее часто встречающимися артефактами, характерными для сигналов различных аналитических методов. Многие описанные в данной главе дефекты могут присутствовать в данных нескольких аналитических методов. В таблице 3.1 представлен обзор наиболее распространенных артефактов для различных инструментальных методов исследования. В частности, предварительная обработка включает в себя коррекцию базовой линии и фазы сигналов, устранения эффекта рассеяния света, сглаживание данных, выравнивание сигналов по оси переменных (химические

сдвиги, волновое число и др.), и др. Обратите внимание, что разные физические явления, относящиеся к различным аналитическим методам, могут вызвать очень похожие артефакты в экспериментальных данных и, поэтому, могут быть устранены одним и тем же методом предварительной обработки. Например, шум присутствует во всех типах экспериментальных сигналов, а рассеяние света, зависящее от соотношения размера частиц исследуемой матрицы и длины волны излучения, характерно, в первую очередь, для сигналов флуоресценции и колебательных методов.

Таблица. 3.1. Артефакты, встречающиеся в данных различных инструментальных методов исследования. Здесь молекулярная спектроскопия включает в себя спектроскопию в ультрафиолетовом, ближнем и среднем инфракрасном спектра, а также Рамановскую спектроскопию. Сокращения: ЯМР - ядерно-магнитный резонанс, ХФ - хроматография, ЭФ - электрофорез, МС - масс-спектроскопия.

Метод	Молекулярная спектроскопия	ЯМР	ХФ	ЭФ	МС	Флуоресценция
Фаза	-	x	-	-	-	-
Базовая линия	x	x	x	x	x	x
Выравнивание	-	x	x	x	x	-
Рассеяние	x	-	-	-	-	x
Шум	x	x	x	x	x	x

Кроме того, предварительная обработка может быть отнесена не к определенному инструментальному методу, выбранному исследователем, а обусловлена различием в характеристиках образцов или важности переменных для конкретной многомерной модели. Для коррекции артефактов данного типа применяют центрирование, шкалирование, нелинейные преобразования и нормирование. К предварительной обработке часто относят и другие виды преобразования данных, например, обнаружение выбросов, выбор переменных, деконволюцию сигналов и обращение с пропущенными значениями. Однако, поскольку эти манипуляции сильно связаны с непосредственным анализом данных, в рамках данной главы они не обсуждаются.

3.2 Простейшие математические преобразования

В процессе предварительной обработки исследователи работают либо со столбцами (переменными), либо со строками (образцами) в матрице экспериментальных данных. В первом случае методы предварительной обработки направлены на нивелирование различий между порядком величин, в которых выражены сигналы (или концентрации) отдельных аналитов. Например, средние концентрации соединений в биологических исследованиях значительно меньше концентраций макрокомпонентов (например, АТФ). Однако, метаболиты с большими концентрациями не обязательно более важны для модели, чем микрокомпоненты. Классическим примером преобразования данных в этом случае является шкалирование. С другой стороны, нормирование (преобразование по строкам) позволяет проводить

сравнение в серии образцов, используя один из факторов нормализации, например, постоянная сумма, референтный сигнал или определенное свойство образца (содержание сухого вещества или навеска образца).

3.2.1 Нормирование

Целью нормирования (нормализации) является удаление различий в данных, вызванных экспериментальными факторами, так чтобы при этом не затронуть важную для моделирования информацию о системе (например, различие в ботаническом происхождении образцов). Нормализацию следует, например, использовать в случае различий в навеске образцов. Так, можно рассмотреть ЯМР спектры мочи, в которых площадь сигнала напрямую связана с содержанием соединения в образце. При этом концентрация соединения зависит от содержания воды в моче, которое, в свою очередь зависит от множества других факторов: диета, водный режим, возраст и т.д., что делает некорректным использование абсолютных концентраций соединений для сравнения образцов. Подобные «неинтересные» различия необходимо устранять с использованием нормирования.

Известны две стратегии для нивелирования подобных систематических различий в данных. В первой из них используют внутренний стандарт, который добавляют во все образцы в процессе пробоподготовки. Для хроматографических измерений это соединение должно быть похоже по химических свойствам и хроматографическому поведению на большинство остальных компонентов смеси. Предварительная обработка заключается в делении максимума интенсивности (или площади) всех сигналов на данное значение для внутреннего стандарта. Например, в случае образцов мочи интенсивность каждой полосы выражают относительно сигнала креатинина, который выводится из организма с постоянной скоростью и служит своего рода внутренним стандартом. В других случаях используют фактор нормализации для каждого измерения в серии. В качестве фактора нормализации можно использовать единичную длину, интегральную интенсивность всего сигнала, или сумму интенсивностей всех переменных (рис. 3.3). К примеру, последний вариант нормирования математически выглядит следующим образом:

$$x'_{ij} = \frac{x_{ij}}{\sum x_{ij}}$$

Здесь и далее мы будем использовать обозначение x_{ij} для величины i-го сигнала (например, отдельного спектра) в исходном наборе данных в точке j (например, для конкретной длины волны). Через x'_{ij} мы обозначим величину этого сигнала в данной точке после обработки. Если используется только один индекс, то индекс i (например x_i) обозначает все значения конкретного сигнала, а индекс j обозначает значения всех сигналов в конкретной точке (для конкретной переменной, например, на определенной длине волны).

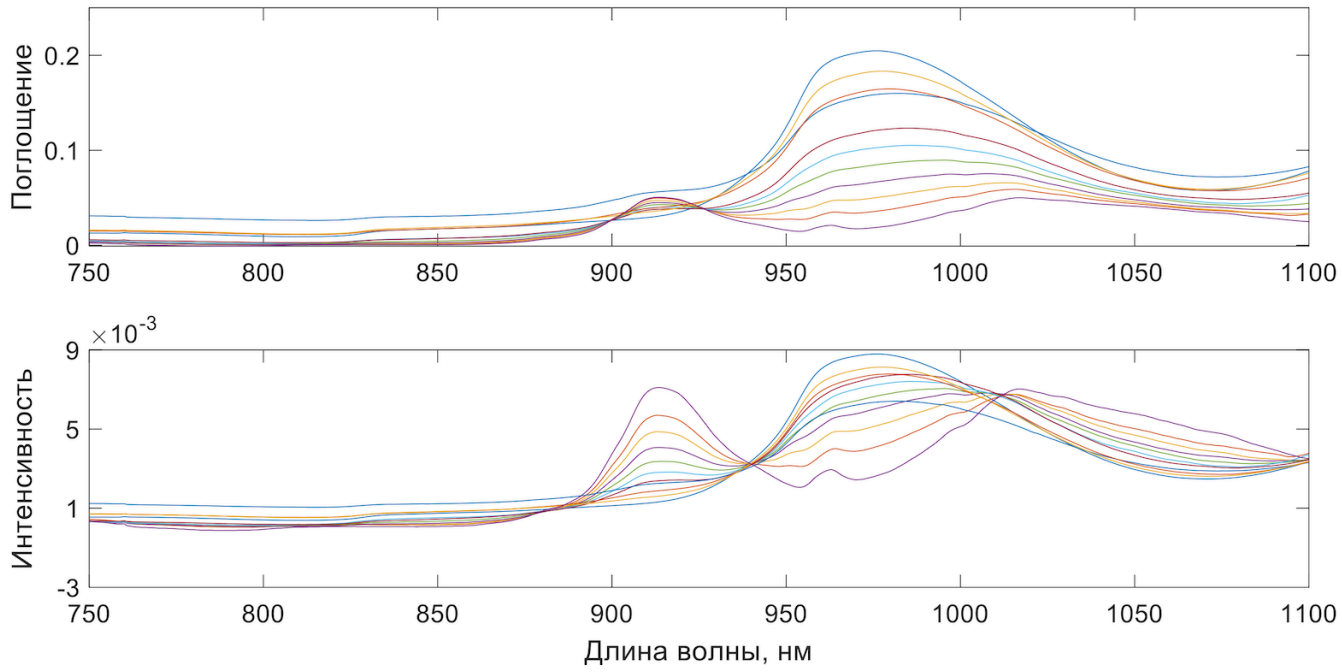


Рис. 3.3. Использование нормирования на суммарную интенсивность для серии БИК спектров смесей вода-пропанол: исходные данные (верхний рисунок) и нормированные данные (нижний рисунок). Интегральная интенсивность каждого профиля приравнена к единице.

3.2.2 Центрирование

Центрирование является одним из наиболее распространенных вариантов предварительной обработки данных. Оно удаляет общее смещение из каждого измеренного профиля и позволяет сфокусироваться на значимом различии между образцами. Для центрирования набора данных среднее значение каждого столбца (каждой переменной) вычитают из каждого элемента этого столбца. После использования процедуры центрирования интенсивность сигналов колеблется около нуля вместо среднего значения интенсивности для каждого сигнала (рис. 3.4). На рисунке 3.5 приведены результаты центрирования для БИК спектров смесей пропанол-вода.

3.2.3 Шкалирование

Очевидно, что даже при соблюдении стандартизированных пробоподготовки и измерений, не все переменные содержат существенную информацию в контексте решаемой аналитической задачи. Кроме того, «важные» переменные могут быть «скрыты» незначащими переменными, если они выражены в разных единицах измерения с резко отличающихся масштабом.

Следует заметить, что в большинстве инструментальных данных все переменные имеют одну и ту же размерность. Однако, иногда, например, в ЯМР-спектрометрических измерениях, интенсивность полос микро и макрокомпонентов может отличаться на несколько порядков. В подобных случаях для одной переменной дисперсия может составлять порядка нескольких тысяч, а для другой переменной - порядка

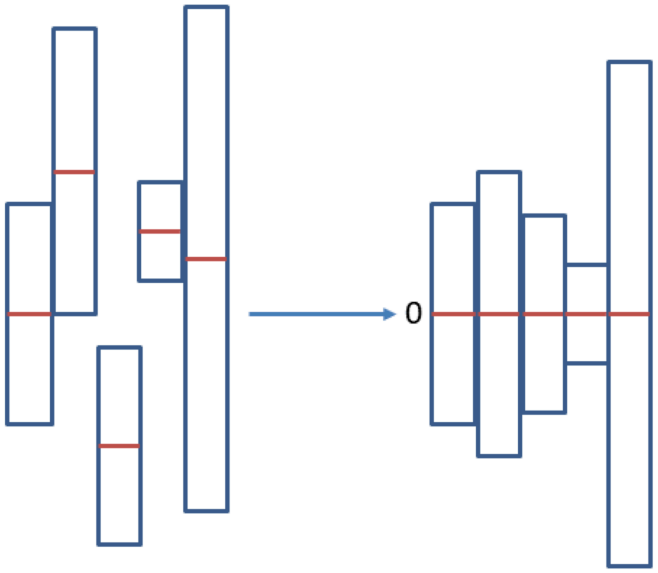


Рис. 3.4. Схематичное представление процедуры центрирования.

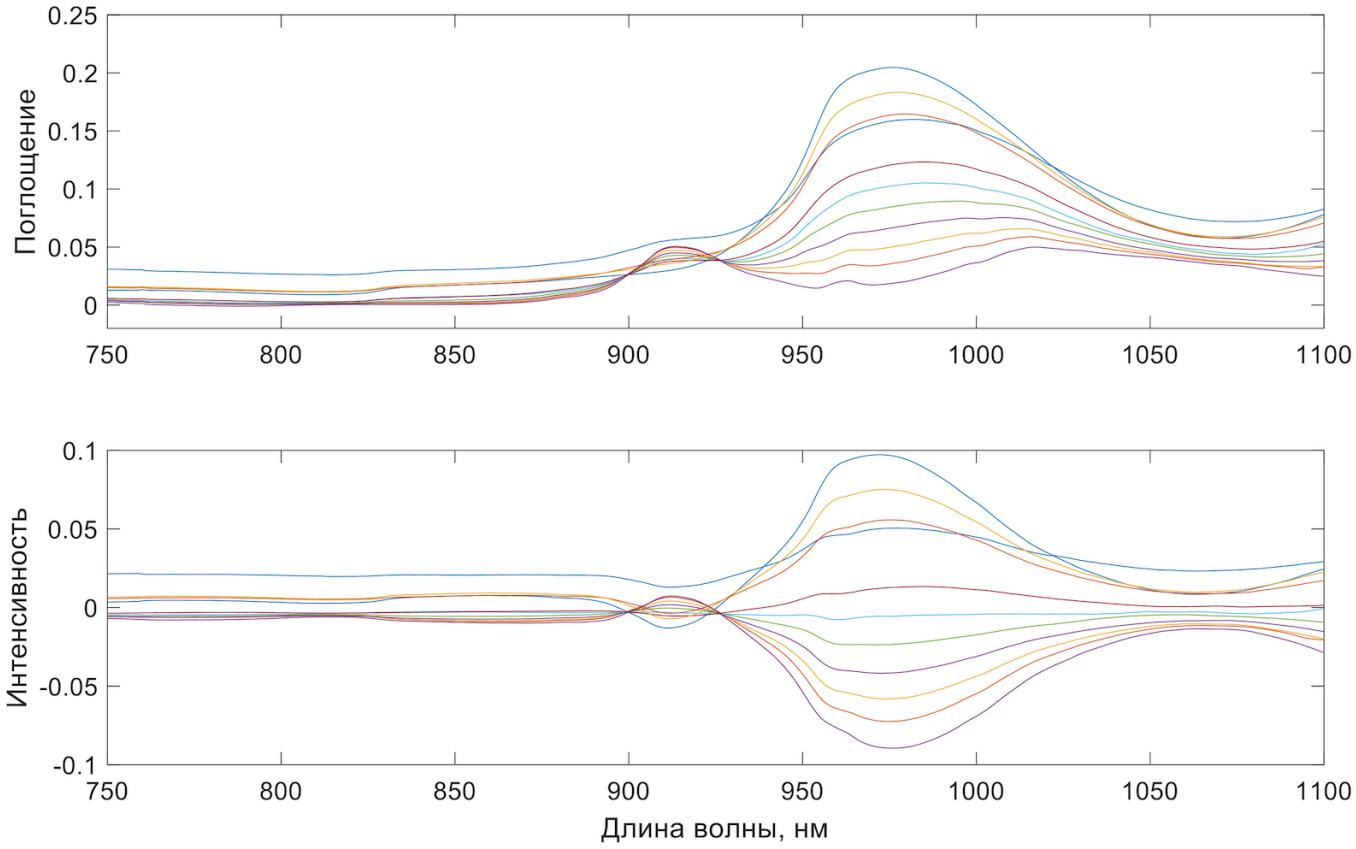


Рис. 3.5. Центрирование БИК спектров смесей вода-пропанол: исходные (верхний рисунок) и центрированные данные (нижний рисунок).

0,001. Представьте себе ситуацию, в которой одна переменная измеряется в килограммах, а другая - в миллиграммах.

Распространенным способом решения данной проблемы является шкалирование и нелинейные преобразования, которые используются для обработки разных видов данных для выравнивания вклада переменных в хемометрическую модель. Наиболее популярным способом шкалирования данных является *автошкалирование* (англ. *autoscaling*), уже рассмотренное нами выше в Главах 1 и 2. Автошкалирование заключается в вычитании среднего значения для данной переменной (по каждому столбцу j), $\bar{x}_j$ и делении на стандартное отклонение, s_j:

$$x'_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j}$$

Автошкалирование часто используют перед применением МГК, в котором переменные с наибольшей дисперсией априорно доминируют в модели (рис. 3.6). Недостаток автошкалирования проявляется в значительном увеличении шума в спектральных областях с малыми интенсивностями сигналов, вследствие деления на величины, близкие к нулю.

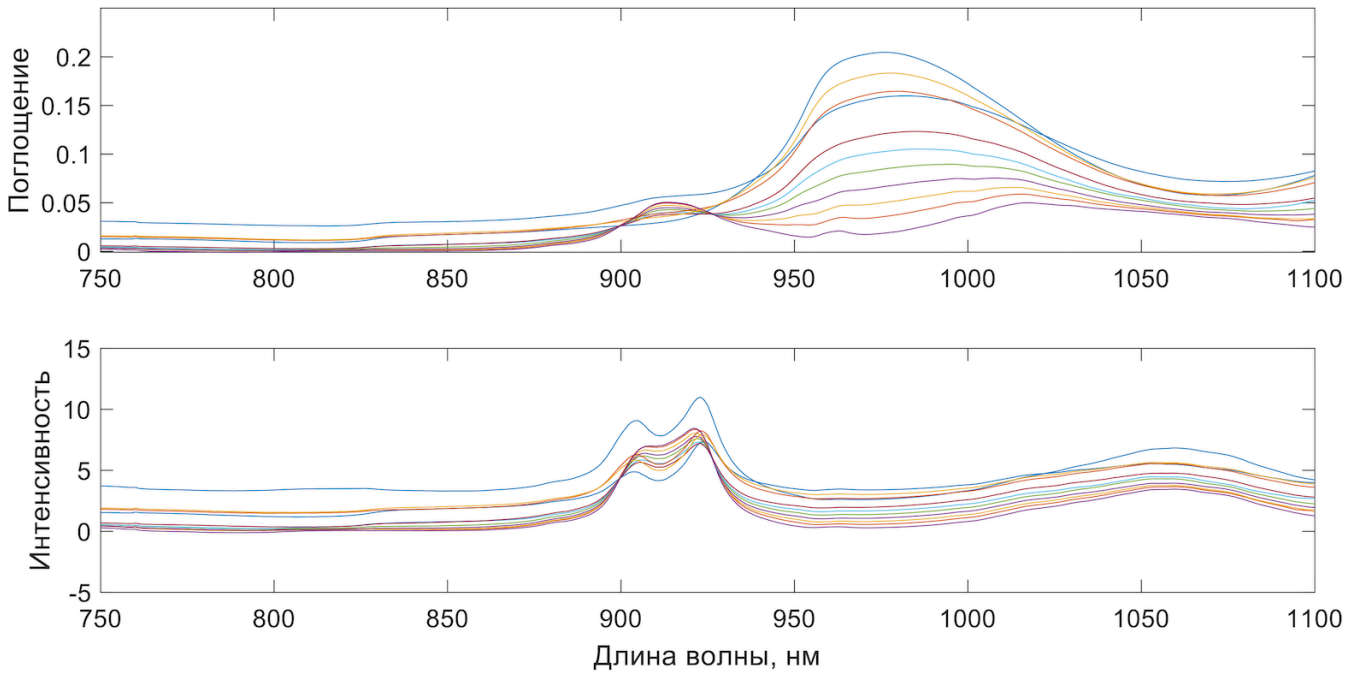


Рис. 3.6. Применение процедуры шкалирования для БИК спектров смесей пропанол-вода. Верхний рисунок — исходные данные, нижний — после шкалирования

Хорошей альтернативой автошкалированию является Парето-шкалирование, которое заключается в делении разницы измеренного сигнала и среднего значения интенсивности на корень из стандартного отклонения:

$$x'_{ij} = \frac{x_{ij} - \bar{x}_j}{\sqrt{s_j}}$$

Парето-шкалирование не полностью устраняет эффект вариации сигналов, но предлагает оптимальное соотношение вкладов сигналов с малой и большой амплитудой в суммарную модель. Парето-шкалирование не приводит к такому сильному увеличению влияния шума, как в случае автошкалирования. Кроме того, обработанные данные ближе к исходным, чем после автошкалирования.

Другими часто использующимися вариантами шкалирования является шкалирование диапазоном (англ. *range scaling*):

$$x'_{ij} = \frac{x_{ij} - \bar{x}_i}{\max(x_j) - \min(x_j)}$$

«обширное» шкалирование (англ. *vast scaling*):

$$x'_{ij} = \frac{(x_{ij} - \bar{x}_j)}{s_j} \frac{\bar{x}_j}{s_j}$$

и «уровневое» шкалирование (англ. *level scaling*):

$$x'_{ij} = \frac{x_{ij} - \bar{x}_j}{\bar{x}_j}$$

Цель любого из рассмотренных вариантов шкалирования заключается в выравнивании вклада различных переменных путем взвешивания каждой из них.

3.2.4 Нелинейные преобразования

Альтернативой шкалированию являются нелинейные «поэлементные» преобразования исходных данных. Подобные преобразования отличаются от шкалирования тем, что они изменяют отдельные элементы данных матрицы, а не целые переменные. Примеры таких преобразований включают силовую и логарифмическую трансформации, в процессе которых рассчитывается квадратный корень, или логарифм каждого отдельного элемента данных, соответственно:

$$x'_{ij} = \sqrt{x_{ij}}$$

$$x'_{ij} = \log(x_{ij})$$

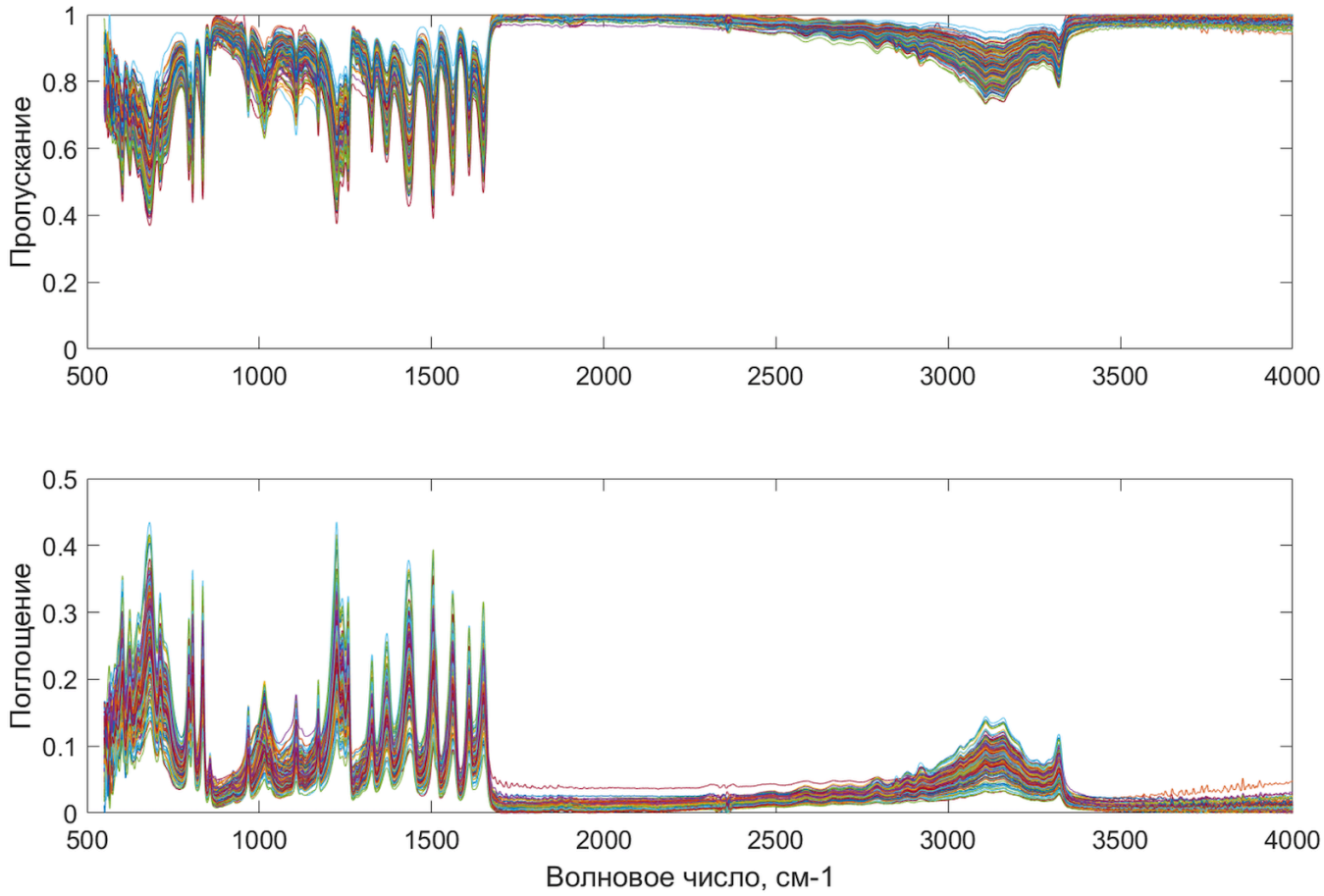


Рис. 3.7. Логарифмическая трансформация ИК из пропускания (верхний рисунок) в поглощение (нижний рисунок) на примере серии препаратов парацетамола.

Данные преобразования помогают уменьшить значимость наиболее интенсивных сигналов, что позволяет говорить об эффекте «псевдошкалирования». Ярким примером использования логарифмического преобразования является перевод интенсивностей ИК спектров из формата “на пропускание” в формат “на поглощение” (рис. 3.7).

3.2.5 Простейшие преобразования в R

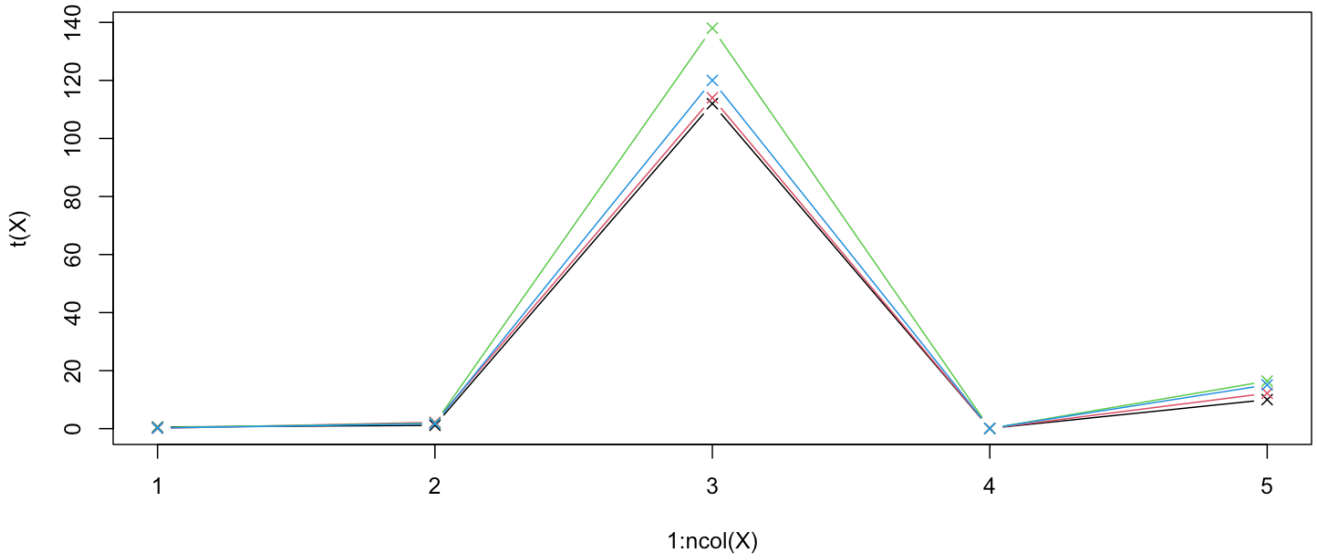
Для того, чтобы показать, как делать простые преобразования, описанные в этом разделе, с помощью R, будем использовать простой набор данных, который состоит из четырех строк (индивидуальные измерения, например, спектры, или какие-то другие измерения, сделанные для четырех образцов) и пяти столбцов (например, длины волн, на которых спектры были сняты, или измеренные характеристики образцов).

```
X <- matrix(c(0.55, 0.12, 0.34, 0.21, 1.15, 2.20, 1.89, 1.75, 112, 114,
   138, 120, 0.09, 0.085, 0.072, 0.081, 10.0, 12.2, 16.4, 15.1), nrow = 4)
show(X)
```

```
     [,1] [,2] [,3]  [,4] [,5]
[1,] 0.55 1.15  112 0.090 10.0
[2,] 0.12 2.20  114 0.085 12.2
[3,] 0.34 1.89  138 0.072 16.4
[4,] 0.21 1.75  120 0.081 15.1
```

Покажем эти данные с помощью графика, где ось x будет представлять номер переменной (столбца), а каждая линия — индивидуальный образец (строку). Сами значения обозначим точками на графике.

```
matplot(1:ncol(X), t(X), type = "b", lty = 1, pch = 4)
```



Хорошо заметно, что значения для каждой переменной варьируются в разных диапазонах и имеют разные центры.

Начнем с центрирования и шкалирования. Но в начале вспомним, что в R есть специальный метод, который позволяет применять различные функции либо к строкам, либо к столбцам матрицы с данными. Этот метод называется `apply()` и его синтаксис выглядит следующим образом:

```
apply(X, margin, function)
```

Здесь:

- `X` — это объект с данными, например, матрица или фрейм
- `margin` — это число, показывающее к каким элементам объекта с данными нужно применить функцию, в простейшем случае это либо 1 (к строкам), либо 2 (к столбцам)
- `function` — собственно функция, которая возьмет значения строки или столбца и вернет какой-то результат.

Если функция возвращает одно значение (например, считает сумму, или среднее значение), то результатом выполнения функции метода `apply()` станет вектор с вычисленными значениями. Например, код ниже показывает, как вычислить среднее значение и стандартное отклонение для каждого столбца `X`:

```
m <- apply(X, 2, mean)
s <- apply(X, 2, sd)
show(rbind(m, s))
```

```
      [,1]      [,2]      [,3]        [,4]      [,5]
m 0.3050000 1.7475000 121.00000 0.082000000 13.425000
s 0.1866369 0.4404827  11.83216 0.007615773  2.880249
```

Однако, функция может возвращать и вектор значений. В этом случае метод `apply()` вычислит вектор таких значений для каждого столбца, или строки исходной матрицы и затем объединит все векторы снова в матрицу.

Код ниже показывает, как можно использовать это для центрирования данных.

```
# вычислить среднее для каждого столбца X
m <- apply(X, 2, mean)
# вычесть вектор средних значений из каждой строки X
cX <- apply(X, 1, function(x) x - m)

show(cX)
```

```
       [,1]    [,2]    [,3]    [,4]
[1,]  0.2450 -0.1850  0.0350 -0.0950
[2,] -0.5975  0.4525  0.1425  0.0025
[3,] -9.0000 -7.0000 17.0000 -1.0000
[4,]  0.0080  0.0030 -0.0100 -0.0010
[5,] -3.4250 -1.2250  2.9750  1.6750
```

Здесь мы используем `apply()` два раза. В начале, применяем функцию вычисления среднего к каждому столбцу (`margin = 2`) и получаем вектор из 5 средних значений, как и в предыдущем примере.

А затем применяем нашу собственную функцию к каждой строке. Функцию мы задаем внутри без присваивания ей какого-то имени (такие функции называются анонимными). Более того, мы используем вектор `m` внутри функции, как постоянный параметр. Собственно, сама функция довольно проста — она вычитает значения вектора `m` из вектора `x`, который является единственным аргументом этой функции.

Так как в данном случае `apply()` применяется к строкам исходной матрицы данных, эти строки и отправятся в нашу функцию в качестве аргумента `x`. Т.е. для каждой строки мы вычтем вектор средних значений, что и требовалось.

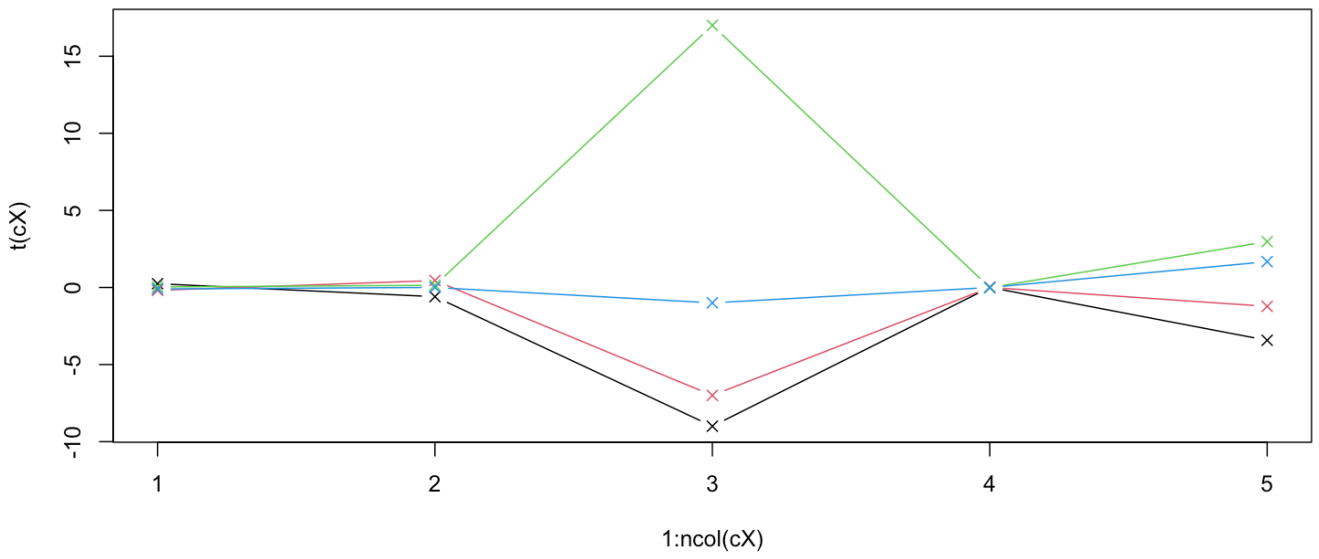
Единственный недостаток такого подхода — это то, что `apply()` соединяет результат не по строкам, а по столбцам, т.е. результат получился транспонированным и нам нужно транспонировать его обратно. Вот как это можно сделать:

```
cX <- t(apply(X, 1, function(x) x - m))
show(cX)
```

```
      [,1]    [,2] [,3]   [,4]   [,5]
[1,]  0.245 -0.5975   -9  0.008 -3.425
[2,] -0.185  0.4525   -7  0.003 -1.225
[3,]  0.035  0.1425   17 -0.010  2.975
[4,] -0.095  0.0025   -1 -0.001  1.675
```

Вот так центрированные данные выглядят на графике:

```
matplot(1:ncol(cX), t(cX), type = "b", lty = 1, pch = 4)
```



Код выше легко модифицировать, чтобы добавить шкалирование:

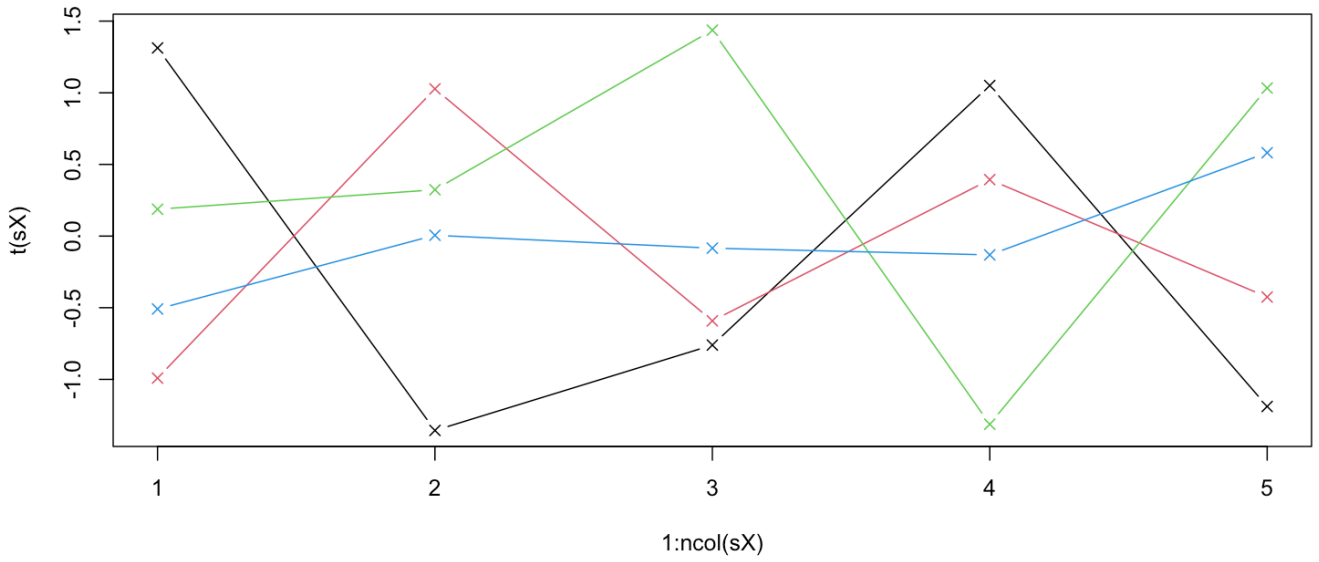
```
# вычислить среднее и стандартное отклонение для каждого столбца X
m <- apply(X, 2, mean)
```

```
s <- apply(X, 2, sd)

# вычесть вектор m  из каждой строки X и разделить ее на вектор s
# плюс транспонировать результат
sX <- t(apply(X, 1, function(x) (x - m) / s))

show(sX)
```

```
           [,1]         [,2]        [,3]       [,4]       [,5]
[1,]  1.3127093 -1.356466472 -0.76063883  1.0504515 -1.1891334
[2,] -0.9912295  1.027282140 -0.59160798  0.3939193 -0.4253105
[3,]  0.1875299  0.323508740  1.43676223 -1.3130643  1.0328969
[4,] -0.5090097  0.005675592 -0.08451543 -0.1313064  0.5815470
```

Вот так центрированные и шкалированные данные выглядят на графике:

```
matplot(1:ncol(sX), t(sX), type = "b", lty = 1, pch = 4)
```



Операции центрирования и шкалирования довольно часто используется, поэтому в R есть отдельная функция `scale()`, которая очень упрощает это действие.

```
sX <- scale(X, center = TRUE, scale = TRUE)
show(sX)
```

```
           [,1]         [,2]        [,3]       [,4]       [,5]
[1,]  1.3127093 -1.356466472 -0.76063883  1.0504515 -1.1891334
[2,] -0.9912295  1.027282140 -0.59160798  0.3939193 -0.4253105
[3,]  0.1875299  0.323508740  1.43676223 -1.3130643  1.0328969
[4,] -0.5090097  0.005675592 -0.08451543 -0.1313064  0.5815470
attr(,"scaled:center")
[1]   0.3050   1.7475 121.0000   0.0820  13.4250
attr(,"scaled:scale")
[1]  0.186636902  0.440482690 11.832159566  0.007615773  2.880248832
```

Как можно видеть, `scale()`, имеет два дополнительных логических аргумента, которые говорят функции нужно ли делать центрирование, или шкалирование или, как в примере выше, обе операции. Кроме этого `scale()` также добавляет два атрибута к полученной матрице: собственно векторы со средними значениями и стандартными отклонениями, которые использовались для преобразования.

Для того, чтобы данные нормировать, необходимо лишь поменять местами строки и столбцы при применении метода `apply()`. Вот, например, как нормировать строки `X`, чтобы сумма их значений была равна единице:

```
# считаем сумму значений каждой строки X
rs <- apply(X, 1, sum)
# делим значения в каждом столбца X на вектор сумм
nX <- apply(X, 2, function(x) x / rs)

show(nX)
```

```
             [,1]        [,2]      [,3]         [,4]       [,5]
[1,] 0.0044430083 0.009289926 0.9047581 0.0007270377 0.08078197
[2,] 0.0009330897 0.017106644 0.8864352 0.0006609385 0.09486412
[3,] 0.0021697234 0.012061110 0.8806524 0.0004594708 0.10465725
[4,] 0.0015312707 0.012760589 0.8750118 0.0005906330 0.11010566
```

Отметим, что в этом случае транспонировать ничего не нужно. Проверим, что нормализация сработала правильно, вычислив сумму значений в каждой строке нормированной матрицы:

```
rsn <- apply(nX, 1, sum)
show(rsn)
```

```
[1] 1 1 1 1
```

Как и ожидалось, все четыре значения равны единице.

В случае нелинейных преобразований все еще проще, так как в R любая функция, которая любому числу ставит в соответствие другое число, полученное в результате вычислений, натурально работает с матрицами. Вот как, например, сделать логарифмическое преобразование:

```
lX <- log(X)
show(lX)
```

```
          [,1]      [,2]     [,3]      [,4]     [,5]
[1,] -0.597837 0.1397619 4.718499 -2.407946 2.302585
[2,] -2.120264 0.7884574 4.736198 -2.465104 2.501436
[3,] -1.078810 0.6365768 4.927254 -2.631089 2.797281
[4,] -1.560648 0.5596158 4.787492 -2.513306 2.714695
```

По умолчанию вычисляется натуральный логарифм.

3.3 Сглаживание

Далее рассмотрим артефакты, связанные с инструментальными методами, используемыми для получения данных. Общеизвестно, что наличие шума может затруднить моделирование практически всех типов экспериментальных данных. При этом в большинстве хемометрических подходов непосредственно в процессе моделирования не делается никакой оценки характера, или величины шума, поэтому результаты уже априорно содержат некоторую степень неопределенности, зависящую от величины и характера шума в исходных данных. Известно большое число методов сглаживания инструментальных данных, например, сплайн-интерполяция, Фурье-фильтры, использование вейвлетов. В данной главе будут рассмотрены наиболее эффективные и широко распространенные на практике алгоритмы.

3.3.1 Метод скользящего среднего

Фильтр скользящего среднего сглаживает данные, заменяя каждое значение интенсивности в исходных данных на среднее значение заранее заданного числа соседних точек справа и слева от него:

$$x'_j = \frac{x_{j+N} + x_{j+N-1} + ... + x_{j-N}}{2N+1}$$

где x_j - значение исходного сигнала для j-й точки данных, x'_j - сглаженное значение для этой точки, N - количество соседних точек по обе стороны от x_j, а $2N + 1$ – ширина фильтра (англ. *span*).

При сглаживании методом скользящего среднего придерживаются следующих правил:

- ширина фильтра должна быть нечетной;
- точка, подлежащая сглаживанию, должна быть расположена в точке, соответствующей половине значения ширины фильтра;
- ширина фильтра уменьшается для точек, которые не имеют указанное количество соседей с обеих сторон.
- процедура сглаживания не проводится для первой и последней точки в данных

Предположим, что вы производите сглаживание данных, используя фильтр скользящего среднего с шириной фильтра равной 5 точкам. Используя правила, описанные выше, первые четыре элемента сглаженного сигнала x' определяются как:

$$x'_1 = x_1$$

$$x'_2 = \frac{x_1 + x_2 + x_3}{3}$$

$$x'_3 = \frac{x_1 + x_2 + x_3 + x_4 + x_5}{5}$$

$$x'_4 = \frac{x_2 + x_3 + x_4 + x_5 + x_6}{5}$$

Сглаженные значения для первых четырех точек произвольного набора данных показаны на рисунке 3.8. Первая точка данных не сглажена (рис. 3.8 А), поскольку у нее нет соседей слева, вторая точка сглаживается с использованием значений интенсивностей для одного соседа с каждой стороны (рис. 3.8 Б), а для третей и последующих точек используется все пять значений (рис. 3.8 В,Г).

3.3.2 Фильтр Савицкого-Голея

Сглаживающий фильтр Савицкого-Голея является эффективным способом предварительной обработки инструментальных данных для существенного уменьшения уровня шума и, следовательно, снижения погрешностей моделирования. Сглаживание Савицкого-Голея можно рассматривать, как обобщение метода скользящего среднего. Сглаженные значения вычисляют, используя метод наименьших квадратов с применением полинома заданной степени. По этой причине фильтр Савицкого-Голея также называют сглаживающим полиномиальным фильтром, или фильтром сглаживания по методу наименьших

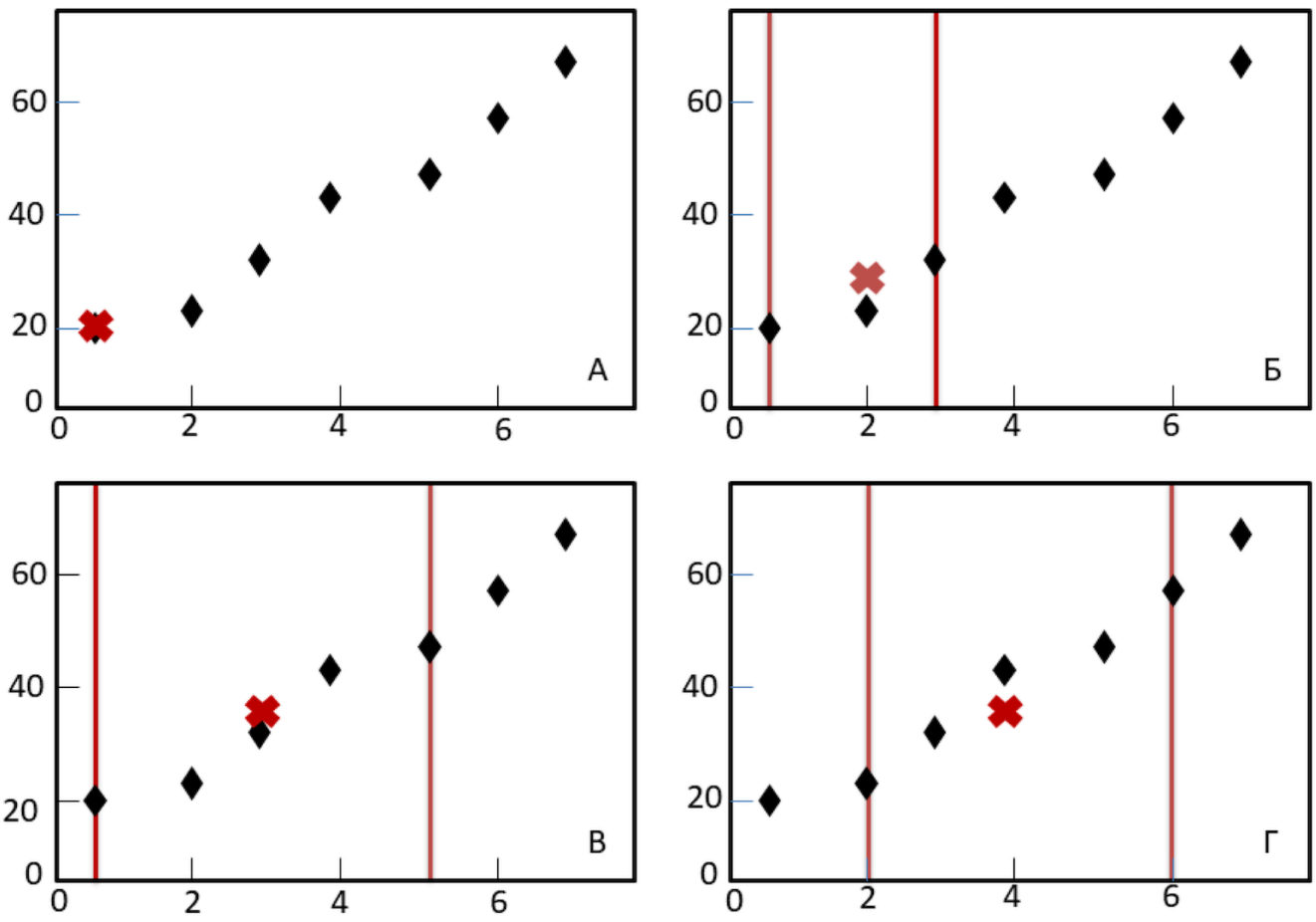


Рис. 3.8. Сглаживание методом скользящего среднего. Ширина диапазона показана вертикальными линиями.

квадратов. Обратите внимание, что полином более высокой степени позволяет достичь высокого уровня сглаживания без понижения разрешения данных.

В отличие от метода скользящего среднего, который был рассмотрен нами ранее, метод Савицкого-Голея часто используется для данных, для которых важно сохранение тонкой структуры сигнала. Однако, установлено, что сглаживание Савицкого-Голея не так эффективно для подавления шума, как метод скользящего среднего.

В процессе сглаживания методом Савицкого-Голея придерживаются следующих правил:

- ширина фильтра должна быть нечетной;
- степень полинома должна быть меньше, чем ширина фильтра;
- точки данных не обязательно должны быть равномерно распределены по оси переменных.

На рисунке 3.9 приведены зашумленный (показан светло-красным цветом) и сглаженный (показан черным цветом) с помощью метода Савицкого-Голея сигнал. Ширина фильтра в обоих случаях была равна пяти точкам. Сигнал имеет два пика: широкий, с центром около спектрального канала номер 30, и узкий, с центром около спектрального канала номер 60. Сигнал также характеризуется высоким уровнем шума.

График А на рис. 3.9 показывает результат сглаживания с использованием полинома первой степени (линии). График Б показывает результат сглаживания с помощью полинома третьей степени. Хорошо видно, что линейный полином лучше подавляет шум, но, при этом, вносит искажение в величину узкого пика. Полином третьей степени искажает пик гораздо в меньшей степени но и оставляет шум практически нетронутым.

Результаты фильтрации также сильно зависят и от ширины фильтра и от ширины спектральных пиков (сколько измеренных значений составляют данный пик). Так, если в примере выше увеличить ширину с 5 точек до 9, то результат значительно изменится. Всегда подбирайте параметры сглаживания оценивая их и визуально, и то, как они влияют на результаты моделирования.

3.3.3 Сглаживание методом ассиметричных наименьших квадратов

Рассмотренный выше популярный сглаживающий фильтр Савицкого-Голея имеет ряд недостатков. В частности, исследователи должны обращать особое внимание на недостающие значения, а также на результат вычислений на границах данных.

Привлекательной альтернативой является подход, основанный на методе ассиметричных наименьших квадратах, идея которого была предложена Витакером еще 80 лет назад. Этот фильтр чрезвычайно быстр, позволяет непрерывно контролировать процесс сглаживания, особенно на границах данных, выполняет автоматическую интерполяцию и позволяет работать с недостающими значениями и выполнять полную перекрестную проверку.

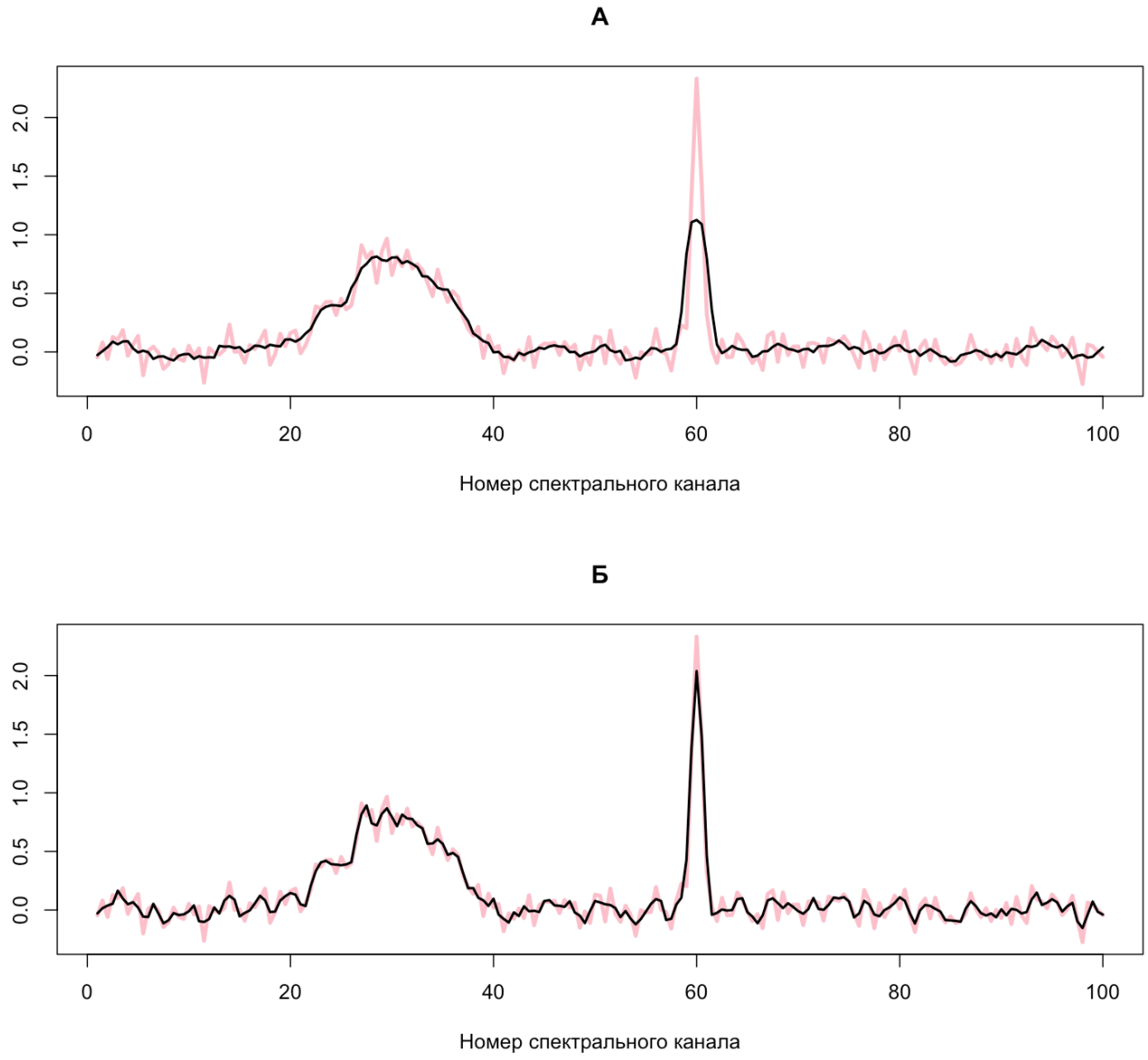


Рис. 3.9. Сглаживание методом Савицкого-Голея сгенерированного набора данных с использованием полиномов первой (А) и третьей (Б) степени. Светло-красным цветом показаны исходные спектры, черным — сглаженные.

Предположим, что нам необходимо сгладить зашумленный сигнал x, который измерен на m переменных. Для этого надо найти сглаженный сигнал x'. Однако, чем более сглажен x', тем больше он отличается от исходного сигнала x. Для контроля степени сглаживания можно использовать разницу между отдельными точками в наборе данных:

$$\Delta x'_j = x'_j - x'_{j-1}$$

Возведение в квадрат и суммирование всех разностей представляет собой простую и эффективную меру сглаживания R:

$$R = \sum (\Delta x'_j)^2$$

С другой стороны, разница между исходным и результирующим сигналом может быть выражена как сумма квадратов разностей:

$$S = \sum (x_j - x'_j)^2$$

Таким образом, исследователь может контролировать степень сглаживания с помощью параметра Q:

$$Q = S + \lambda R$$

где λ – параметр, выбираемый исследователем.

Таким образом, идея метода ассиметричных квадратов состоит в нахождении сигнала x', для которого значение Q минимально. Чем больше значение λ, тем больше влияние R на параметр Q. Это означает, что конечный сигнал будет сглажен в большей степени, однако, будет сильно отличаться от исходного профиля.

В матричной форме метод ассиметричных наименьших квадратов может быть представлен следующим образом:

$$Q = |\mathbf{x} - \mathbf{x'}|^2 + \lambda |\mathbf{D}\mathbf{x'}|^2$$

здесь $\mathbf{x}$ и $\mathbf{x'}$ - векторы, представляющие исходный и сглаженный спектр, выражение вида $|...|^2$ представляет собой сумму квадратов всех элементов вектора, а $\mathbf{D}$ – это матрица с $m-1$ строк и m столбцов, такая, что:

$$\mathbf{D}\mathbf{x'} = \Delta \mathbf{x}$$

Например, для сигнала из пяти переменных ($m = 5$) матрица $\mathbf{D}$ выглядит следующим образом:

$$\mathbf{D} = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Используя матричные вычисления, можно найти вектор частичных производных:

$$\frac{\partial Q}{\partial \mathbf{x}'} = -2(\mathbf{x} - \mathbf{x}') + 2\lambda \mathbf{D}\mathbf{D}^{\mathrm{T}}$$

Приравняв его к нулю, получим систему линейных уравнений для нахождения сглаженных сигналов:

$$(\mathbf{I} + \lambda \mathbf{D}^{\mathrm{T}}\mathbf{D})\mathbf{x}' = \mathbf{x}$$

где $\mathbf{I}$ – это единичная матрица (англ. *identity matrix*).

Для упрощения понимания описанный выше алгоритм основан на разнице первого порядка:

$$\Delta x'_j = x'_j - x'_{j-1}$$

Можно использовать и более высокий порядок, например считать разницу второго порядка:

$$\Delta(\Delta x'_j) = (x'_j - x'_{j-1}) - (x'_{j-1} - x'_{j-2}) = x'_j - 2x'_{j-1} - x'_{j-2}$$

Для иллюстрации работы фильтра на рисунке 3.10 представлен исходный и сглаженный ЯМР спектр с различными значениями d и λ. Эффективность фильтра достаточно высока, за исключением узкого сигнала в середине, который «округляется» даже при небольших значениях λ.

Подбор оптимального значения λ происходит эмпирически, пока не будет получен оптимальный результат, визуально устраивающий исследователя. Более объективный выбор может быть сделан с использованием кросс-валидации. Идея состоит в поочередном выбросе каждого элемента x и проведении сглаживания оставшихся $N - 1$ точек данных. Ошибку кросс-валидации в данном случае можно вычислить следующим образом:

$$s_{\mathrm{CV}} = \sqrt{\frac{(x_j - ?)^2}{m}}$$

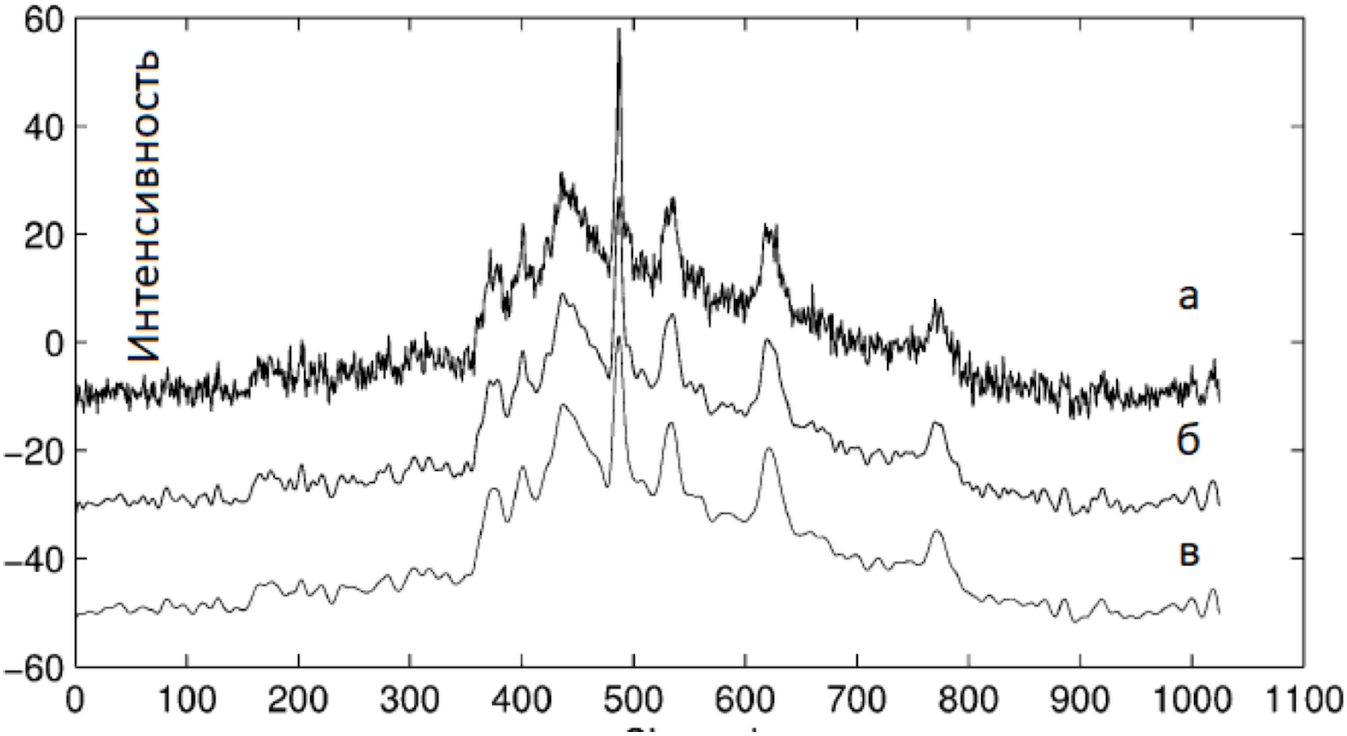


Рис. 3.10. Сглаживание зашумленного (а) ЯМР спектра древесины с использованием разниц третьего порядка (d=3) и λ=10 (б) λ=100 (в). Разрешено для воспроизведения с сайта издательства ACS Publications [7].

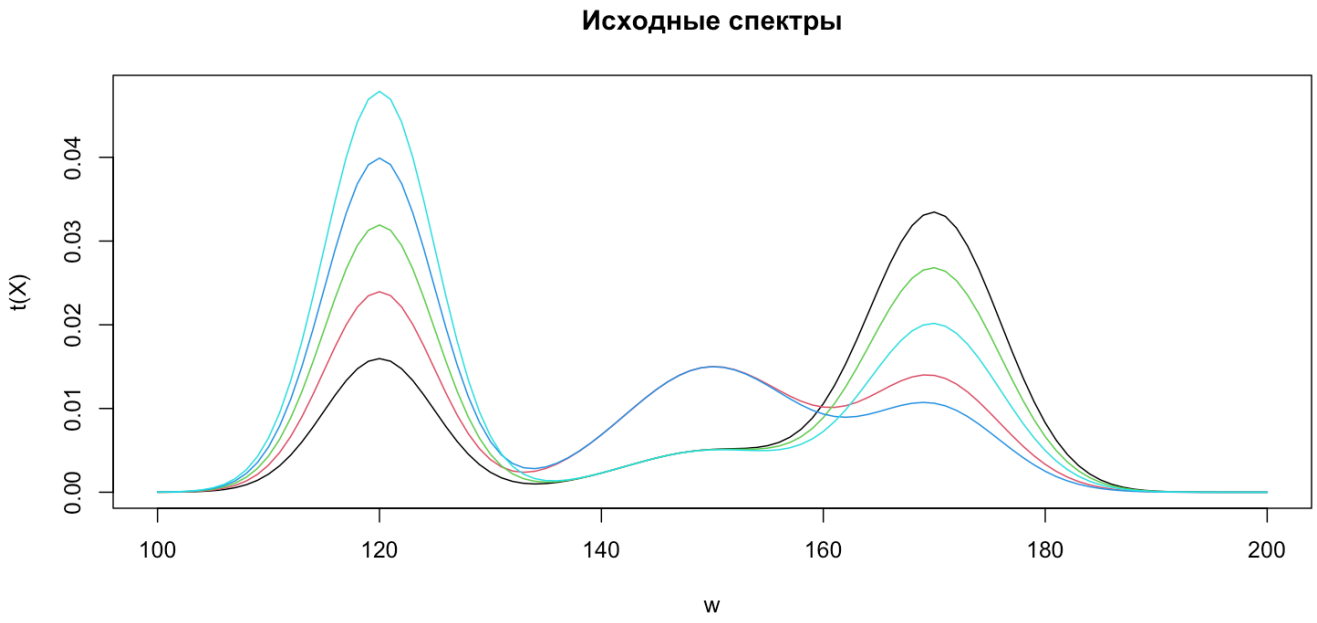
Оптимальным является значение λ для которого значение ошибки кросс-валидации минимально. В настоящее время описанные процедуры сглаживания используются в рутинном анализе и включены в стандартные пакеты обработки инструментальных данных.

3.3.4 Сглаживание сигналов в R

Для начала сгенерируем несколько “спектров” используя закон Ламберта-Бера (мы это уже делали в главе 1). Спектры будем генерировать используя функцию Гаусса, которая описывает форму случайного распределения.

```
# зададим вектор с "длинами волн" от 100 до 200 нм
w <- 100:200

# зададим спектры трех "чистых" компонент с пиками около 120, 150 и 170 нм
s1 <- dnorm(w, mean = 120, sd = 5)
s2 <- dnorm(w, mean = 150, sd = 8)
s3 <- dnorm(w, mean = 170, sd = 6)
S <- cbind(s1, s2, s3)
```

```r
# зададим 5 концентраций этих трех компонент в разных пропорциях
C <- matrix(
   c( 0.20, 0.10, 0.50,
      0.30, 0.30, 0.20,
      0.40, 0.10, 0.40,
      0.50, 0.30, 0.15,
      0.60, 0.10, 0.30
   ), byrow = TRUE, ncol = 3
)

# сгенерируем спектры смесей используя закон Бера-Ламберта
X <- tcrossprod(C, S)

# строим график для сгенерированных спектров
matplot(w, t(X), type = "l", lty = 1, main = "Исходные спектры")
```

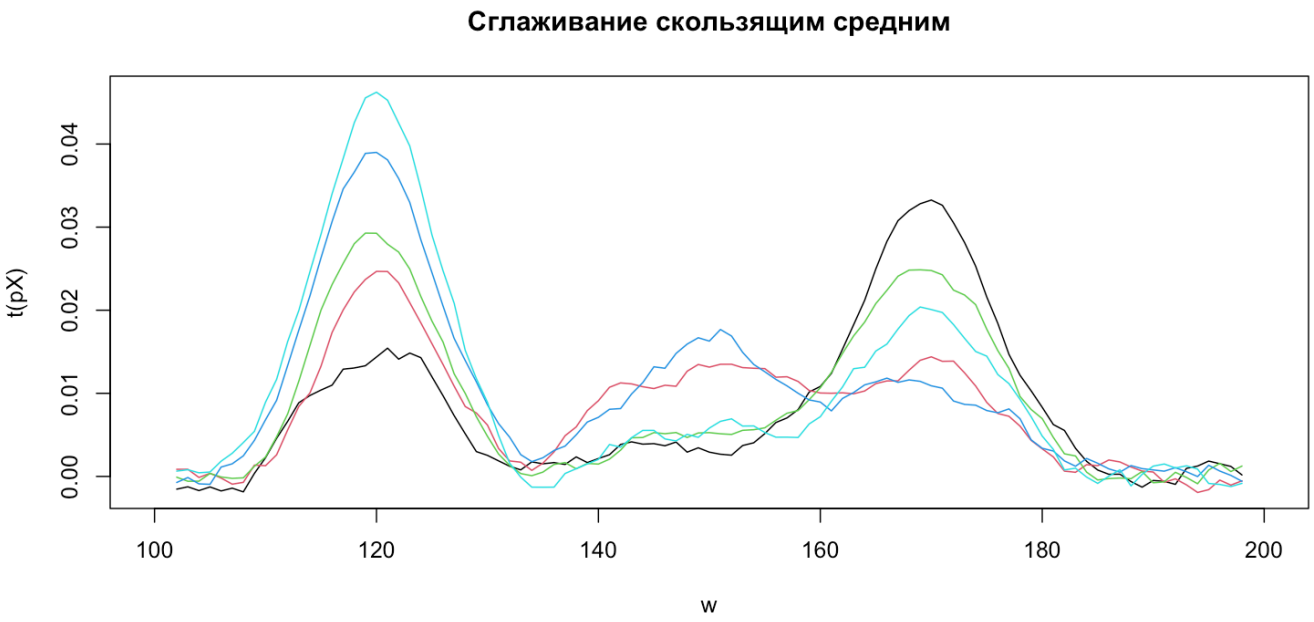


Теперь добавим к этим “спектрам” немного шума.

```r
nX <- X + matrix(rnorm(length(X), mean = 0, sd = max(X) * 0.05), nrow(X), ncol(X))
matplot(w, t(nX), type = "l", lty = 1, main = "Зашумленные спектры")
```

Самым простым способом сглаживания сигналов в R является их фильтрация с помощью метода `filter()`. Он имеет два аргумента, первый — это собственно сигнал или матрица сигналов, которые нужно обработать, а второй — это сам фильтр, точнее его веса. Скажем, для метода скользящего среднего для пяти точек каждый вес будет равняться 1/5.:

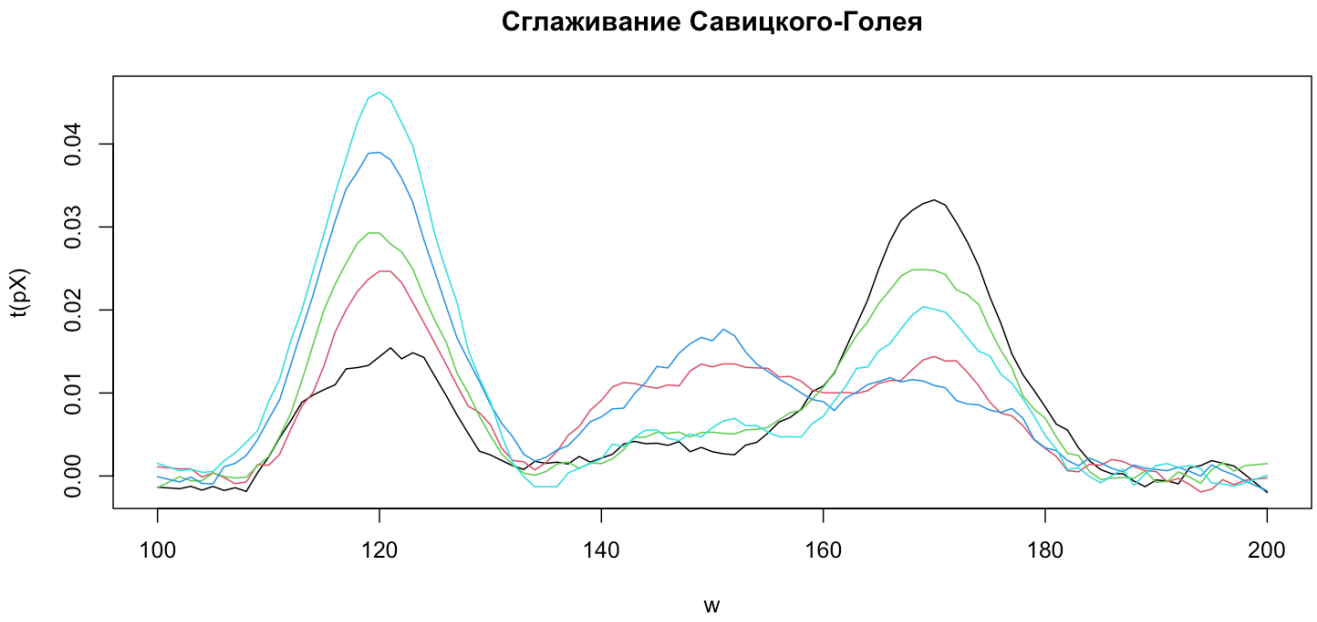
```
pX <- t(filter(t(nX), rep(1/5, 5)))
matplot(w, t(pX), type = "l", lty = 1, main = "Сглаживание скользящим средним")
```



Обратите внимание, что метод `filter()`, считает, что отдельные сигналы (спектры) находятся в столбцах матрицы с данными, а не в строках. Поэтому в блоке кода выше мы сначала транспонировали матрицу с зашумленными спектрами, а после фильтрации транспонировали ее обратно. Как можно видеть из графика, часть шума на самом деле была подавлена.

Более сложные фильтры можно найти в многочисленных пакетах для R. Например, фильтр Савицкого-Голея реализован в пакете `mdatools` который мы уже использовали в предыдущей главе. Вот пример его применения:

```
library(mdatools)
pX <- prep.savgol(nX, width = 5, porder = 1, dorder = 0)
matplot(w, t(pX), type = "l", lty = 1, main = "Сглаживание Савицкого-Голея")
```



Здесь аргумент `width` — это ширина фильтра, аргумент `porder` — степень полинома для аппроксимации (в нашем случае 1 — линия), а параметр `dorder` показывает какую производную (первую или вторую) нужно вычислить после сглаживания. Так как вычисление производной нам не требуется, его значение равно нулю в этом примере.

3.4 Коррекция базовой линии

Неидеальная базовая линия приводит к смещению сигналов относительно нулевой линии, что в процессе количественного анализа приводит к завышенным, или заниженным оценкам концентраций аналитов (рис. 3.11). В тоже время вариации базовой линии в серии образцов могут повлиять даже на результаты разведочного анализа, так как рутинные математические методы предварительной обработки данных (например, центрирование) не могут устранить этот тип дисперсии данных. Особое внимание на искажения базовой линии следует обратить в исследованиях в области метаболомики, где сигналы микрокомпонентов могут быть очень важны для построения многомерных моделей. Искажения базовой линии включают в себя вертикальное смещение и наклон сигнала, которые вызваны различными причинами в зависимости от типа инструментального сигнала. На рисунке 3.11 показано вертикальное смещение и наклон базовой линии на примере ИК спектра.

Большинство методов коррекции базовой линии основаны на моделировании базовой линии с помощью подгонки полиномиальной, синусоидальной, или экспоненциальной функции к экспериментальным данным. Наиболее часто применяемым способом является использование многочленов различного, чаще

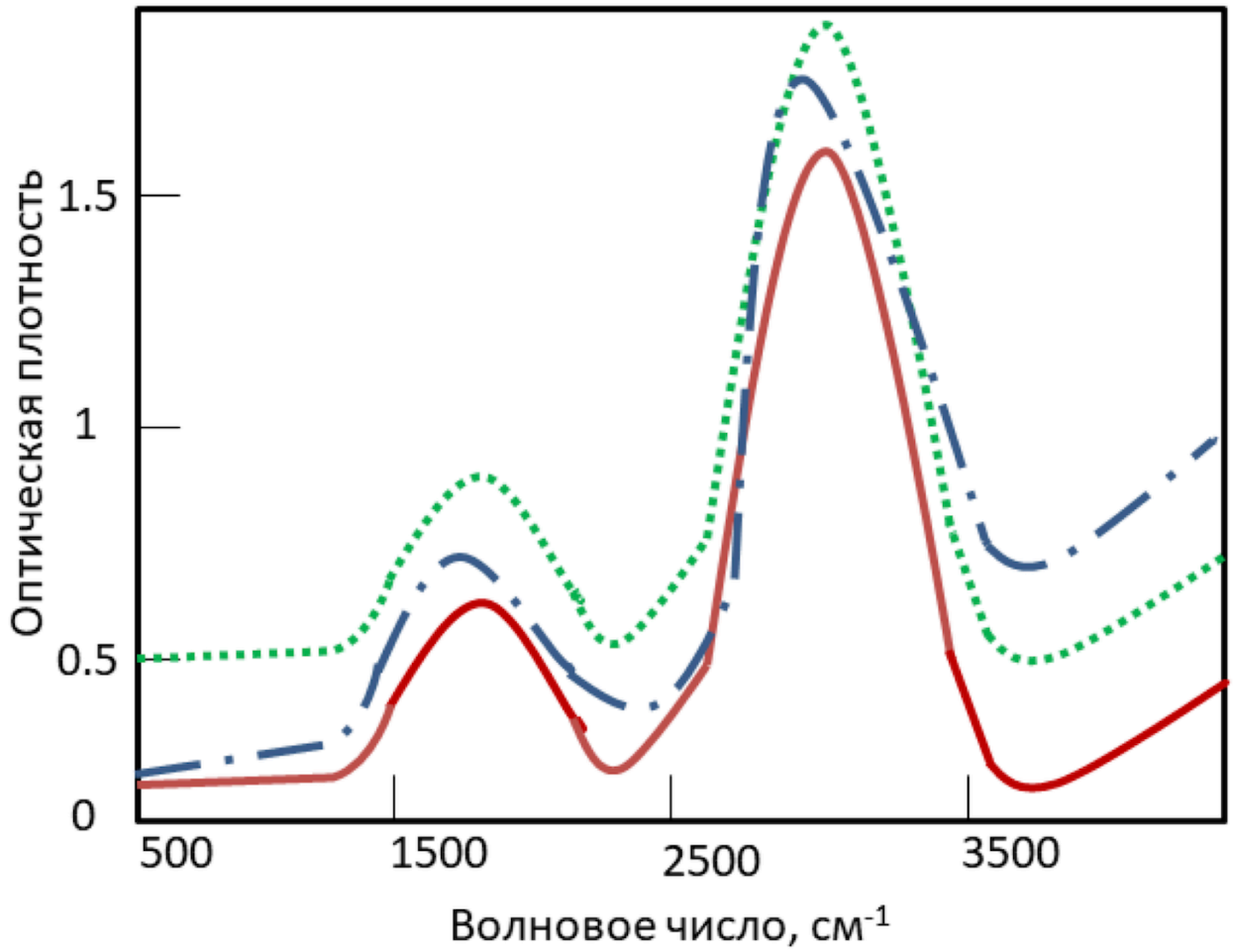


Рис. 3.11. Искажения базовой линии на примере ИК сигнала: вертикальное смещение (пунктир) и наклон (штрих-пунктир). Скорректированный спектр представлен сплошной линией.

всего, нулевого или первого порядка. Скорректированный сигнал получают вычитанием найденной функции из первоначальных данных.

Для экспериментальных данных с большим количеством полос рекомендуется проводить коррекцию базовой линии отдельно для каждого искаженного сигнала. Для этого он должен быть выделен из данных, а затем расстояние между начальной и конечной переменной моделируется искомым полиномом (рис. 3.12 — рис. 3.14). Это позволяет подбирать параметры базовой линии для каждого сигнала, так как неидеальность базовой линии проявляется по-разному в различных диапазонах и зависит от параметров сигнала (интенсивность, полуширина и др.).

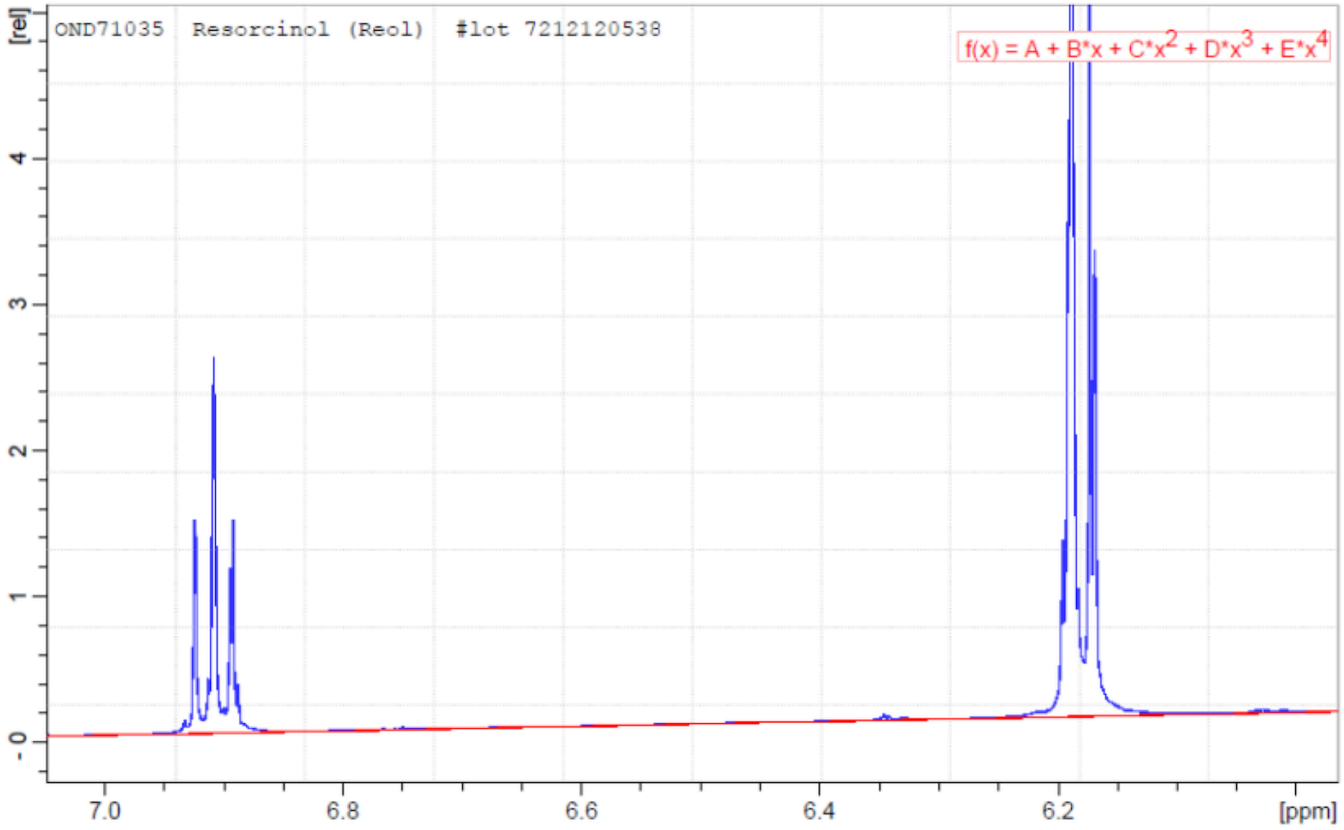


Рис. 3.12. Моделирование базовой линии ЯМР спектра красителя резорцинола полиномом второго порядка. Базовая линия обозначена красным цветом.

В альтернативном методе скользящего минимума базовую линию моделируют путем нахождения минимума в интервалах определенной ширины, а затем вычитают найденную кривую из исходных данных. В случае ЯМР спектроскопии размер интервала колебался от 5 до 30 Гц. На рисунке 3.15 представлен сигнал оксиметилфурфурола в ЯМР-спектре образца кока-колы до и после коррекции базовой линии с использованием метода скользящего минимума. Метод скользящего минимума эффективен для быстрой обработки данных, содержащих десятки или сотни тысяч переменных.

Еще одним способом коррекции базовой линии является дифференцирование сигнала. При этом с

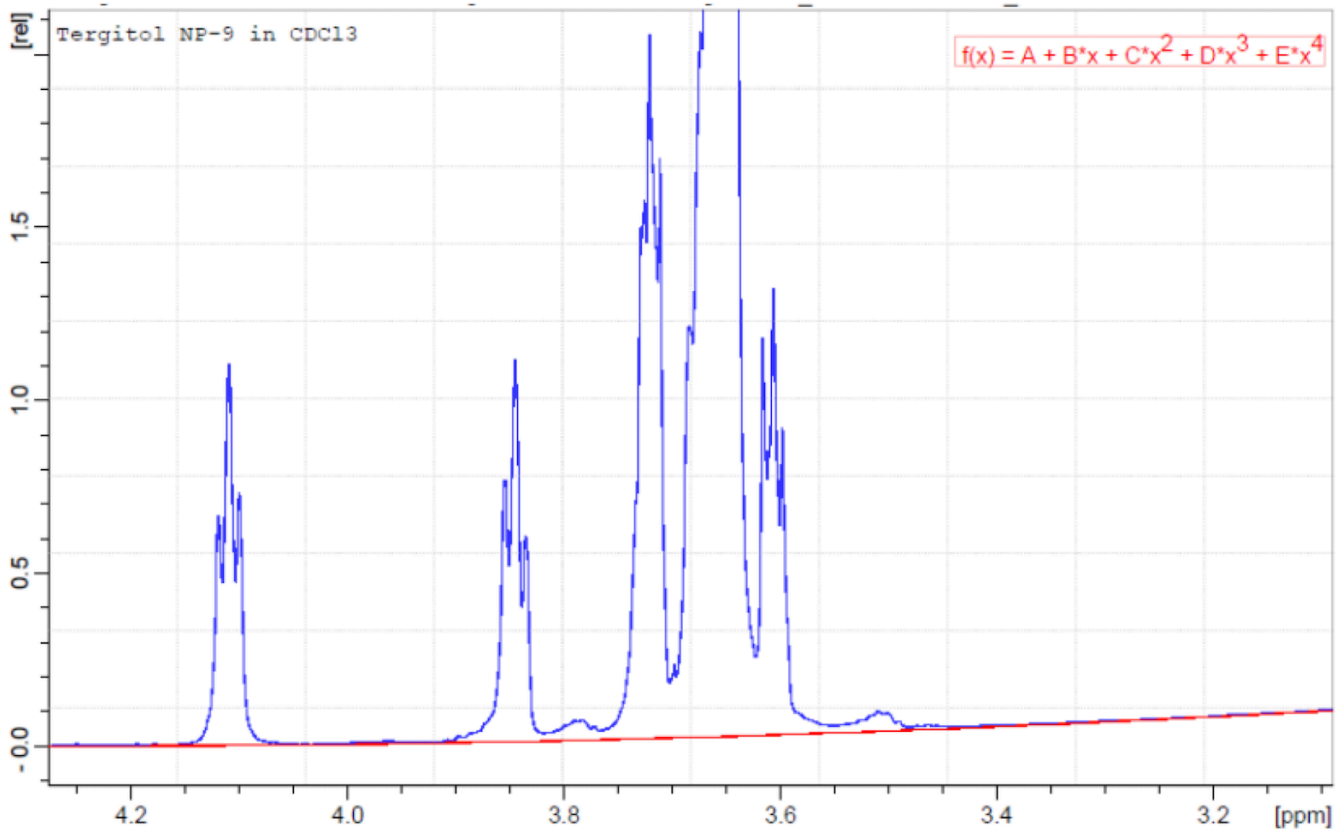


Рис. 3.13. Моделирование базовой линии ЯМР спектра тергитола (моющее средство) полиномом второго порядка. Базовая линия обозначена красным цветом.

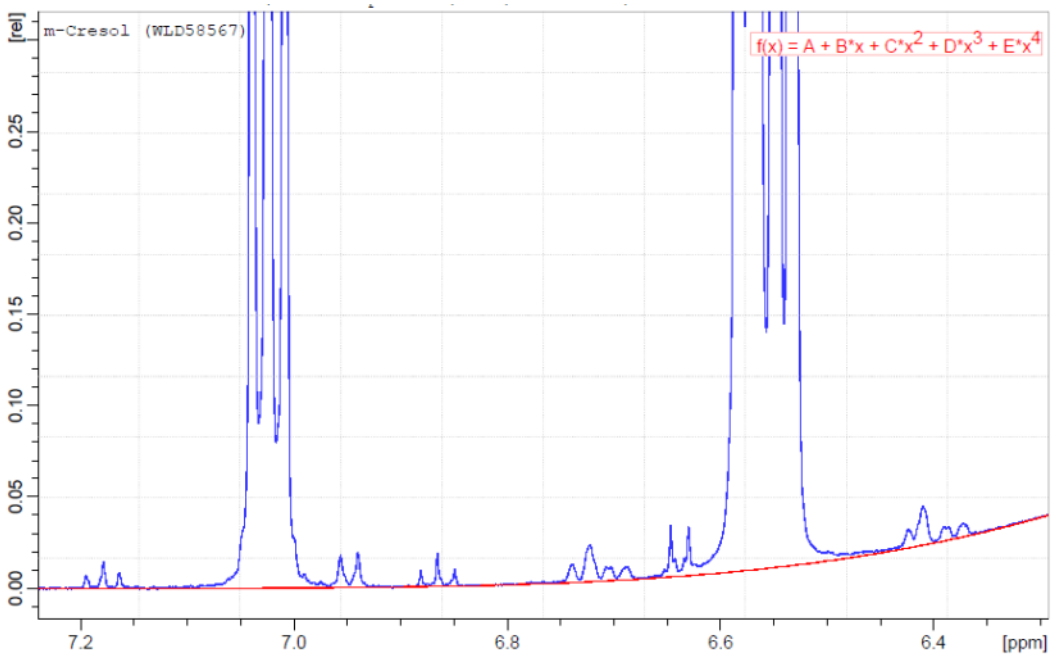


Рис. 3.14. Моделирование базовой линии ЯМР спектра красителя крезола полиномом четвертого порядка. Базовая линия обозначена красным цветом.

помощью производных первого порядка устраняется вертикальное смещение базовой линии, в то время как производные второго порядка корректируют ее наклон. Так как использование производных может уменьшить отношение сигнал/шум, алгоритм Савицкого-Голея также часто применяют перед дифференцированием для сглаживания сигнала. Кроме того, коррекцию базовой линии можно успешно проводить с помощью ассиметричных наименьших квадратов, описанных нами в разделе 4.3.

3.4.1 Коррекция базовой линии в R

Для R имеется несколько пакетов, которые реализуют различные методы коррекции базовой линии. Наиболее продвинутым, пожалуй, является пакет `baseline`, вы можете ознакомится со списком реализованных методов на GitHub репозитории этого пакета: https://github.com/khliland/baseline/.

3.5 Коррекция фазы

Одной из проблем, часто встречающихся в экспериментальных данных, являются дефекты фазы. Например, в случае ЯМР спектроскопии выделяют две главных причины подобных искажений: задержка в импульсной программе (англ. *pulse delay*) и внерезонансные эффекты (англ. *off-resonance effects*). В импульсных программах небольшие задержки в несколько микросекунд вводят между закрытием передатчика и открытием приемника для записи сигнала спада свободной индукции в целях защиты электроники приемника от негативного воздействия магнитного импульса. Из-за этих задержек регистрируемый сигнал представляет собой смесь поглощения и рассеяния, вследствие чего

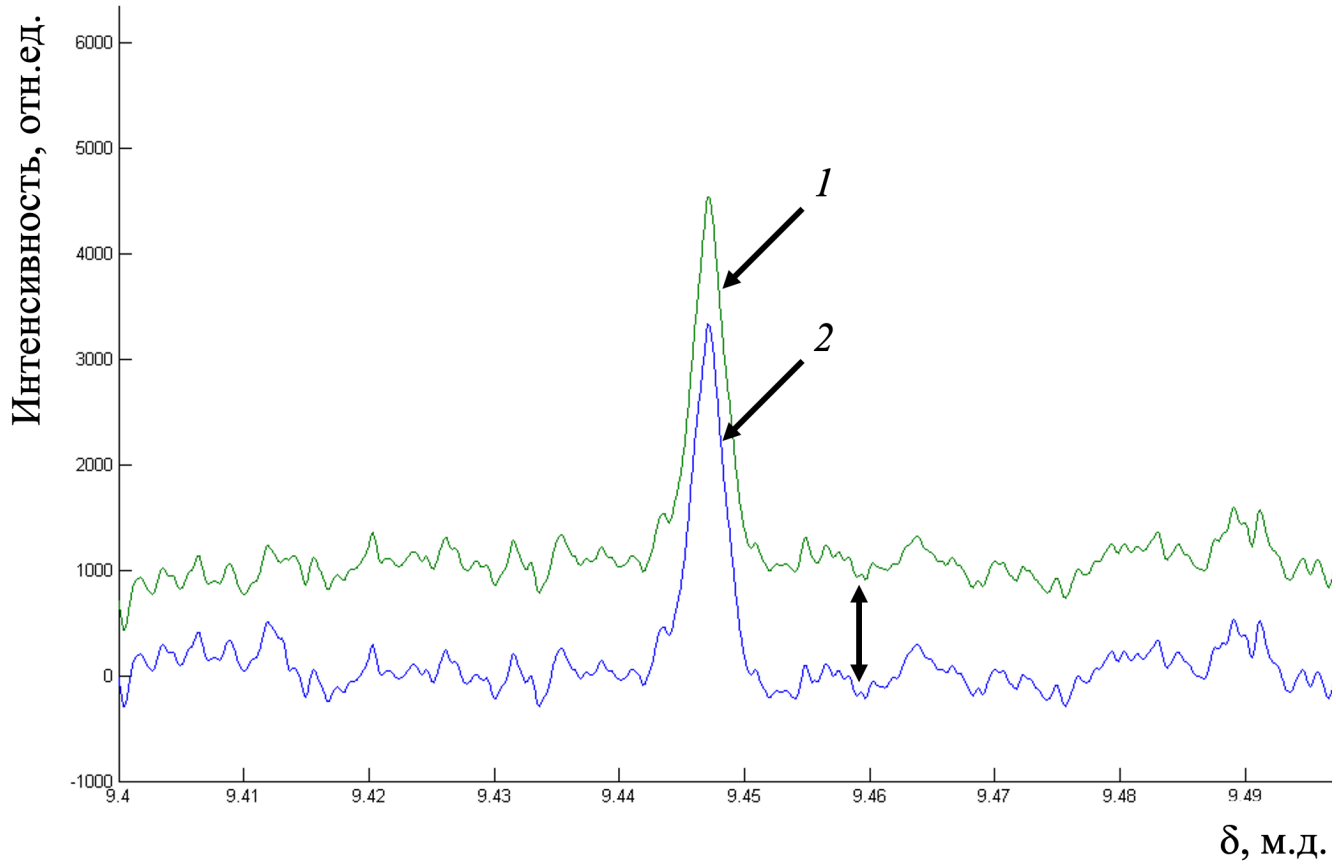


Рис. 3.15. ЯМР сигнал оксиметилфурфурола в спектре образца кока-колы до (1) и после коррекции (2) базовой линии методом скользящего минимума. Стрелкой обозначено вертикальное отклонение базовой линии от нулевой линии.

часть наблюдаемых сигналов смещается вниз относительно базовой линии. Внерезонансные эффекты являются второй причиной нарушения фазы спектров. Если мощность импульса недостаточная для одинакового возбуждения всех ядер в молекуле, то для ядер, которые резонируют вне возбуждающей пропускной способности импульса, вектор намагниченности не переворачивается точно на плоскость xy после 90° импульса. Это приводит к разности в фазе между сигналами для внерезонансных и нормально резонирующих ядер.

Выравнивание фазы представляет собой рутинную процедуру и включает в себя коррекцию нулевого и первого порядка. Коррекция нулевого порядка (задержка в импульсной программе) не зависит от частоты сигнала и затрагивает все ядра в одинаковой степени. С другой стороны, коррекция первого порядка (внерезонансные эффекты) зависит от положения сигнала. Она состоит в подборе параметров для их подстраивания к референтному сигналу. Можно выбрать любой референтный сигнал, но часто это самый интенсивный сигнал в спектре. Для него сначала проводится коррекция фазы нулевого порядка. Затем проводят коррекцию фазы первого порядка для остальных сигналов. В силу линейности процедура коррекции первого порядка одинаково подходит для сигналов микро- и макро- соединений.

3.6 Эффект рассеяния света

Эффекты рассеяния (рис. 3.16) являются общими для всех аналитических методов, использующих свет, например, ИК или УФ-спектроскопии. Рассеяние света происходит потому, что размер частиц в образце примерно равен длине волны источника света.

В большинстве случаев эффекты рассеяния корректируют, сравнивая интенсивность сигнала с референтным сигналом. Предполагается, что большинство систематических различий между референтным и корректируемым сигналом обусловлены рассеянием. Все спектры в серии данных корректируют делением интенсивностей на найденную константу рассеяния.

Для коррекции рассеяния света используют различные методы. Например, метод стандартного нормального изменения (standard normal variate, SNV) заключается в вычитании средней интенсивности спектра из каждой переменой и последующего деления на стандартное отклонение инструментального сигнала:

$$x'_{ij} = \frac{x_{ij} - \bar{x}_i}{s_i}$$

где x'_{ij} и x_{ij} - интенсивность i-го сигнала (спектра) в точке j после и до корректуры, соответственно; $\bar{x}_i$ и s_i - среднее значение и стандартное отклонение вычисленные для всех значений i-го сигнала, соответственно.

На рисунке 3.17 показан пример использования метода SNV для серии БИК спектров препаратов парацетамола.

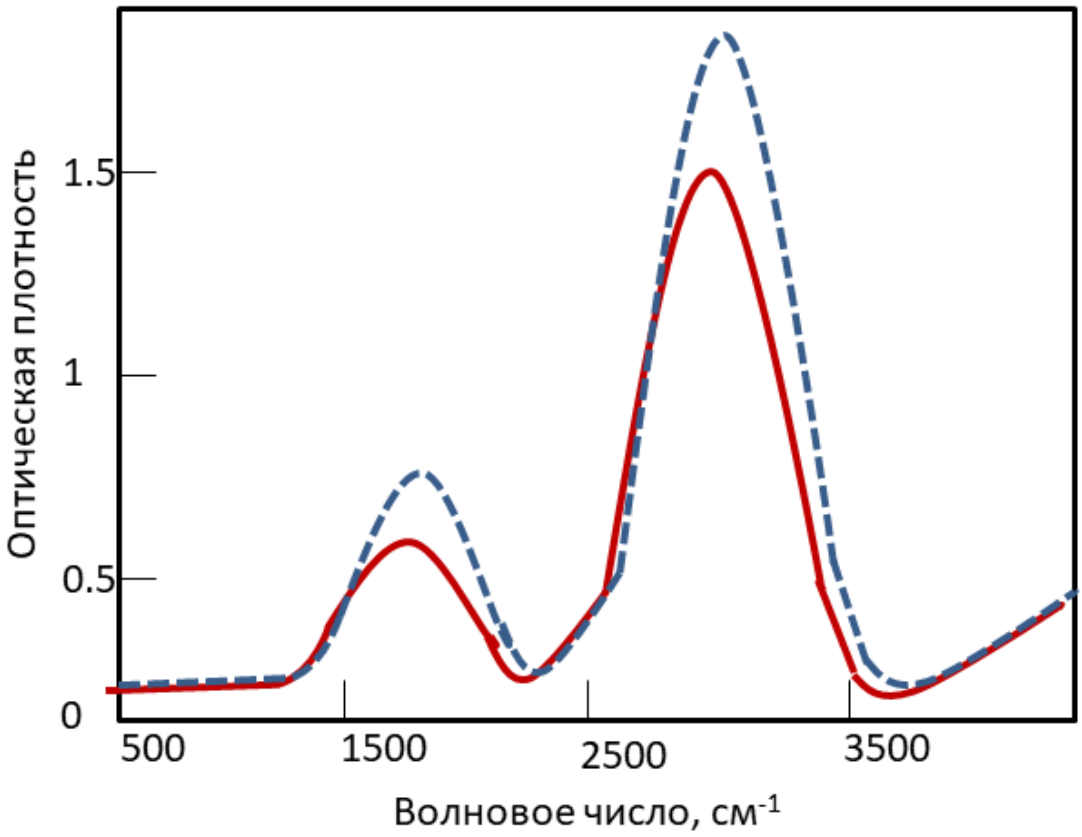


Рис. 3.16. Влияние рассеяния на смоделированный ИК спектр: исходный (сплошная линия) и искаженный (пунктир) сигналы.

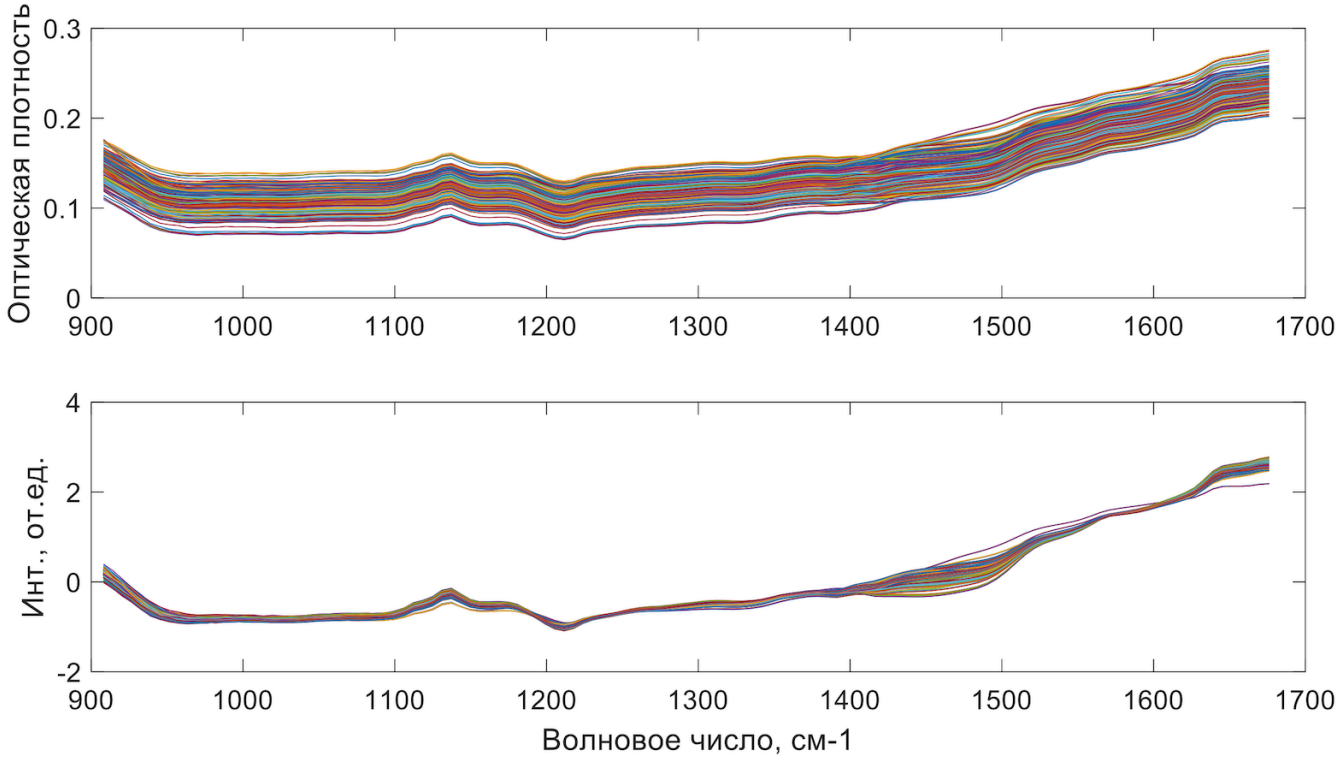


Рис. 3.17. Использование процедуры SNV для серии БИК спектров парацетамола. Исходные и скорректированные спектры представлены на верхнем и нижнем рисунке, соответственно.

К альтернативному подходу относится мультипликативная коррекция сигнала (multiplicative scatter correction, MSC). Сущность метода состоит в оценке коэффициента рассеяния путем подгонки сигнала x к референтному сигналу x_{ref} путем расчета коэффициентов b_0 и b_1:

$$x = b_0 + b_1 x_{\text{ref}}$$

$$x' = (x - b_0)/b_1$$

В качестве примера на рис. 3.18 приведены исходные и скорректированные методом MSC ИК спектры серии препаратов парацетамола.

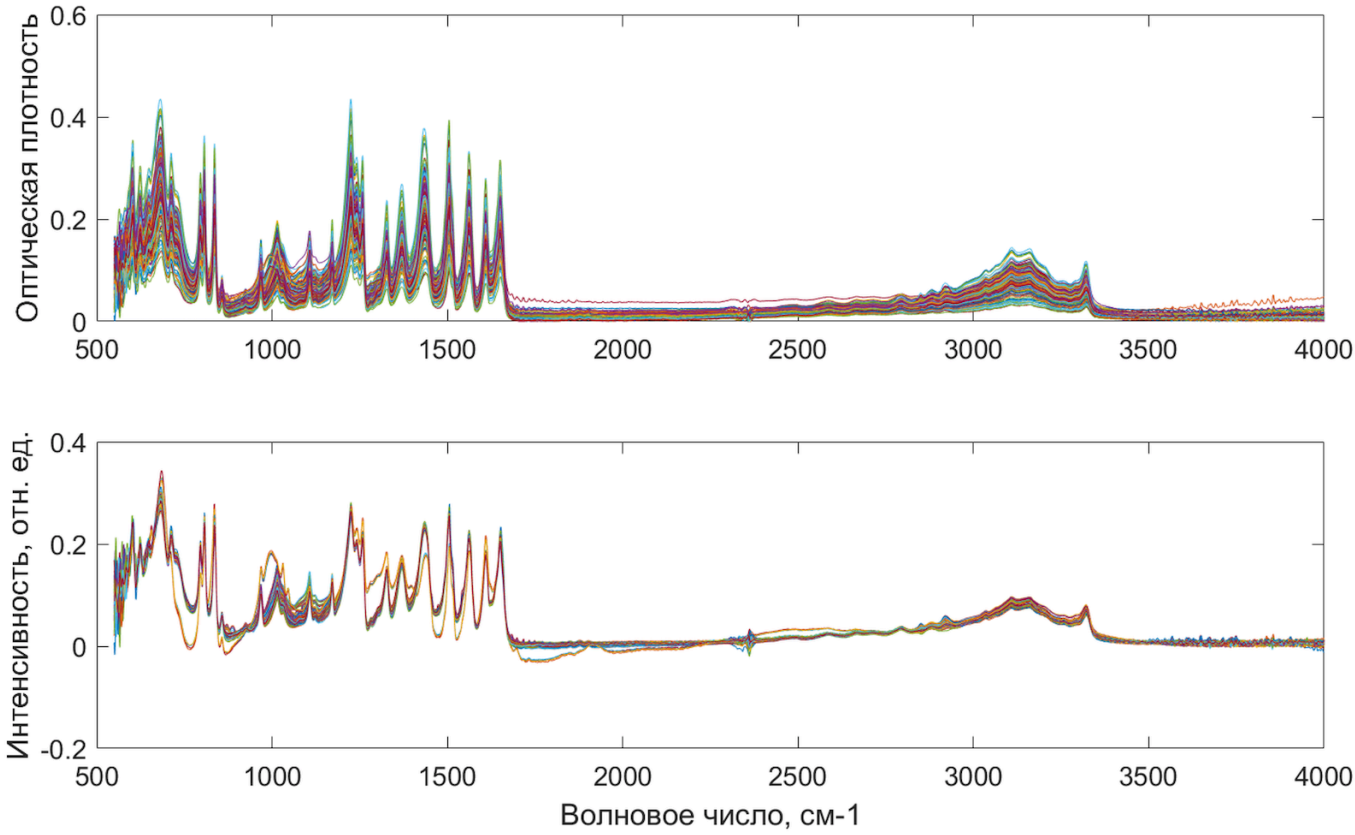


Рис. 3.18. Использование процедуры MSC для серии ИК спектров препаратов парацетамола. Исходные и скорректированные спектры представлены на верхнем и нижнем рисунке, соответственно.

3.6.1 Реализация в R

Оба описанных метода, SNV и MSC, легко реализуются самостоятельно в R. Например, для SNV можно использовать код для шкалирования, которые мы написали выше, но применять его к строкам матрицы с

данными, а не к столбцам.

Эти методы также реализованы в различных пакетах, включая, уже знакомый нам `mdatools`. Соответствующие функции носят названия `prep.snv()` и `prep.msc()`, они не требуют дополнительных аргументов.

3.7 Горизонтальное смещение сигналов и методы его выравнивания

Одной из основных предпосылок успешного хемометрического моделирования является обеспечение воспроизводимости положения максимумов сигналов в серии измерений. Горизонтальное смещение чаще всего проявляется при работе с данными хроматографического или ЯМР эксперимента. Следует отметить, что необходимость коррекции горизонтального смещения сигналов отсутствует для ряда хемометрических методов, таких как PARAFAC2, который косвенным образом компенсирует искажения во время моделирования.

В самом простом случае все сигналы могут быть сдвинуты горизонтально на определенную константу, как показано на рисунке 3.19 для ЯМР спектров образцов гепарина. Подобное явление связывают с флуктуациями условий окружающей среды, в частности, температуры. С искажениями подобного типа легко справиться, фиксируя положение одного из сигналов на оси переменных. В случае ЯМР спектрометрии источником такого сигнала является тетраметилсилан для органических растворителей и триметилсилил пропановой кислоты (или 4,4-диметил-4-силапентан-1-сульфоновая кислота (DSS)) для водных сред. Тщательное воспроизведение пробоподготовки и настроек прибора значительно уменьшает подобное горизонтальное смещение полос в серии однотипных экспериментов.

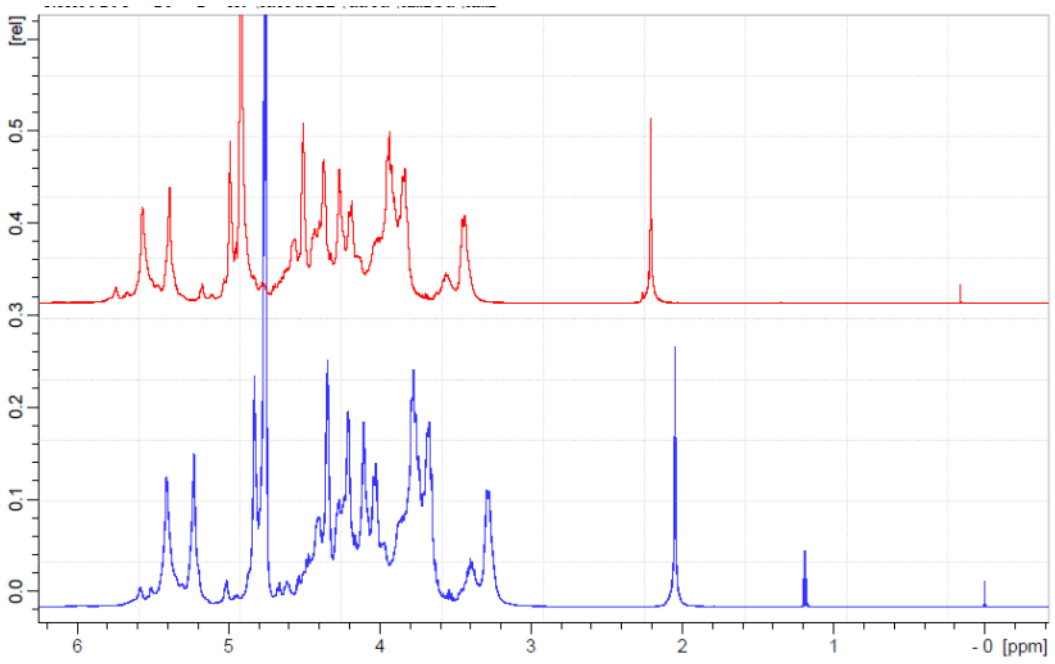


Рис. 3.19. Горизонтальное смещение двух ЯМР спектров гепарина относительно друг друга.

Однако, несмотря на стандартизацию условий, даже в хорошо спланированном исследовании экспериментальные данные иногда требуют применения более сложных алгоритмов совмещения сигналов. Дело в том, что сигналы разных соединений могут иметь случайное (как положительное, так и отрицательное) горизонтальное смещение по шкале переменных. Подобная картина особенно характерна для сигналов, положение которых зависит от pH среды. Далее рассмотрим два наиболее часто встречающихся метода коррекции подобного случайного смещения сигналов: бакетинг и алгоритм интервального коррекционного смещения (англ. *interval-correlation-shifting, icoshift*).

3.7.1 Бакетинг

Бакетинг, или биннинг – один из наиболее широко применяемых подходов при предварительной обработке ЯМР спектров. Процедура бакетинга состоит в сегментировании спектров на небольшие участки, называемые бакетами, и расчете площади под каждым таким сегментом спектра. Интегральная интенсивность может быть заменена одним значением максимальной интенсивности в выбранном диапазоне. Полученную матрицу, состоящую из значений интегральной интенсивности (максимумов полос), затем моделируют с помощью хемометрических методов (рис. 3.20).

Известно несколько вариантов бакетинга. Стандартным является эквидистантный бакетинг (англ. *rectangular bucketing*), который удобен для моделирования в случае отсутствия априорных знаний о данных. Ширина бакетов остается постоянной для всех сигналов в спектре, и ее подбирают эмпирически, для ЯМР спектров обычно в пределах 0.005 – 0.05 м.д.

В варианте бакетинга по переменным (англ. *variable size bucketing*) ширину бакета устанавливают для каждого сегмента в зависимости от ширины сигналов в нем. В данном случае небольшие горизонтальные сдвиги сигналов не мешают моделированию.

Самым «тонким» вариантом является динамический бакетинг (англ. *dynamic bucketing*). В данном случае спектры моделируются один за другим. При этом новый столбец в обработанной матрице появляется при обнаружении сигнала в спектре. Спектры образцов, не имеющие данный сигнал, аппроксимируются нулем в результирующей таблице.

Бакетинг рутинно используют в качестве метода предварительной обработки данных перед применением метода главных компонент и классификационных алгоритмов для ЯМР-спектров пищевых продуктов и напитков (например, географическое и ботаническое происхождение, проверка подлинности маркировки). На рисунке 3.21 представлены исходные и обработанные спектры серии препаратов гепарина.

Кроме коррекции смещения сигналов, дополнительным преимуществом бакетинга является значительное уменьшение размера данных, что приводит к ускорению расчетов. С другой стороны, основным недостатком бакетинга является потеря большого количества информации, заключенной в исходных ЯМР-спектрах (тонкая структура мультиплетов) (рис. Figure 3.20 и Figure 3.21). Кроме того, попадание сигналов

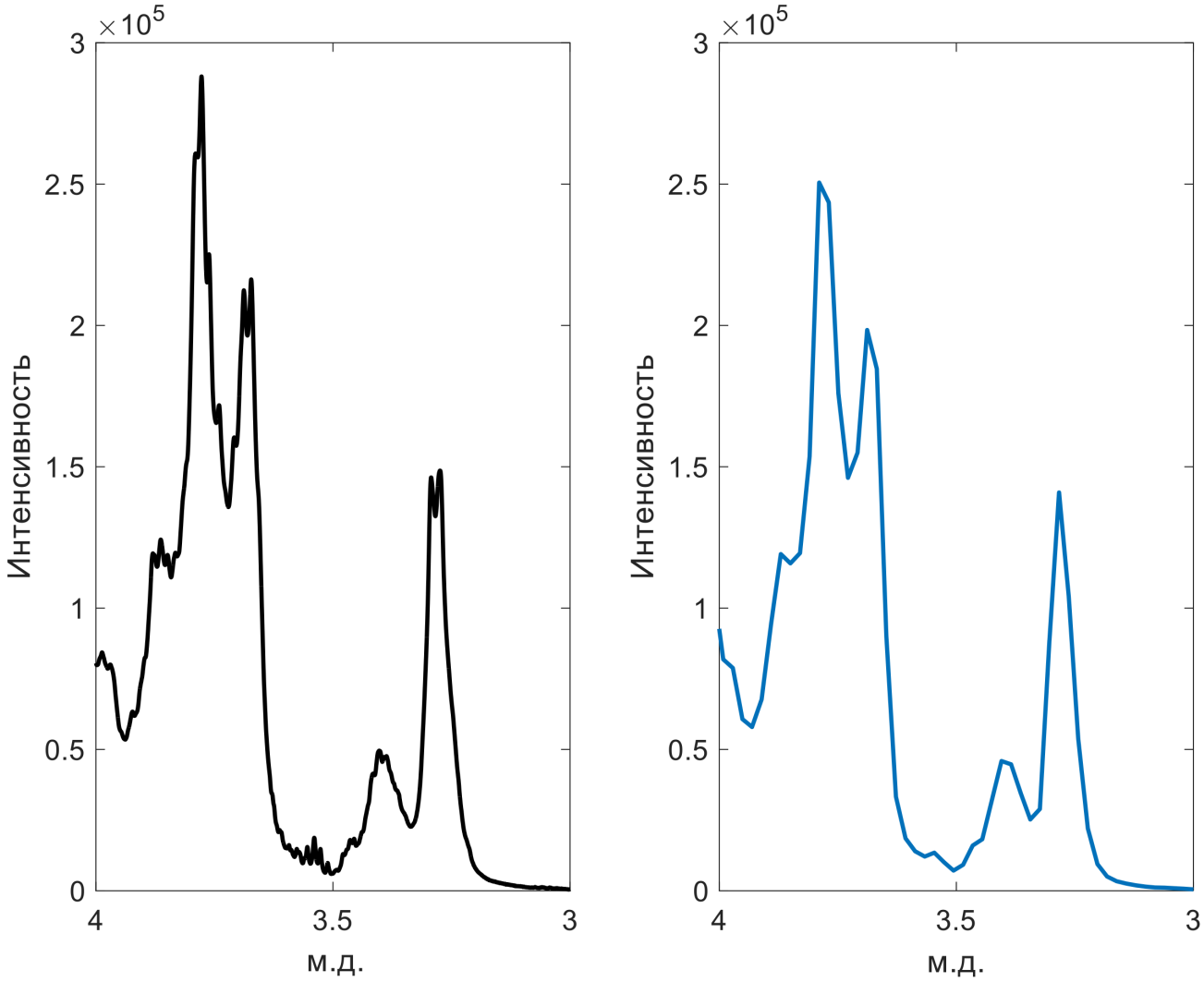


Рис. 3.20. Применение процедуры бакетинга для ЯМР спектра гепарина. В процессе обработки теряется тонкая структура сигналов (δ 3.3, 3.7, 3.8 м.д.).

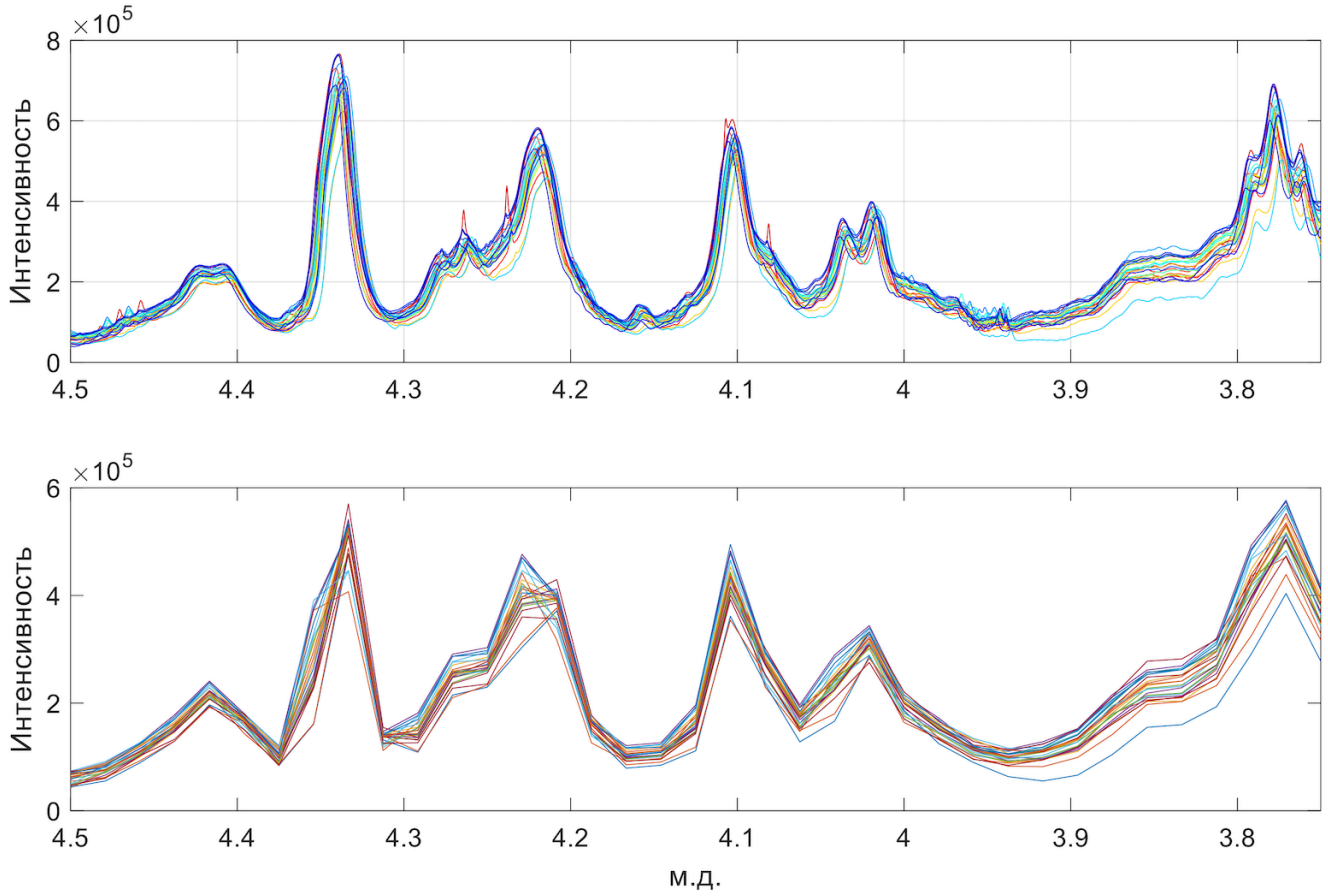


Рис. 3.21. Применение процедуры бакетинга с шириной бакета 0.01 м.д. для серии ЯМР спектров образцов гепарина. Исходные и скорректированные спектры представлены на верхнем и нижнем рисунке, соответственно.

одного мультиплета в соседние бакеты может внести дополнительные искажения в экспериментальные данные и, следовательно, затруднить моделирование данных.

3.7.2 Интервальные методы

В последнее время тенденцией в развитии алгоритмов совмещения сигналов является разработка так называемых интервальных методов, в которых глобальную проблему выравнивания данных измерений сокращают до небольших локальных задач в определенных диапазонах. При этом большинство подобных алгоритмов растягивают, или сжимают каждый сигнал таким образом, чтобы он в наибольшей степени соответствовал референтному профилю. Локальное совмещение применяется в тех случаях, где любой сигнал может быть сдвинут в двух направлениях.

Одним из наиболее часто используемых алгоритмов является интервальное коррекционное смещение (англ. *icoshift*), основанное на использовании Фурье-преобразования для одновременной обработки всех измерений в наборе данных. Алгоритм был первоначально предложен для ЯМР спектров, однако, впоследствии также нашел широкое применение в хроматографии. Таким образом, один и тот же метод коррекции может быть с равным успехом использован для артефактов с различным физическим происхождением. Метод *icoshift* совмещает сигнал каждого образца с «целевым» спектром, который может представлять собой одно из измерений, или усредненный сигнал, с помощью максимизации кросс-корреляции между выбранными интервалами. Следует отметить, что время последующей обработки данных в этом случае лишь незначительно увеличивается.

На рис. 3.22 представлены результаты работы алгоритма *icoshift* для обработки ЯМР 13C спектров жировой фракции молока в области сигнала карбоксильных групп жирных кислот. Видно, что в данном случае алгоритм приводит к полному совмещению сигналов без потери разрешения спектров. Спектры, обработанные подобным образом, использованы для разработки хемометрической модели определения типа производства молока (био/традиционное) на основе ЯМР 13C спектрометрии и данных по стабильным изотопам (δ13C для жировой фракции и казеина) с процентом правильных классификаций до 95%.

Интервальные методы выравнивания имеют два основных недостатка. Первый чисто технический: алгоритмы относительно медленные, что усложняет оптимизацию параметров. Более того, временной фактор ограничивает применимость метода к очень большим наборам данных. Во-вторых, другие присутствующие в данных искажения, например, дефекты базовой линии, эффекты рассеяния или высокий уровень шума значительно снижают производительность большинства интервальных методов. В связи с этим, еще раз подчеркнем, что выбор оптимальной стратегии предварительной обработки данных не всегда является простой задачей.

3.7.3 Реализация в R

Для коррекции горизонтального смещения и других подобных операций с ЯМР спектрами в R имеется замечательный пакет `speaq` https://cran.rstudio.com/web/packages/speaq/index.html.

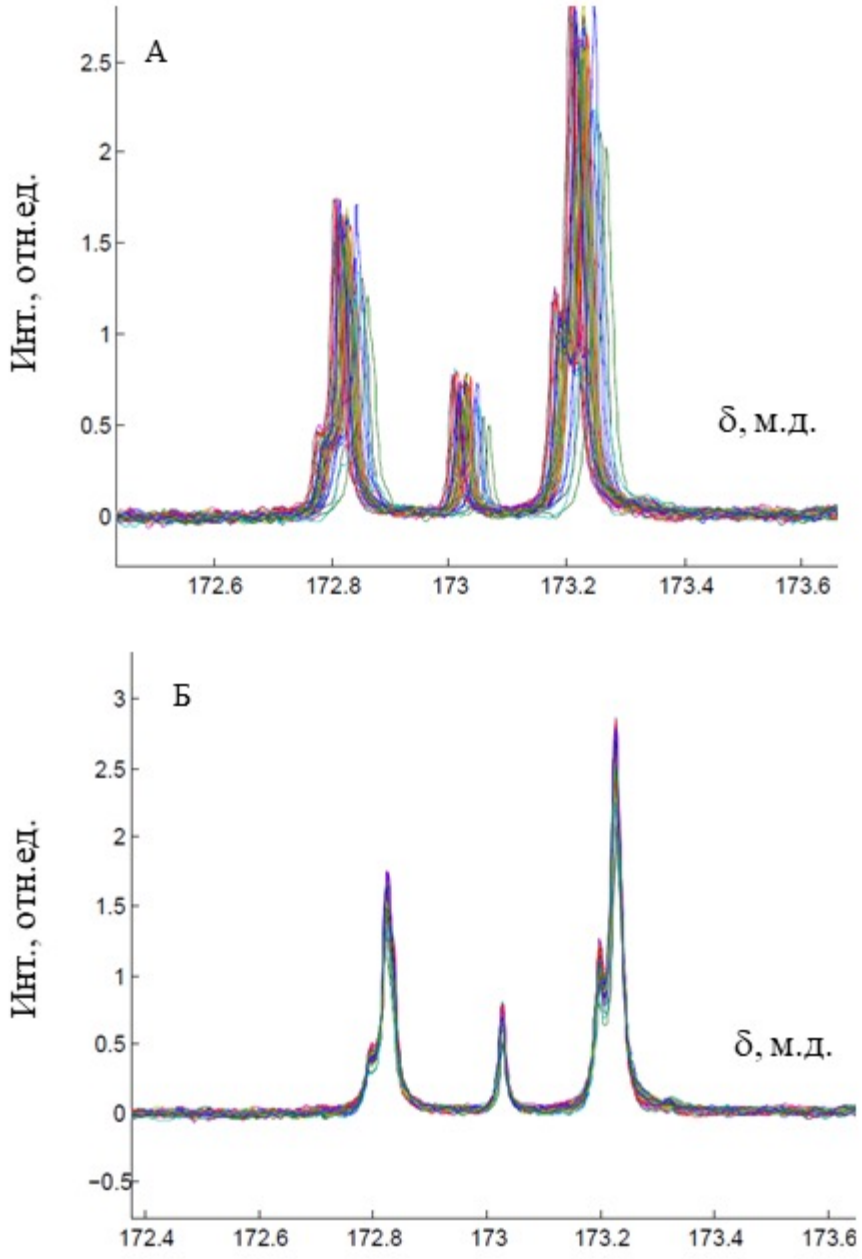


Рис. 3.22. Спектры ЯМР 13С серии образцов молока в области сигналов карбоксильных групп жирных кислот: исходные (а) и обработанные методом icoshift (б) данные.

3.8 Другие методы предварительной обработки данных

Для предварительной обработки данных часто используют дифференцирование различных порядков Чаще всего производные используют для разрешения перекрывающихся между собой полос. Например, применение производных является эффективным практическим приемом, позволяющим заметно повысить точность декомпозиции инструментальных данных методом независимых компонент, о котором речь пойдет в главе 7. Для этого на вход алгоритмов подаются данные по сигналам смесей после численного дифференцирования по длине волны с использованием метода конечных приращений (рис. 3.23). Фактически, это эквивалентно повышению контраста данных, усилению деталей, отличающих один компонент от другого. В силу линейности метода моделирование в пространстве производных приводит к тем же результатам, что и декомпозиция исходной матрицы.

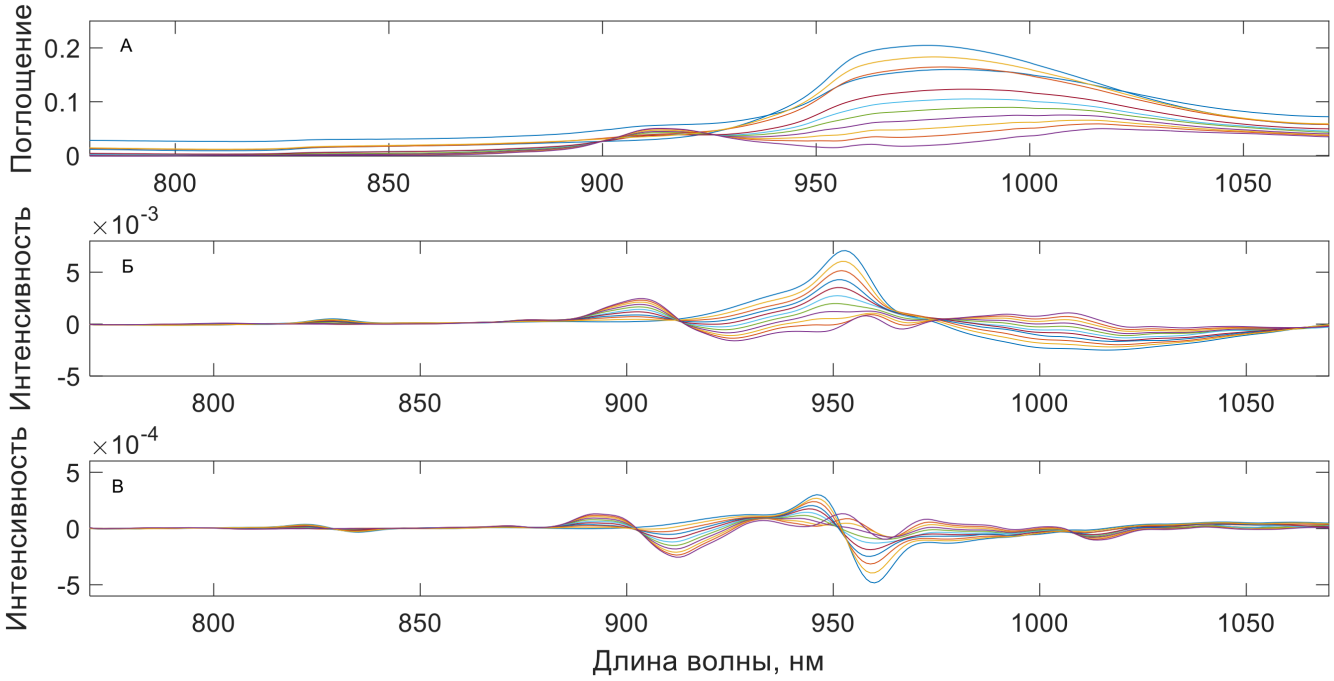


Рис. 3.23. БИК спектры парацетамола: исходные данные (А) и результат их дифференцирования с использованием производных первого (Б) и второго (В) порядков.

Выброс сигналов растворителя и «нулевых» областей. Кроме расчета производных исследователи часто «выбрасывают» незначащие для конкретной аналитической проблемы данные, например, сигналы растворителей, «нулевые» диапазоны и помехи (например, сигнал CO2 в ИК спектрах). Это позволяет, с одной стороны, уменьшить размерность данных, а с другой сконцентрироваться на важных переменных для моделирования.

Таким образом, предварительная обработка, направленная на удаление нежелательных искажений из экспериментальных данных, является ключевым моментом для проведения удачного многомерного анализа, будь то классификация, разведочный анализ, декомпозиция экспериментальных профилей,

или многомерная градуировка. Все рассмотренные методы могут быть использованы при коррекции экспериментальных данных для создания эффективных хемометрических моделей анализа продуктов питания, объектов окружающей среды и других образцов сложного состава.

Как отмечалось ранее, результаты хемометрического анализа сильно зависят от выбора методов предварительной подготовки данных и их последовательности. Опыт показывает, что очень часто на начальном этапе анализа необходимо тестировать разные способы преобразования данных. Важно помнить, что такие преобразования почти всегда оказывают влияние на графики МГК счетов в случае разведочного анализа, и это может привести к их различной интерпретации.

4 Методы регрессионного анализа

4.1 Основные понятия

Задачей *регрессионного анализа* является количественное определение характеристик образца (концентрации отдельных компонентов, физические, либо химические свойства, и т.д.) на основе результатов измерений каких-либо других свойств, связанных с этими характеристиками. Такой подход имеет смысл, если прямое измерение интересующих характеристик невозможно, или требует серьезных временных и трудозатрат. Например, оценить температуру окружающей среды можно по высоте столбца рабочей жидкости в капилляре, а определить содержание металла в растворе — по интенсивности поглощения раствором света определенной длины волны. Очевидно, что измерение поглощения света гораздо быстрее и требует меньше ресурсов по сравнению с измерением концентрации металлов с помощью методов «мокрой» химии.

В обоих этих случаях связь между непосредственно измеряемой величиной (высота капилляра, доля поглощенного излучения) и определяемой характеристикой (температура, концентрация металла) задается *регрессионным уравнением* (*регрессионной моделью*). Простейшим примером такого уравнения является линейная зависимость вида:

$$y = b_0 + b_1 x \tag{4.1}$$

где y – определяемая характеристика (также называемая *зависимой переменной*), x – измеряемая величина (*независимая переменная*), b_0, b_1 – *регрессионные коэффициенты.*

Если значения коэффициентов b_0 и b_1 известны, то, измерив независимую переменную x, по регрессионному уравнению 4.1 можно вычислить значение определяемой характеристики y в исследуемых образцах. Этот процесс называется *прогнозированием.* Точность прогноза является главным критерием качества модели.

Установление значений регрессионных коэффициентов проводится в ходе процедуры, называемой *градуировка* (*калибровка*). В англоязычной литературе по хемометрике принят термин *calibration*, в то время как в российской литературе по аналитической химии наиболее распространен термин градуировка, который мы и будем использовать в этой главе. Смысл процедуры от этого не меняется, и то и другое сводится к определению значений коэффициентов регрессионной модели, которые наиболее хорошо описывают связь между x и y, а значит, позволяют делать наилучший прогноз.

Для нахождения регрессионных коэффициентов необходимо иметь набор образцов с известными значениями обеих переменных: независимой (x) и зависимой (y). Такой набор образцов называется *градуировочным*, или *обучающим*. На первый взгляд задача представляется крайне простой: возьмем два образца для которых известны обе характеристики, x и y, составим систему двух уравнений с двумя неизвестными (b_0 и b_1); решим эту систему и получим искомые коэффициенты.

В идеальном мире, где можно абсолютно точно измерить значения независимой переменной x и знать абсолютно точные значения y, такой подход бы хорошо работал. Однако, в реальном мире любые физические величины (в нашем случае x и y) содержат определенные погрешности — случайные отклонения измеренных значений от истинных. У каждого образца эти отклонения индивидуальны. Поэтому нахождение коэффициентов по двум парам значений приведет к существенным погрешностям в регрессионных коэффициентах, и, как следствие, к низкой точности вычислений определяемого параметра y в новых образцах. На рис. 4.1 изображен как раз такой случай: линейная зависимость проведенная по двум синим точкам существенно отличается от действительного вида линейной зависимости в случае использования всей выборки (серая линия).

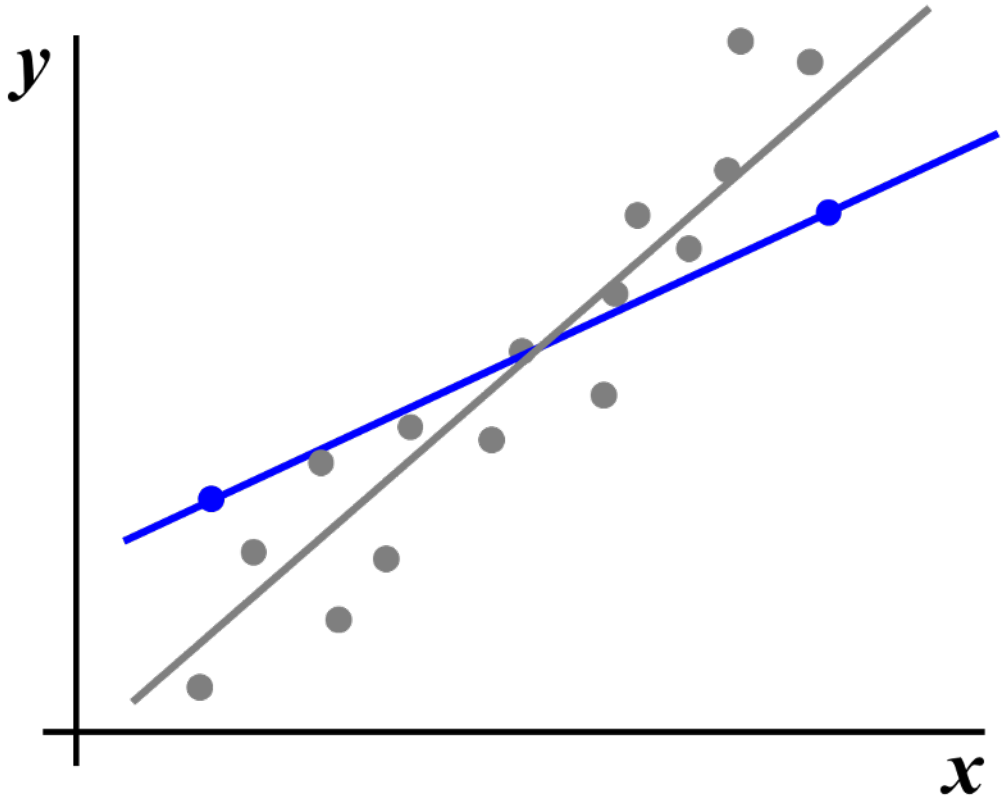


Рис. 4.1. Линейные зависимости, построенные по всем точкам выборки (серый цвет) и только по двум (синий цвет).

На практике, чтобы минимизировать такие погрешности, используют заведомо большее число образцов,

чем число регрессионных коэффициентов. В этом случае значения регрессионных коэффициентов вычисляют с помощью различных статистических методов, таких как, например, метод наименьших квадратов, или метод максимального правдоподобия, которые позволяют минимизировать влияние случайных отклонений. Для оценки качества регрессионных моделей используют две основные величины: коэффициент детерминации R^2 и среднеквадратичное отклонение (СКО), рассчитанные для известных и спрогнозированных значений определяемой характеристики y.

Коэффициент детерминации показывает процент дисперсии y, который может объяснить данная модель. Например, если $R^2 = 0.90$, это означает, что 90% дисперсии y можно объяснить с помощью данной модели, зная значения x. Соответственно 10% — это величина, связанная с относительной ошибкой прогнозирования. Очевидно, что для хорошей модели R^2 должен быть как можно ближе к единице.

Вторую величину, СКО (в англоязычной литературе *root-mean-square error*, RMSE), рассчитывают по уравнению:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_i^{pred} - y_i^{real})^2}{n}} \tag{4.2}$$

где y_i^{pred} и y_i^{real} – спрогнозированное по регрессионной модели значение определяемой величины y и ее истинное значение для i-го стандартного образца, соответственно; n – число образцов в обучающем наборе. Хорошая модель дает точный прогноз, а значит спрогнозированные моделью значения откликов будут ближе к реальным, соответственно значение RMSE будет ближе к нулю а значение коэффициента детерминации будет ближе к единице. Подробно про методы оценки качества моделей (*методы валидации*) будет рассказано ниже.

Очевидно, что зависимость между переменными может быть и нелинейной, в этом, случае, как правило, используют полиномиальные уравнения вида:

$$y = b_0 + b_1 x + b_2 x^2 + ... + b_n x^n \tag{4.3}$$

где b_0, b_1 и т.д. – регрессионные коэффициенты при соответствующих членах полинома. Чем старше степень полинома, тем более сложные виды нелинейности способно учесть уравнение. Однако, в этом случае число регрессионных коэффициентов, которые требуется оценить, будет тоже больше, по сравнению с линейной моделью, а значит для построения надежной модели необходимо большее количество градуировочных образцов. На практике выбор степени полинома зависит от решаемой задачи. С одной стороны регрессионная модель должна как можно более точно описывать зависимость переменных между собой, с другой – не должна быть слишком сложной или *переобученной*. На рис. 4.2 показан пример нелинейной зависимости между x и y для обучающего набора из шести образцов. Точками

обозначены экспериментальные значения, полученные для обучающего набора, сплошными линиями – регрессионные модели.

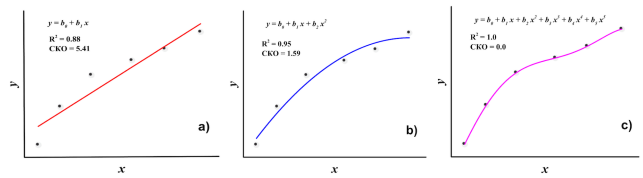


Рис. 4.2. Описание одного набора данных различными регрессионными уравнениями.

Если экспериментальные значения описать линейной зависимостью (рис. 4.2a.), то качество такой модели будет не вполне удовлетворительным (невысокое значения R^2 и большое СКО). Кроме того, из графика видно, что экспериментальные точки “плохо ложатся” на регрессионную кривую. Такая модель называется *недообученной* (англ. *underfitted*). С другой стороны, полином пятой степени (рис. 4.2c) идеально описывает экспериментальные данные ($R^2 = 1$, СКО = 0), однако, применение подобной модели для определения y в новых образцах приведет к большим погрешностям рис. 4.3. Такая модель называется *переобученной* (англ. *overfitted*). Полином второй степени (рис. 4.2b) представляется в этом случае разумным компромиссом между двумя крайностями и позволяет получать значения искомого параметра в новых образцах с максимальной точностью.

На рис. 4.3 зеленым цветом показаны новые образцы. Видно, что точность прогноза по линейной модели и полиному пятой степени будет хуже, чем для квадратичной модели (зеленые точки находятся ближе к синей кривой). В данном примере степень полинома является параметром модели, который необходимо оптимизировать на этапе ее построения.

Весьма часто использование одной независимой переменной не позволяет получить регрессионные модели хорошего качества. Такая ситуация возникает, например, если в значение искомой характеристики вносят вклад одновременно несколько факторов. Например, при определении ионов металла в растворе по интенсивности поглощения света на определенной длине волны, на этой же длине волны поглощают излучение и другие компоненты раствора.

В этом случае возможны два выхода. Первый предполагает физическое устранение мешающего влияния других компонентов (разделение компонентов анализируемого раствора, маскирование, осаждение мешающих веществ и т.п.). Во втором варианте пытаются получить дополнительные сигналы (в нашем случае интенсивность поглощения на других длинах волн) и используют не одну независимую переменную, а несколько. Тогда регрессионное уравнение примет вид:

$$y = b_0 + b_1 x_1 + b_2 x_2 + ... + b_p x_p \tag{4.4}$$

где x_1, x_2, ..., x_p – независимые переменные, b_0, b_1 и т.д. – соответствующие регрессионные коэффициенты.

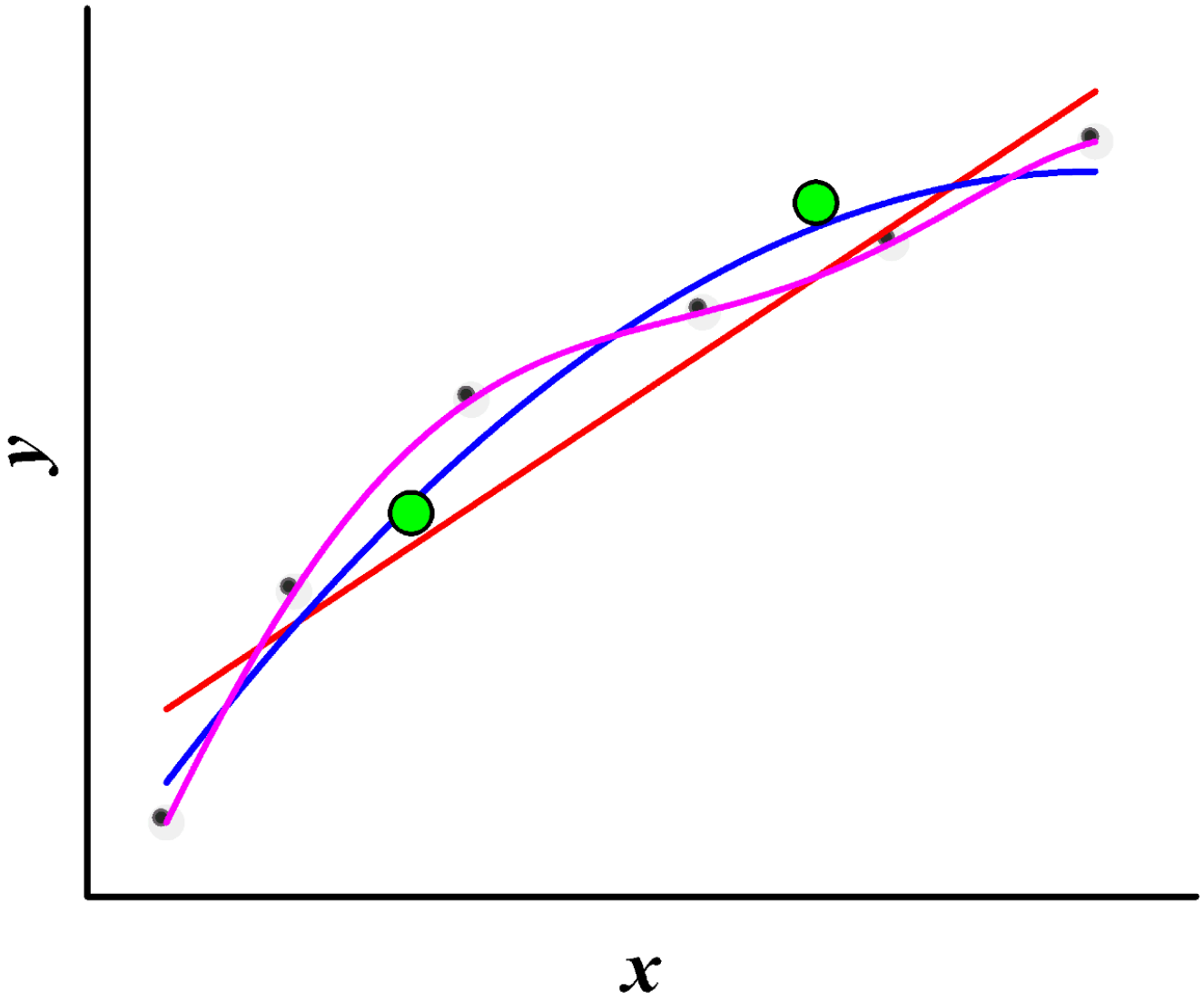


Рис. 4.3. Описание одного набора данных различными регрессионными уравнениями.

В простейшем случае такая модель называется *множественная линейная регрессия* (МЛР), однако, общему виду уравнения 4.4 удовлетворяет целый ряд других популярных хемометрических методов, например, регрессия по главным компонентам и проекции на латентные структуры.

В общем виде процесс построения, оптимизации и использования градуировочной модели представлен на рис. 4.4.

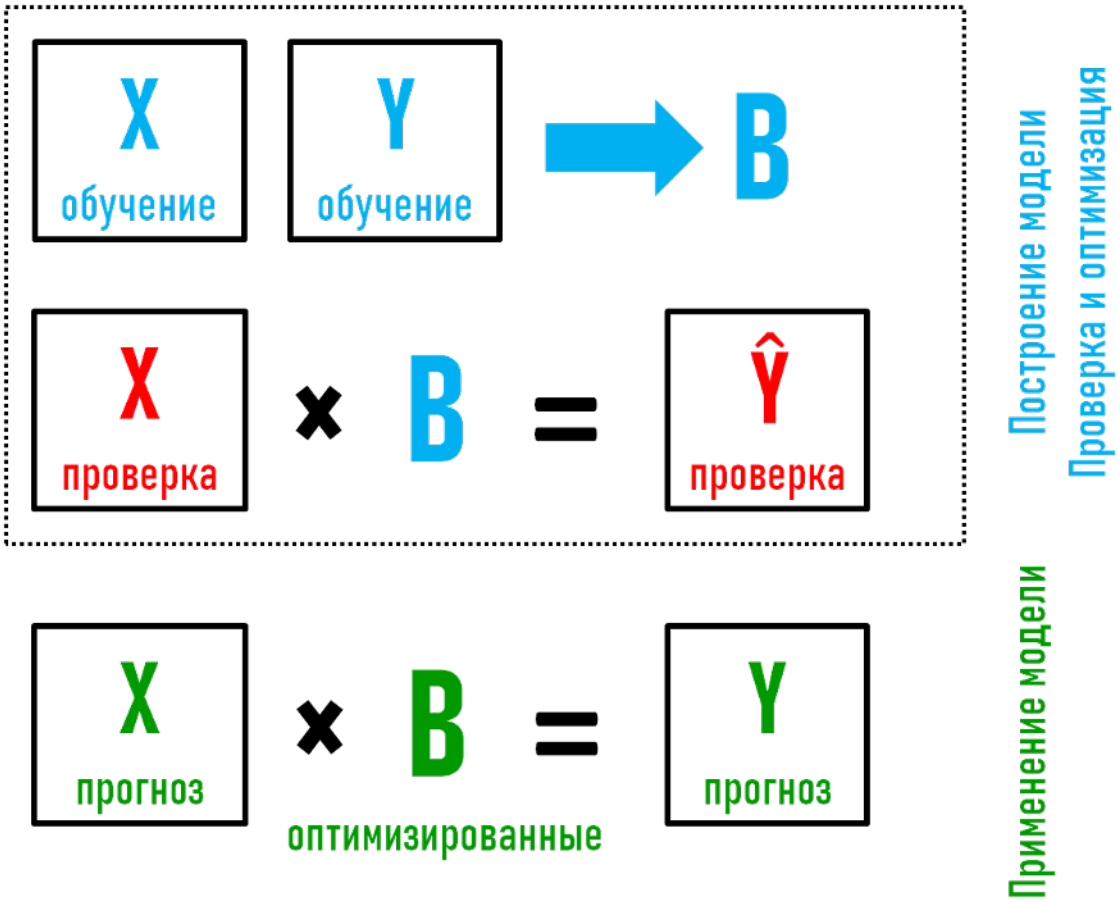


Рис. 4.4. Общая схема построения и применения регрессионных моделей.

Поскольку мы хотим получить модель, которая будет делать хороший прогноз, как для имеющихся образцов, так и для новых, то этапы построения модели и оценки ее качества нужно разделить. Более того, нужно использовать два независимых набора образцов с известными значениями X и Y для каждого этапа.

Первый набор (тренировочный, обучающий, калибровочный) используется только для градуировки — нахождения коэффициентов модели по известным значениям X и Y (обозначенные как «X обучение» и «Y обучение» на рис. 4.4). Второй набор (тестовый) используется только для оценки качества прогноза полученной модели — ее тестирования или проверки. Мерой качества прогноза является ошибка, которая

вычисляется на основе значений определяемой характеристики, полученных с помощью модели («Ŷ проверка» на рис. 4.4) и её известных значений («Y проверка») в проверочных образцах.

Если модель предполагает наличие одного или нескольких параметров, то необходим еще один этап — *оптимизация*. По сути, оптимизация — это повторение этапов градуировки и тестирования для разных значений параметра. После чего выбирается то значение, которое дает минимальную ошибку прогноза, полученного для тестового набора. Такой подход позволяет избежать переобученных моделей. Часто, чтобы процесс оптимизации не влиял на оценку качества прогноза окончательной оптимальной модели, для тестирования моделей на этапе оптимизации используют не тестовый набор, а кросс-валидацию, о которой будет рассказано ниже.

4.2 Метод наименьших квадратов

В большинстве случаев для нахождения коэффициентов в регрессионных уравнениях используется *метод наименьших квадратов* (МНК). МНК стремится найти оптимальные значения регрессионных коэффициентов, минимизируя сумму квадратов отклонений спрогнозированных моделью значений зависимой переменной от истинных значений зависимой переменной y для всех образцов обучающего набора.

Рассмотрим МНК на примере простой линейной регрессии (выражение 4.1). Определим сумму квадратов отклонений зависимой переменной через функцию $f(b_0, b_1)$, зависящую от регрессионных коэффициентов b_0 и b_1:

$$f(b_0, b_1) = \sum_{i=1}^{n} (y_i^{pred} - y_i^{real})^2 = \sum_{i=1}^{n} (b_0 + b_1 x_i - y_i^{real})^2 \tag{4.5}$$

где y_i^{pred} и y_i^{real} – спрогнозированное по регрессионной модели значение определяемой величины y и ее истинное значение для i-го стандартного образца, соответственно; n – число образцов в обучающем наборе. Другими словами, функция $f(b_0, b_1)$ описывает, как ошибка прогнозирования для нашего набора данных зависит от регрессионных коэффициентов. Соответственно, нам нужно найти такие b_0 и b_1, для которых эта функция будет иметь наименьшее значение.

Так как данная функция является квадратичным полиномом с двумя переменными, b_0 и b_1, (представьте себе параболу, которая «вращается» в 3D пространстве), то такая функция точно имеет глобальный минимум. Более того, из курса школьной алгебры мы знаем, что в точке минимума производная такой функции равна нулю. Воспользуемся этим знанием.

Так как функция зависит от двух переменных, для нахождения ее минимума решается система двух уравнений, приравнивающих к нулю частные производные функции $f(b_0, b_1)$ по каждой переменной b_0 и b_1, которые после упрощения принимают вид:

$$\begin{cases} b_0 n + b_1 \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \\ b_0 \sum_{i=1}^{n} x_i + b_1 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i \end{cases} \tag{4.6}$$

Перепишем систему 4.6 в матричном виде:

$$\begin{bmatrix} n & \sum_{i=1^n} x_i \\ \sum_{i=1^n} x_i & \sum_{i=1^n} x_i^2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} \sum_{i=1^n} y_i \\ \sum_{i=1^n} x_i y_i \end{bmatrix} \tag{4.7}$$

решение этой системы относительно регрессионных коэффициентов b_0 и b_1 выглядит следующим образом:

$$\begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} n & \sum_{i=1^n} x_i \\ \sum_{i=1^n} x_i & \sum_{i=1^n} x_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1^n} y_i \\ \sum_{i=1^n} x_i y_i \end{bmatrix} \tag{4.8}$$

Для упрощения этого выражения воспользуемся вспомогательной матрицей $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ \dots & \dots \\ 1 & x_n \end{bmatrix}$$

которая содержит независимые переменные, измеренные для n образцов обучающего набора x_i. Первый столбец матрицы содержит единицы для вычисления свободного члена b_0. Тогда, по правилам линейной алгебры:

$$\begin{bmatrix} n & \sum_{i=1^n} x_i \\ \sum_{i=1^n} x_i & \sum_{i=1^n} x_i^2 \end{bmatrix} = \mathbf{X}^{\mathrm{T}} \mathbf{X}$$

а матрица

$$\begin{bmatrix} \sum_{i=1^n} y_i \\ \sum_{i=1^n} x_i y_i \end{bmatrix} = \mathbf{X}^{\mathrm{T}} \mathbf{y}$$

где $\mathbf{y}$ – вектор, содержащий значения зависимой переменной y в обучающих образцах:

$$\mathbf{y} = \begin{bmatrix} y_1^{real} \\ \dots \\ y_n^{real} \end{bmatrix}$$

Объединяя все эти выражения, можно переписать систему 4.8 в виде уравнения:

$$\mathbf{b} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{4.9}$$

где $\mathbf{b}$ – вектор искомых регрессионных коэффициентов:

$$\mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

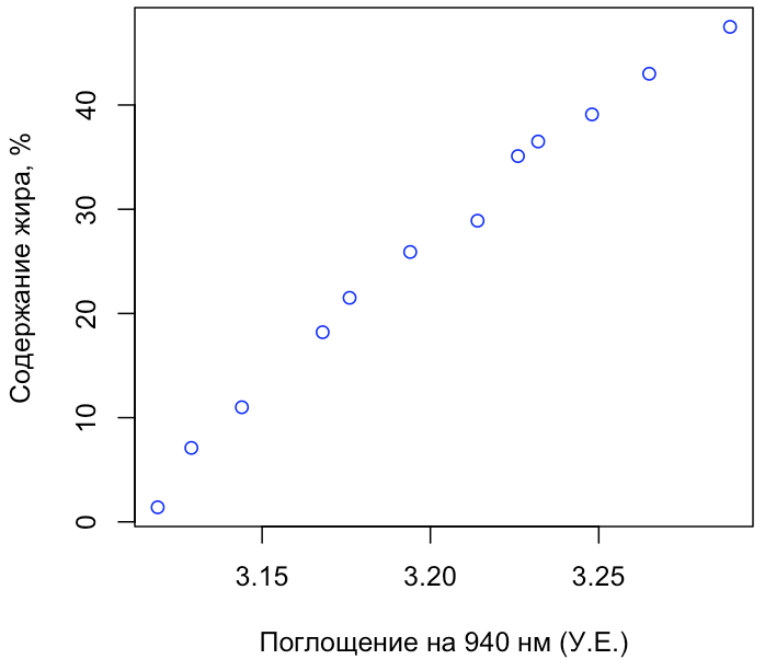
Уравнение 4.9, называемое *нормальным*, можно использовать для вычисления регрессионных коэффициентов в полиномиальных моделях (соотношение 4.3) и для моделей множественной линейной регрессии (соотношение 4.4). Для этого в матрицу $\mathbf{X}$ дописываются новые столбцы, содержащие значения других независимых переменных x, в случае МЛР, или степени x, в случае полиномиальной регрессии.

Почему в минимизируемой функции (уравнение 4.5) используются квадраты отклонений, и можно ли использовать вместо них их абсолютные значения? Абсолютные отклонения вместо квадратов использовать, разумеется, можно, как, впрочем, и любые другие функции, оценивающие близость спрогнозированными и известными значениями определяемого параметра y. Такие методы использовались математиками в XVIII веке в период зарождения регрессионного анализа. Однако, в современной математической статистике использование МНК стало практически абсолютным стандартом регрессионного анализа данных из-за простоты вычислений.

Одномерная регрессия в R

Рассмотрим как рассчитывать регрессионные коэффициенты для одномерных регрессионных моделей в R. Для начала введем необходимые данные, для этого будем использовать измерения сделанные для 12 образцов мясного фарша с разным содержанием жира. В качестве независимой переменной, x, возьмем коэффициент поглощения, измеренный на длине волны 940 нм, а в качестве зависимой переменной, y, — содержание жира в виде процента от массы образца. Блок кода ниже показывает, как ввести данные и представить их в виде графика рассеяния.

```
x <- c(3.119, 3.129, 3.144, 3.168, 3.176, 3.194, 3.214, 3.226, 3.232, 3.248,
       3.265, 3.289)
y <- c(1.4, 7.1, 11.0, 18.2, 21.5, 25.9, 28.9, 35.1, 36.5, 39.1, 43.0, 47.5)

plot(x, y, col = "blue",
   xlab = "Поглощение на 940 нм (У.Е.)",
   ylab = "Содержание жира, %"
)
```



Как можно видеть из графика, переменные коррелируют друг с другом. Можно заметить, что точки довольно хорошо лежат вдоль некоторой прямой. С другой стороны, можно также заметить небольшую выпуклость точек посередине, т.е. возможно полином второй степени также будет разумным решением.

Для построения одномерной линейной и полиномиальной, как и для многомерной регрессии в R используется одна и также функция: `lm()`. Для того, чтобы задать соотношение между переменными внутри этой функции используются формулы, которые мы рассматривали коротко в примерах R кода в главе 1 (например, для дисперсионного анализа).

Результат градуировки сохраняется в специальную переменную, которая хранит как саму модель (в виде регрессионных коэффициентов), так и различную дополнительную информацию, например, спрогнозированные значения для откликов, статистику прогноза (R^2 и другие величины), а так же результат тестирования регрессионных коэффициентов на значимость.

Вот пример построения линейной модели и вывода основной информации о ней.

```
m <- lm(y ~ x)
summary(m)
```

```
Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-3.01084 -0.74267  0.01318  1.43239  1.93621

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -833.724     30.306  -27.51 9.33e-11 ***
x            268.719      9.468   28.38 6.86e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.713 on 10 degrees of freedom
Multiple R-squared:  0.9877,    Adjusted R-squared:  0.9865
F-statistic: 805.5 on 1 and 10 DF,  p-value: 6.859e-11
```

Информация о модели разделена на несколько блоков. Первый блок *Call* просто содержит команду с помощью которой была создана эта модель. Второй блок *Residuals* содержит статистическое описание остатков (ошибок предсказания) — разницы между спрогнозированными и истинными значениями отклика, $e = y^{real} - y^{pred}$. В этом блоке показаны максимальная и минимальная ошибки (в нашем случае -3.01084 и 1.93621) а также медиана, первый и третий квартили.

Блок *Coefficients* самый большой, он содержит регрессионные коэффициенты и результаты их тестирования на значимость. Вся информация представлена в виде таблицы, где столбец *Estimated* содержит, собственно, сами коэффициенты: *Intercept* (b_0) и коэффициент влияния независимой переменной на зависимую (b_1). В нашем случае $b_0 = -833.724$ и $b_1 = 268.719$, а сама модель выглядит как:

$$y = -833.724 + 268.719x$$

Следующие три столбца содержат результаты применения теста Стьюдента (мы о нем подробно говорили в главе 1) для каждого коэффициента. В данном случае тестировалась гипотеза, что соответствующий коэффициент для популяции равен нулю (т.е. в случае b_1 это означает, что коэффициент поглощения на 940 нм и содержание жира в образцах никак не связаны). Математически для коэффициента b_1 это можно записать как: $H0 : \beta_1 = 0$, где β_1, это значение этого коэффициента для популяции, мы его получим если построим модель не для 12 образцов, а для 12 миллиардов (точнее для бесконечного числа образцов).

Из таблицы видно, что вероятности того, что значения коэффициентов b_0, b_1 были получены случайно, а на самом деле для популяции они нулевые ($\beta_0 = 0$, $\beta_1 = 0$), очень низкая. Другими словами мы можем констатировать, что регрессионные коэффициенты на самом деле описывают систематическую линейную связь между двумя переменными (т.е. являются значимыми для модели).

Следующие три блока, каждый из которых занимает одну строку, это показатели качества модели, рассчитанные для прогнозов сделанных по градуировочному набору. Так *Residual standard error* - это стандартная ошибка прогнозирования, по сути это стандартное отклонения для ошибок e.

Вторая строка содержит два значения для коэффициента детерминации, R^2, обычное и скорректированное. Скорректированное значение имеет смысл использовать в случае многомерной регрессии, так как оно зависит от соотношения числа наблюдений и числа независимых переменных.

Наконец, последняя строка показывает результат F-теста, по сути, это — дисперсионный анализ качества модели — насколько предсказанные значения значимо отличаются от простой модели, которая будет всегда прогнозировать среднее значения отклика независимо от значений x. Опять же, мы можем видеть, что наша модель значительно отличается от такой примитивной модели, F-метрика составляет 805.5 (если модель слабая, то она не будет превышать 3-4).

Как было написано выше, объект `m`, в который функция `lm()` сохранила результат, содержит много разных статистик и вычислений. Например, предсказанные значения для откликов:

```
show(m$fitted)
```

```
       1         2         3         4         5         6         7         8 
 4.410840  7.098032 11.128819 17.578080 19.727833 24.564779 29.939162 33.163792 
       9        10        11        12 
34.776107 39.075614 43.643840 50.093101 
```

Теперь мы можем, например, посчитать RMSE или R^2 и сравнить последний с тем, что получили в таблице выше:

```
# вычисляем остатки (ошибки предсказания)
e <- y - m$fitted

# считаем сумму квадратов ошибок
SSe <- sum(e^2)

# считаем сумму квадратов отклонений y от среднего
SSy <- sum( (y - mean(y))^2 )
```

```
# считаем RMSE
RMSE <- sqrt(SSe / length(x))

# считаем R2
R2 <- 1 - SSe/SSy

# показываем обе статистики
show(c(RMSE = RMSE, R2 = R2))
```

```
    RMSE        R2
1.5634308 0.9877373
```

Как можно видеть, рассчитанное нами самостоятельно значение для R^2 совпадает с табличным.

Можно также делать прогнозы для новых значений x, например, из тестового набора. Для этого используется функция `predict()`.

```
xnew <- c(3.15, 3.16, 3.20)
ynewp <- predict(m, data.frame(x = xnew))
show(ynewp)
```

```
       1        2        3
12.74113 15.42833 26.17709
```

Обратите внимание, что не смотря на то, что при построении модели мы использовали простые вектора, а не фреймы данных, новые данные все равно нужно предоставлять в виде фрейма. Названия столбцов в этом фрейме должны совпадать с названиями переменных, которые использовались в формуле при градуировке модели.

Наконец, попробуем найти коэффициенты самостоятельно, используя матричную форму МНК, воспользовавшись формулой 4.9:

```
# формируем матрицу X соединяя столбец x со столбцом единичек
n <- length(x)
X <- cbind(rep(1, n), x)

# вычисляем коэффициенты по формуле 5.10
b <- solve(crossprod(X)) %*% crossprod(X, y)
```

```
show(b)
```

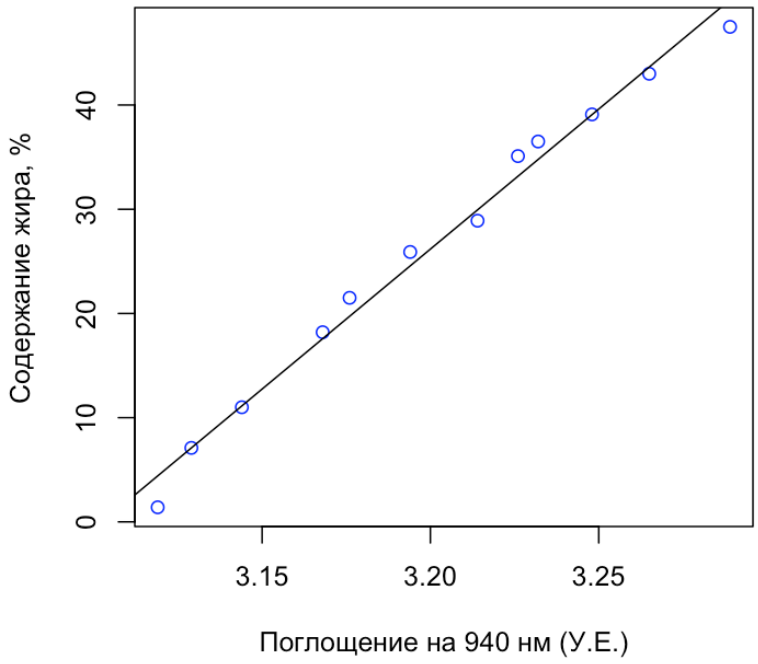
```
      [,1]
  -833.7243
x  268.7192
```

Здесь мы используем знакомые нам по главе 1 функции `crossprod()`, которая вычисляет "внутреннее" скалярное произведение ($\mathbf{A}^T\mathbf{B}$) и `solve()`, которая в данном случае вычисляет обратную матрицу.

Как можно видеть, вычисленные таким образом коэффициенты совпадают с полученными с помощью `lm()`.

Полученную модель можно добавить к графику рассеяния в виде линии. Это можно сделать с помощью функции `abline()` использовав объект с моделью в качестве аргумента, как показано в коде ниже (здесь мы подразумеваем, что этот объект вы уже получили, запустив предыдущие блоки кода):

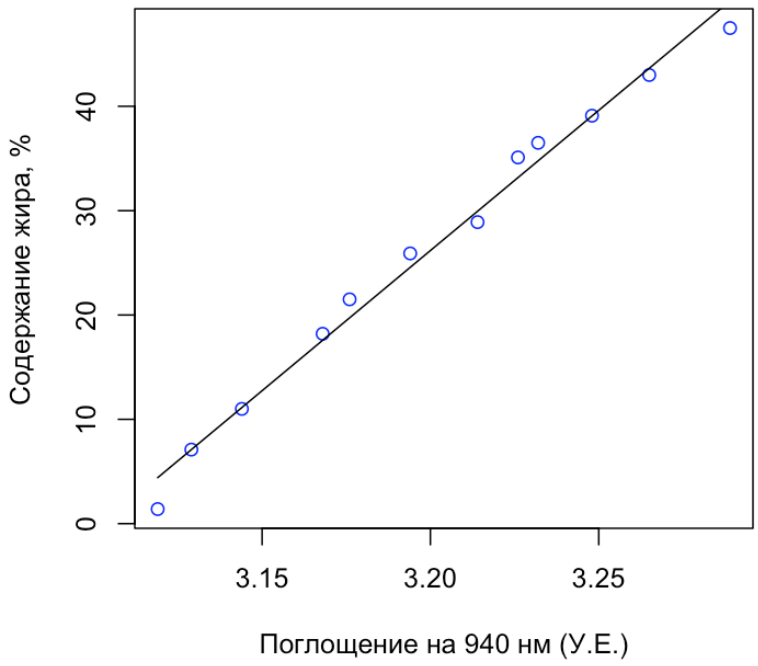
```
plot(x, y, col = "blue",
   xlab = "Поглощение на 940 нм (У.Е.)",
   ylab = "Содержание жира, %"
)
abline(m)
```



Однако такой способ будет работать только для линейных моделей. Код ниже показывает, как получить такой же результат другим способом. Он более длинный, но зато подходит и для полиномиальных моделей.

```
# генерируем 100 значений x равноудаленных друг от друга
xm <- seq(min(x), max(x), length.out = 100)

# для каждого сгенерированного значения вычисляем предсказанное значение отклика
ym <- predict(m, data.frame(x = xm))

# строим исходный график
plot(x, y, col = "blue",
   xlab = "Поглощение на 940 нм (У.Е.)",
   ylab = "Содержание жира, %"
)

# добавляем вычисленные точки соединенные короткими линиями
```

```
lines(xm, ym)
```



Как видим, результат похож, за исключением того, что линия теперь не выходит за пределы исходных данных.

В заключение покажем, как сделать градуировку полиномиальной модели. Для этого воспользуемся той же функцией, `lm()`, но в качестве независимых аргументов зададим не `x`, а его полином второй степени. Для простоты это можно сделать также через функцию, в данном случае `poly()`.

Код ниже показывает вычисление модели с использованием полинома второй степени.

```
m2 <- lm(y ~ poly(x, 2))
summary(m2)
```

```
Call:
lm(formula = y ~ poly(x, 2))

Residuals:
```

```
     Min       1Q   Median       3Q      Max
-2.30328 -0.20462  0.03976  0.74843  1.17646

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  26.2667     0.3047  86.201 1.93e-14 ***
poly(x, 2)1  48.6067     1.0556  46.048 5.38e-12 ***
poly(x, 2)2  -4.3936     1.0556  -4.162  0.00244 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.056 on 9 degrees of freedom
Multiple R-squared:  0.9958,    Adjusted R-squared:  0.9949
F-statistic:  1069 on 2 and 9 DF,  p-value: 2e-11
```

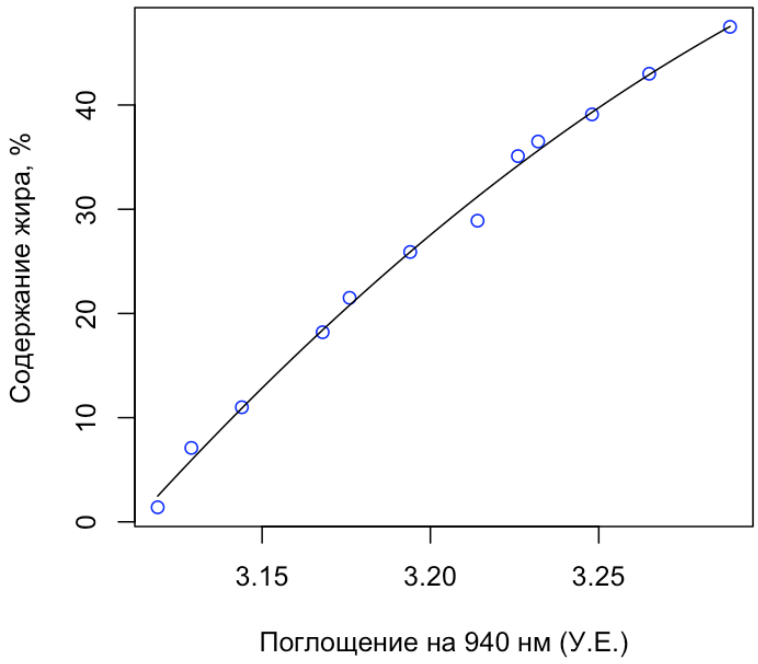
Как можно заметить, в таблице теперь три коэффициента, первые два имеют такой же смысл, как и в случае линейной модели, а третий связан с x^2, он задает степень кривизны, или выпуклости модели. Так как p-значение для теста на значимость для этого коэффициента довольно мало (0.00244), мы можем заключить, что на самом деле кривизна в данных имеется и что это не случайный эффект.

Таким образом, наша новая модель имеет следующий вид (с точностью до второго десятичного знака):

$$y = 26.27 + 48.61x - 4.39x^2$$

Покажем модель на графике:

```
xm <- seq(min(x), max(x), length.out = 100)
ym <- predict(m2, data.frame(x = xm))
plot(x, y, col = "blue",
   xlab = "Поглощение на 940 нм (У.Е.)",
   ylab = "Содержание жира, %"
)
lines(xm, ym)
```



Обратите внимание, что код практически идентичен тому, который мы использовали для визуализации линейной модели. Мы просто убрали комментарии и пустые строки и поменяли имя переменной, в которую сохранили новую модель (`m2` вместо `m`). Кривизна не очень большая, но довольно заметная, и новая модель очевидно лучше описывает общий тренд.

Повторим теперь это для полинома более высокой степени, например, четвертой.

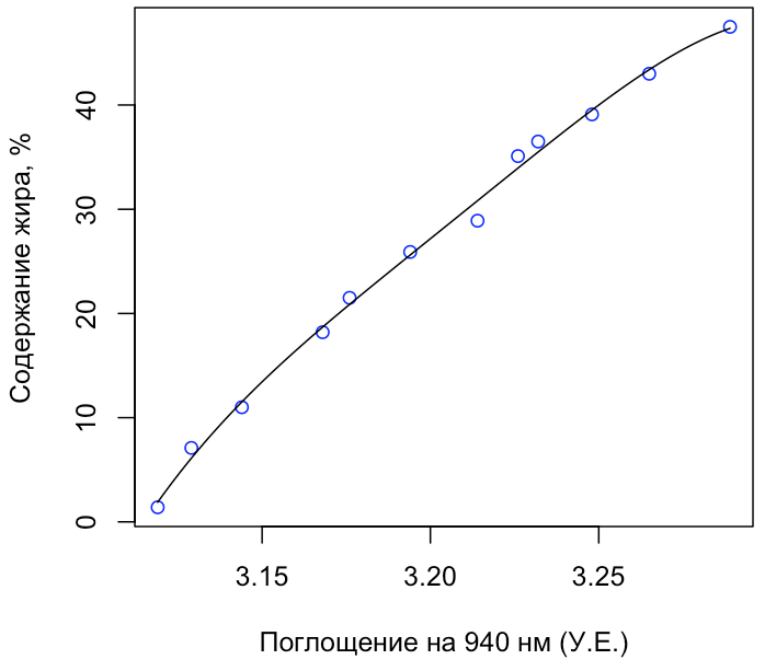
```
m4 <- lm(y ~ poly(x, 4))
summary(m4)
```

```
Call:
lm(formula = y ~ poly(x, 4))

Residuals:
    Min      1Q  Median      3Q     Max
-1.9194 -0.4907 -0.1235  0.7207  1.1576
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  26.2667     0.3197  82.158 1.04e-11 ***
poly(x, 4)1  48.6067     1.1075  43.888 8.33e-10 ***
poly(x, 4)2  -4.3936     1.1075  -3.967  0.00541 **
poly(x, 4)3   0.3607     1.1075   0.326  0.75416
poly(x, 4)4  -1.1453     1.1075  -1.034  0.33546
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.108 on 7 degrees of freedom
Multiple R-squared:  0.9964,    Adjusted R-squared:  0.9944
F-statistic: 485.8 on 4 and 7 DF,  p-value: 1.243e-08
```

Как можно видеть, формально модель стала лучше (коэффициент детерминации немного выше). Однако, коэффициенты для x^3 ($b_3 = 0.3607$) и x^4 ($b_4 = -1.1453$) не проходят тест на значимость. Другими словами, они скорее всего описывают случайные вариации в данных. Посмотрим на то, как выглядит новая модель:

```r
xm <- seq(min(x), max(x), length.out = 100)
ym <- predict(m4, data.frame(x = xm))
plot(x, y, col = "blue",
   xlab = "Поглощение на 940 нм (У.Е.)",
   ylab = "Содержание жира, %"
)
lines(xm, ym)
```



Можно заметить, что кривая имеет более сложную форму, меняя кривизну несколько раз. В целом, результат, полученный выше, очень похож на тот, что показан на рис. 4.2. Линейная модель в данном примере является недоопределенной, модель с полиномом четвертой степени — переопределенной, а квадратичная модель имеет оптимальную сложность.

4.3 Методы многомерной регрессии

Как мы писали выше, для оценки определяемого параметра одну независимую переменную можно использовать, только если на нее не накладываются другие сигналы. Рассмотрим это на примере спектральных данных. Пусть в наших образцах присутствует только определяемый компонент, который дает спектры, приведенные на рис. 4.5. Для построения градуировочной модели достаточно взять значения спектральной интенсивности на одной определенной длине волны x (показано вертикальной черной линией). Как правило, в таких случаях регрессионная модель (график-вставка на рис. 4.5) имеет хорошее качество.

Если в образцах помимо определяемого компонента имеются мешающие (красные линии на Рис. 4.6a), спектры которых перекрываются с целевым, то регистрируемый спектр будет искажен вкладом от мешающего компонента (Рис. 4.6b). В этом случае градуировка по одной переменной (график-вставка на Рис. 4.6b) будет иметь низкое качество. Очевидным путем решения данной проблемы является устранение

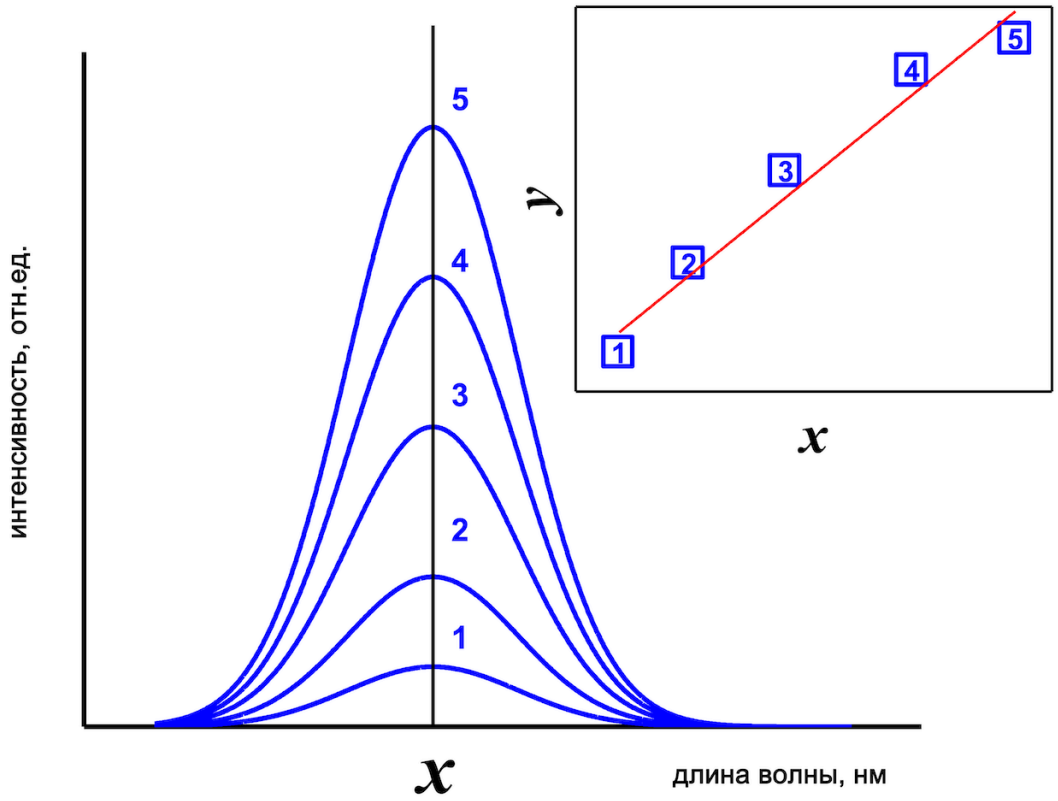


Рис. 4.5. Использование одномерной градуировки.

мешающего компонента из анализируемых образцов путем осаждения, маскирования и т.п.. Однако, зачастую проведение таких операций с образцом может затронуть и анализируемый компонент, особенно, если их химические свойства близки. Кроме того, такие процедуры связаны с увеличением затрат времени, труда и расходом реактивов.

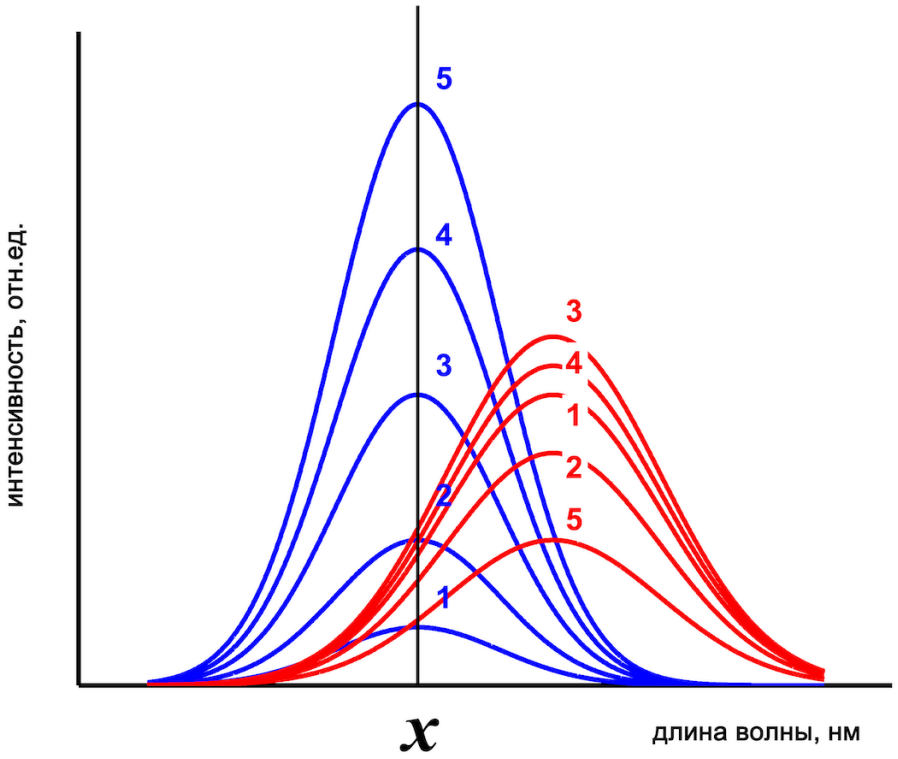
Альтернативным путем решения этой проблемы может являться применение методов многомерной регрессии. Для этого при построении градуировки используют не одну единственную интенсивность на определенной длине волны , а целый набор интенсивностей на разных p длинах волн $_1, _2, ..., _p$.

В этом случае регрессионное уравнение принимает вид 4.4, регрессионные коэффициенты можно найти, например, с помощью нормального уравнения МНК 4.9. Традиционное графическое представление градуировочной зависимости в координатах $y = f(x)$ заменяется на график «введено - найдено», на котором по оси абсцисс откладывается известное значение концентрации определяемого компонента, а по оси ординат значение концентрации, спрогнозированное по уравнению 4.4. Такая замена графического представления связана с тем, что, для регрессии используется множество переменных $(x_1, x_2, ..., x_p)$, и функцию $y = f(x_1, x_2, ..., x_p)$ необходимо отображать в $+1$ мерном пространстве, что неудобно на практике для случаев, когда число независимых переменных больше двух.

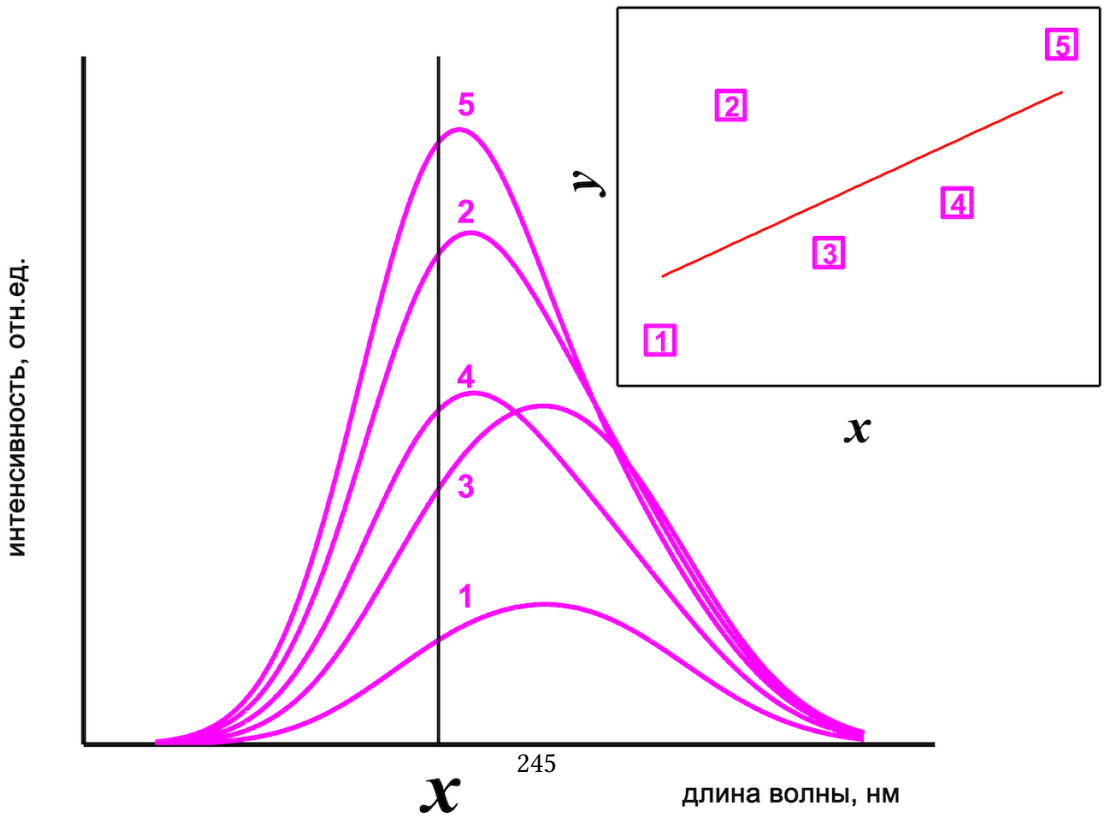
График «введено – найдено», также как и традиционный градуировочный график $y = f(x)$, позволяет судить о качестве модели. В случае идеальной градуировки этот график представляет собой прямую с тангенсом угла наклона равным единице (спрогнозированные моделью значения равны известным значениям), свободным членом равным нулю и коэффициентом корреляции также равным единице. На вставке рис. 4.7 изображен пример такого графика для хорошей модели, точки-образцы лежат практически на прямой, отображающей идеальный случай ($y^{pred} = y^{real}$). Применение нескольких переменных позволяет более эффективно и полно использовать информацию, содержащуюся в спектрах. Так, в нашем примере модель на рис. 4.6b, построенная по одной переменной, имеет гораздо более низкое качество, чем модель на рис. 4.7.

Вычисление регрессионных коэффициентов по уравнению 4.9 возможно только в случае, когда число градуировочных образцов ≥ числа переменных. Это накладывает ограничения на практическое применение метода множественной линейной регрессии (МЛР): так, например, если спектры для градуировки содержат по 150 длин волн, то для применения МЛР будет необходимо проанализировать как минимум 150 градуировочных образцов. Разумеется, это крайне трудоемко и неудобно. Современные аналитические приборы (спектрометры, хроматографы и т.д.) часто имеют высокое разрешение и регистрируют сигналы на тысячах измерительных каналов.

На практике для применения МЛР осуществляют предварительный отбор переменных. В самом простом варианте можно взять не все переменные, а с определенным шагом: каждую пятую, каждую десятую. Если имеется дополнительная информация о том, где в спектре расположены сигналы определяемого и мешающих компонентов, то можно выбрать переменные, отвечающие максимумам этих сигналов.


интенсивность, отн.ед.
5
4
3
2
1
3
4
1
2
5
x
длина волны, нм

а)


интенсивность, отн.ед.
5
2
4
3
1
y
5
2
4
3
1
x
x
длина волны, нм

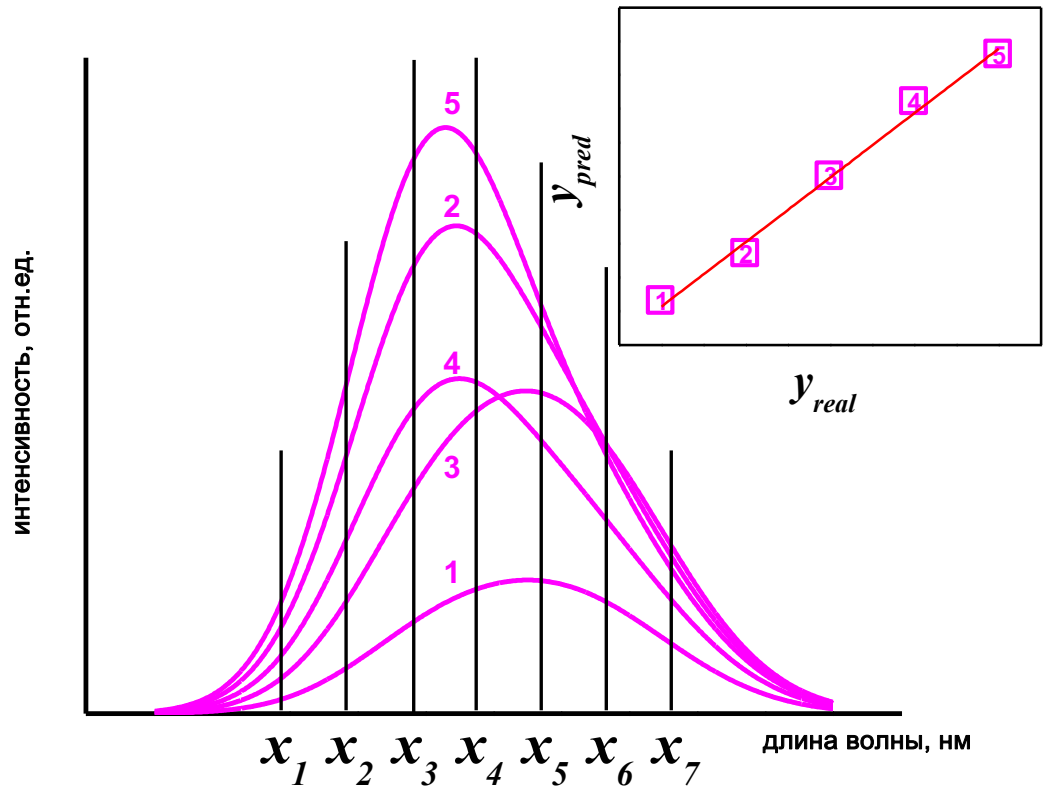


Рис. 4.7. Использование многомерной градуировки.

Очевидно, что качество модели будет зависеть от выбора конкретных переменных. Также очевидно, что часть аналитической информации при этом может теряться, однако, применение МЛР в случае наличия мешающих влияний позволяет существенно повысить качество моделей по сравнению с обычной градуировкой по одной переменной.

Использование большого числа переменных для построения регрессионной модели приводит к риску возникновения случайных корреляций между независимыми переменными и целевым параметром. Рассмотрим это на примере рис. 4.6а. Если бы концентрация мешающего компонента (красный сигнал) менялась симбатно (однонаправлено) с концентрацией основного, т.е. интенсивность бы увеличивалась с номером образца, то при построении модели даже на переменных, отвечающих только мешающему компоненту, получилась бы хорошая градуировка. Однако, ее применение для новых образцов, где соотношение основного и мешающего компонентов может быть любым, привело бы к неудовлетворительной точности прогнозирования. В связи с этим необходимо проводить тщательную поверку моделей на отсутствие таких случайных корреляций. Кроме этого, необходимо уделять особое внимание подбору градуировочных образцов и избегать наличия в них симбатно меняющихся компонентов. Помимо этого градуировочные образцы должны удовлетворять следующим требованиям: концентрации определяемого компонента в градуировочных образцах должны равномерно покрывать весь концентрационный диапазон, в пределах которого будут определяться содержания компонента; градуировочные образцы, помимо определяемого компонента, должны содержать мешающие компоненты, которые присутствуют в анализируемых образцах, если их сигналы влияют на сигналы аналита (это условие необходимо для учета взаимного влияния различных компонентов пробы). В литературе предложены различные способы дизайна наборов градуировочных образцов, удовлетворяющих этим требованиям: диагональный дизайн, циклический пермутационный дизайн, латинские гиперкубы, равномерный дизайн и т.д.

Неотъемлемой частью многомерной градуировки является ее проверка (валидация). В следующем разделе мы рассмотрим основные методы проведения валидации, но сначала поговорим как строить МЛР модели в R.

Градуировка МЛР моделей в R

В качестве примера для построения МЛР и других многомерных моделей будем использовать набор данных *Simdata*. Он доступен в пакете *mdatools*, который мы уже использовали ранее. Если вы по каким-то причинам не хотите ставить этот пакет, данные можно также скачать в виде CSV файла с GitHub репозитория этой книги. В коде ниже мы будем использовать данные из *mdatools*.

Собственно, *Simdata* представляет собой спектры с интенсивностями поглощения, измеренными на 150 длинах волн от 210 нм до 359 нм (УФ диапазон) для смесей трех углеводородов, которые мы будем обозначать как *C1*, *C2*, и *C3*. Для каждой смеси имеется также матрица с известными концентрациями каждого из составляющих.

Наборов, на самом деле, два — спектры и концентрации для калибровки/градуировки (100 смесей), и спектры и концентрации для валидации, или тестирования моделей (50 смесей).

Для начала загрузим данные из *mdatools*:

```
library(mdatools)
data(simdata)
str(simdata)
```

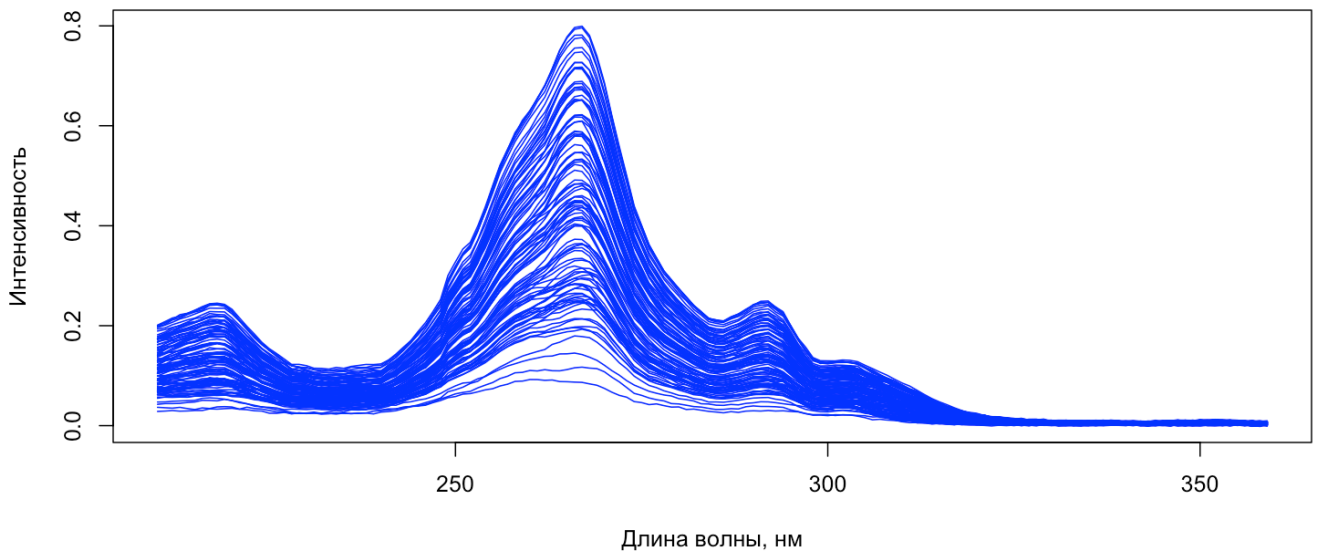
```
List of 5
 $ conc.c    : num [1:100, 1:3] 0.905 0.763 0.363 0.278 0.53 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:100] "T1" "T2" "T3" "T4" ...
  .. ..$ : chr [1:3] "C1" "C2" "C3"
 $ spectra.c : num [1:100, 1:150] 0.146 0.162 0.134 0.104 0.168 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:100] "T1" "T2" "T3" "T4" ...
  .. ..$ : chr [1:150] "X210" "X211" "X212" "X213" ...
 $ spectra.t : num [1:50, 1:150] 0.1105 0.0465 0.2053 0.1108 0.0238 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:50] "V1" "V2" "V3" "V4" ...
  .. ..$ : chr [1:150] "X210" "X211" "X212" "X213" ...
 $ wavelength: int [1:150] 210 211 212 213 214 215 216 217 218 219 ...
 $ conc.t    : num [1:50, 1:3] 0.3733 0.1204 0.9346 0.3097 0.0193 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:50] "V1" "V2" "V3" "V4" ...
  .. ..$ : chr [1:3] "C1" "C2" "C3"
```

Как можно видеть, этот набор содержит список (list) с пятью переменными:

Переменные `simdata$conc.c` (матрица размером 100 × 3) и `simdata$spec.c` (матрица размером 100 × 150) — это известные концентрации и измеренные спектры для градуировочного набора, а переменные `simdata$conc.t` (матрица размером 50 × 3) и `simdata$spec.t` (матрица размером 50 × 150) — содержат концентрации и спектры для тестового набора. Вектор `simdata$wavelength` содержит значения длин волн в нанометрах для каждого столбца матриц со спектрами.

Посмотрим для начала на спектры градуировочного набора:

```
matplot(
   simdata$wavelength, t(simdata$spectra.c),
   type = "l", lty = 1, col = "blue",
   xlab = "Длина волны, нм", ylab = "Интенсивность"
)
```



Для построения МЛР модели необходимо, чтобы число независимых переменных (столбцов **X**) было меньше числа измерений (строк **X**). Кроме этого, требуется отсутствие сильных корреляций между столбцами.

Для того чтобы решить обе проблемы, мы будем использовать не полные, а прореженные спектры выбрав каждый десятый столбец. Надо отметить, что проблема с корреляцией в этом случае решится лишь частично, так как мы не знаем заранее, какие из спектральных пиков коррелируют друг с другом. Подготовим матрицу с прореженными спектрами:

```
# генерируем индексы столбцов которые нужно выбрать
colind <- seq(2, 150, by = 10)

# выбираем столбцы с сгенерированными индексами
Xc <- simdata$spectra.c[, colind]
```

В качестве откликов будем использовать концентрацию третьей составляющей, *C3*.

```r
# подготовим вектор с откликами выбрав третий столбец из матрицы концентраций
yc <- simdata$conc.c[, 3]
```

Теперь можно построить МЛР модель:

```r
# построим МЛР модель
m <- lm(yc ~ ., data = as.data.frame(cbind(yc, Xc)))
summary(m)
```

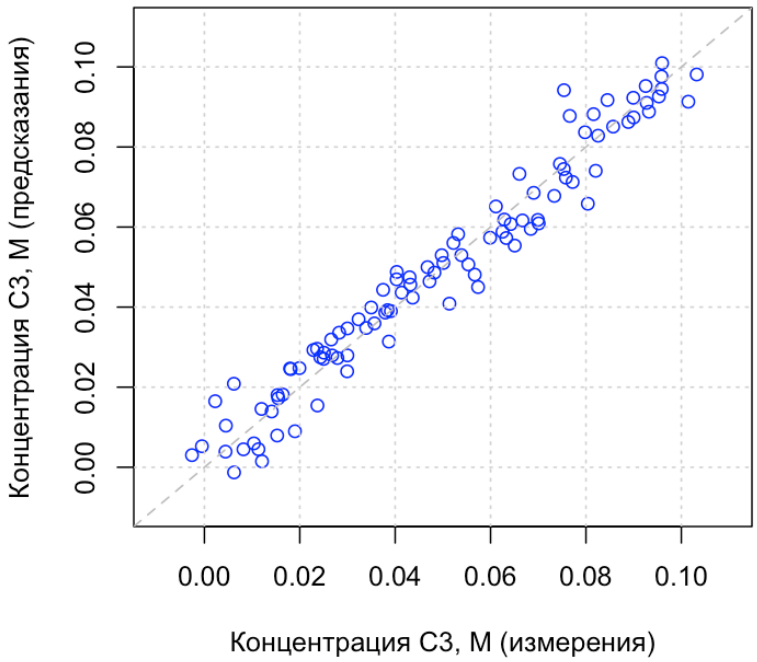
```
Call:
lm(formula = yc ~ ., data = as.data.frame(cbind(yc, Xc)))

Residuals:
       Min         1Q     Median         3Q        Max
-0.0187879 -0.0044746 -0.0003321  0.0044058  0.0145701

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.001242   0.002535   0.490  0.62544
X211        -0.134578   0.803175  -0.168  0.86733
X221         0.280299   0.663430   0.423  0.67374
X231         1.188005   0.724123   1.641  0.10462
X241         1.005483   0.629436   1.597  0.11392
X251         1.583716   0.621726   2.547  0.01268 *
X261        -0.357550   0.632831  -0.565  0.57358
X271        -1.248140   0.442583  -2.820  0.00599 **
X281         0.467185   0.727590   0.642  0.52256
X291        -0.241707   0.743430  -0.325  0.74590
X301         0.685878   0.640956   1.070  0.28765
X311         1.066448   0.392819   2.715  0.00805 **
X321        -0.410706   0.725033  -0.566  0.57259
X331         0.073093   0.741565   0.099  0.92172
X341        -0.094069   0.815851  -0.115  0.90848
X351        -0.442079   0.706782  -0.625  0.53335
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.006643 on 84 degrees of freedom
Multiple R-squared:  0.9546,    Adjusted R-squared:  0.9465
F-statistic: 117.8 on 15 and 84 DF,  p-value: < 2.2e-16
```

Обратите внимание, что для построения модели нам пришлось соединить вектор с откликами, `yc` и матрицу со спектрами, `Xc`, в единый фрейм с данными. Кроме этого формула для модели выглядит как `yc ~ .`. Такой короткий способ записи означает, что все столбцы фрейма, кроме `yc` будут использоваться в качестве независимых переменных.

Как можно видеть, в таблице теперь 16 коэффициентов (15 для выбранных столбцов и b_0). Только три из них прошли тест на значимость. Построим график "введено-найдено" для градуировочного набора:

```
plot(yc, m$fitted,
   xlim = c(-0.01, 0.11),
   ylim = c(-0.01, 0.11),
   xlab = "Концентрация C3, M (измерения)",
   ylab = "Концентрация C3, M (предсказания)",
   col = "blue"
)
grid()
abline(a = 0, b = 1, lty = 2, col = "gray")
```



В целом, как можно видеть из графика и статистик, модель получилась неплохая с $R^2 \approx 0.95$, однако, для того, чтобы оценить реальное качество модели необходимо ее валидировать, о чем и пойдет речь в следующем разделе.

4.4 Методы валидации моделей

Для оценки качества модели используют два основных подхода: проверка *тестовым набором* (англ. *test set validation*) и *перекрестная проверка* (англ. *cross-validation*). Метод проверки тестовым набором является предпочтительным, однако, требует наличия достаточного количества образцов с известными значениями определяемой характеристики *y*. В этом случае с помощью модели прогнозируют свойства образцов тестового набора, сравнивают полученные значения с известными, и рассчитывают значения RMSEP (среднеквадратичная ошибка прогнозирования, root mean square error of prediction) по уравнению Equation 4.2, где y_i^{pred} и y_i^{real} – предсказанное по регрессионной модели и известное значение определяемой величины y в проверочных образцах. Чем ниже величина RMSEP, тем более точно модель прогнозирует свойства образцов проверочного набора. RMSEP имеет размерность y и оценивает среднюю погрешность определения y в новых образцах. Значение RMSEP всегда необходимо приводить вместе с диапазоном, в котором изменяется y в проверяемой модели Так, например, величина RMSEP, равная 2 г/л для модели построенной в диапазоне концентраций от 1 до 4 г/л будет означать модель с очень низкой точностью

прогнозирования, в то время как такая же величина RMSEP для диапазона от 20 до 150 г/л будет означать модель хорошего качества.

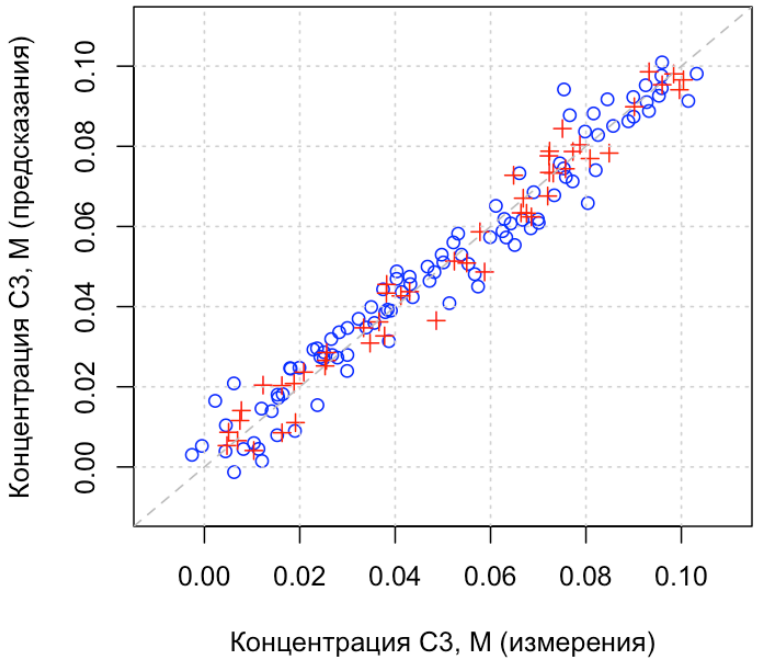
Если мы вернемся к примеру с набором *Simdata*, который мы использовали для примеров с R выше, то можно вспомнить, что он содержит и градуировочный и тестовый набор данных. В этом случае можно использовать тестовый набор, чтобы получить спрогнозированные значения откликов и сравнить их с референтными, т.е. провести валидацию модели с помощью тестового набора. Код ниже показывает как это сделать (он подразумевает, что перед этим вы запустили блоки кода из предыдущего раздела):

```
# выбираем столбцы с сгенерированными индексами из тестового набора
Xt <- simdata$spectra.t[, colind]

# подготовим вектор с откликами выбрав третий столбец из тестовой матрицы концентраций
yt <- simdata$conc.t[, 3]

# вычисляем предсказанные значения y
ytp <- predict(m, newdata = as.data.frame(Xt))

# вычисляем ошибку и коэффициент детерминации
e <- yt - ytp
SSe <- sum(e^2)
SSy <- sum( (yt - mean(yt))^2 )
R2 <- 1 - SSe/SSy
show(R2)
```

```
[1] 0.9724119
```

```
# показываем график введено-посчитано для калибровочного набора
plot(yc, m$fitted,
   xlim = c(-0.01, 0.11),
   ylim = c(-0.01, 0.11),
   xlab = "Концентрация C3, M (измерения)",
   ylab = "Концентрация C3, M (предсказания)",
   col = "blue"
)

# добавляем точки для тестового набора
points(yt, ytp, col = "red", pch = 3)
```

```
grid()
abline(a = 0, b = 1, lty = 2, col = "gray")
```



Обычно тестовый набор представляет собой данные полученные либо для нового набора образцов, которые были отобраны независимо от градуировочных, либо для случайной выборки образцов из градуировочного набора. Однако, иногда получение новых образцов затруднительно, а градуировочный набор содержит небольшое их число, достаточное лишь для построения модели. В этом случае использовать проверку тестовым набором не получится, поскольку это потребует уменьшения количества образцов в градуировочном наборе, что приведет к снижению качества градуировки. В таких ситуациях прибегают к *перекрестной* проверке. Это группа методов, общая идея которой основана на поочередном исключении образцов из калибровочного набора и использования их для проверки.

Предельным случаем является *полная перекрестная проверка* (англ. *full cross-validation*) – в этом варианте на каждом шаге из градуировки исключается один образец, рассчитывается модель без него, затем эта модель используется для прогнозирования y в исключенном образце. Эта процедура повторяется для каждого образца, после чего по уравнению Equation 4.2 рассчитывают величину *RMSECV* – среднеквадратичное отклонение перекрестной проверки. В этом случае y_i^{pred} – значение определяемой

величины, предсказанное по регрессионной модели для исключенных образцов. В англоязычной литературе такой способ проверки также часто называют *leave-one-out cross-validation*.

На Рис. 4.8 представлена схема проведения такой проверки. Из исходной модели (Рис. 4.8a) на первом этапе выбирают один образец обучающего набора (покрашен красным на Рис. 4.8b) и строят модель без него (Рис. 4.8c). Затем этот образец используется в качестве тестового образца (синяя точка на Рис. 4.8c) и для него рассчитывается значение ошибки прогнозирования. Дальше этот образец возвращается обратно в обучающий набор, а процедура исключения и прогнозирования повторяется для следующего образца (Рис. 4.8e,d). Эту последовательность повторяют, пока не переберут все образцы обучающей выборки. На основе рассчитанных значений ошибки прогнозирования для исключенных образцов вычисляют RMSE перекрестной проверки RMSECV (root mean squared error of cross-validation). Обратите внимание – на Рис. 4.8a, b и d приведена одна и та же исходная модель, а красным цветом показаны образцы, которые будут исключены на соответствующем этапе перекрестной проверки.

Если на каждом этапе исключать не один образец, а несколько, то такая проверка называется *сегментарной перекрестной проверкой*. Размер и количество сегментов зависят от количества доступных образцов, а сами образцы можно выбирать случайным образом, либо систематически – например, выбирая в проверочный набор каждый второй, или каждый третий образец, и т.д.. Разумеется, размер сегмента надо выбирать так, чтобы его исключение существенно не повлияло на качество модели. Часто прибегают к перекрестной проверке по методу Монте-Карло – в этом случае проводят многократное случайное разбиение имеющихся образцов на обучающую и проверочную выборки, а полученные в результате метрики усредняют по всем выборкам.

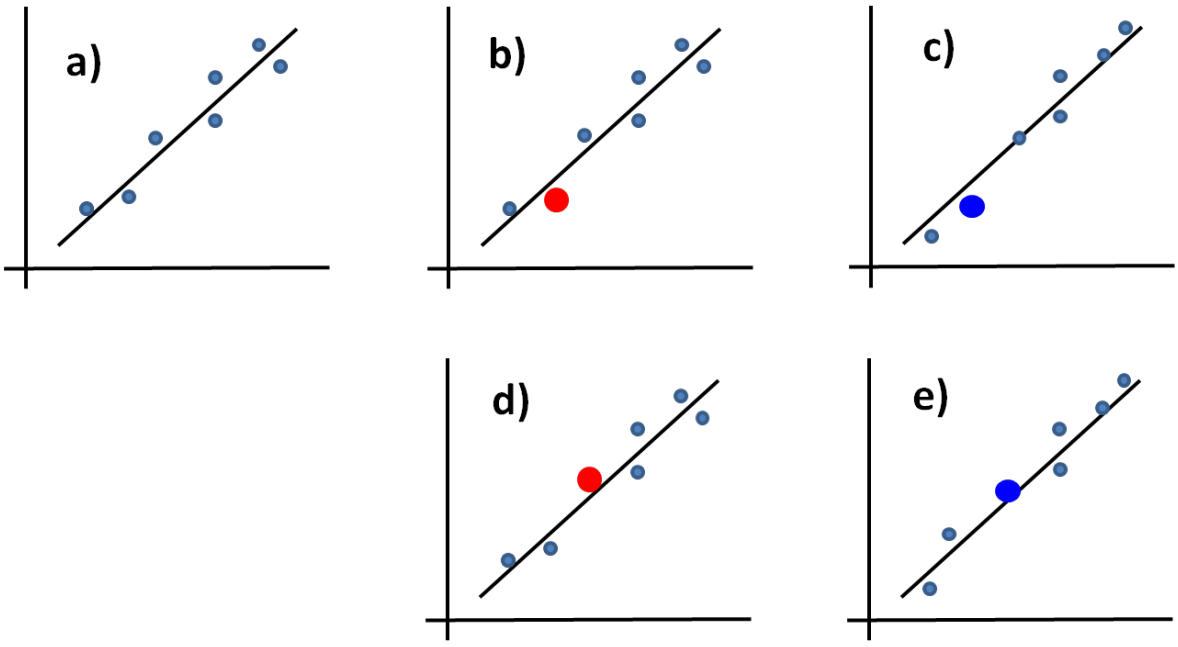


Рис. 4.8. Полная перекрестная проверка.

Следует отметить, что перекрестную проверку, строго говоря, нельзя использовать для оценки прогнозирующей способности модели, поскольку процедура построения модели и ее проверки осуществляется на одних и тех же образцах. Весьма часто величины RMSECV получаются ниже, чем RMSEP, полученные на независимом тестовом наборе, и поэтому являются слишком оптимистичной оценкой точности. Между тем, для оптимизации модели и подбора каких-либо параметров метод перекрестной проверки вполне подходит.

Когда доступных образцов очень мало, при построении многомерных регрессионных моделей резко возрастает риск случайных корреляций. Чтобы проверить модель на наличие случайных корреляций, используют перестановочный (пермутационный) тест. Схема проведения этого теста изображена на Рис. 4.9.

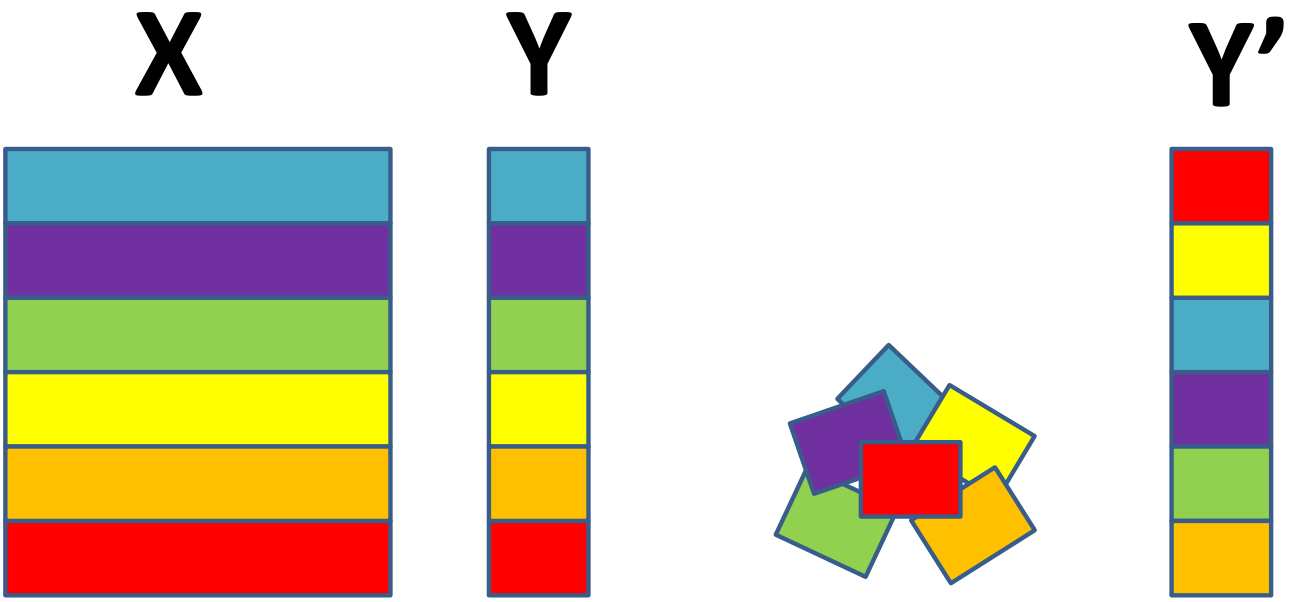


Рис. 4.9. Перестановочный тест.

Идея заключается в «перемешивании» случайным образом вектора **Y** для градуировочных образцов и построении моделей, для которых в качестве **X** берутся исходные данные, а в качестве Y вектора, в которых порядок следования концентраций для образцов изменен (**Y**’). Если модели, построенные на перемешанных **Y**, будут иметь метрики качества (R^2, *RMSE*) не хуже, чем у исходной модели, то это будет являться свидетельством того, что исходная модель была построена на случайных корреляциях и не имеет статистической значимости. На практике «перемешивание» проводят несколько десятков раз и отображают метрики полученных моделей в графическом виде – например, в виде гистограммы «номер пермутации (варианта перемешивания) – величина RMSE». В качестве нулевой пермутации отображают исходную модель.

Давайте рассмотрим два популярных метода многомерной регрессии.

4.5 Регрессия по главным компонентам

Основным ограничением классической МЛР является ограничение по количеству переменных – оно должно быть не больше числа градуировочных образцов. В противном случае решение по МНК для нахождения регрессионных коэффициентов будет не уникальным, что не позволит использовать модель для надежного прогнозирования новых образцов. Как вы помните, существует мощный метод сжатия данных без существенной потери информации – метод главных компонент (Глава 2). Основная идея метода *регрессии по главным компонентам* (англ. *pricncipal component regression*, *PCR*) заключается в том, что перед нахождением регрессионных коэффициентов по МЛР мы предварительно сжимаем данные с помощью МГК и используем вместо исходных переменных их линейные комбинации – счета МГК для нескольких ГК.

Схематично эта идея представлена на Рис. 4.10. Пусть у нас есть результаты спектральных измерений на 1500 длинах волн в 25 градуировочных образцах, записанные в матрицу **X**. Очевидно, что классическую модель МЛР в таком случае построить не удастся, поскольку число переменных гораздо больше числа образцов. Для сжатия данных воспользуемся МГК и проведем разложение матрицы **X** на произведение матрицы счетов и матрицы нагрузок, например, для трех первых ГК. Матрица счетов **T** будет иметь размерность 25 x 3, транспонированная матрица нагрузок $\mathbf{P}^{\mathrm{T}}$ - 3 x 1500. Поскольку счета являются линейной комбинацией исходных переменных, то они по-прежнему несут информацию о концентрационной зависимости в данных и пригодны для построения градуировки. С учетом размерности матрицы **T** расчет модели МЛР теперь не представляет проблемы. Проведем расчет регрессионных коэффициентов B по методу МНК, используя матрицу счетов **T**.

Количество регрессионных коэффициентов будет равно трем – по числу главных компонент, взятых в разложение. Для прогнозирования свойств новых образцов их необходимо спроецировать в то же самое пространство ГК, которое использовалось для расчета регрессионных коэффициентов. Проецирование осуществляется с помощью матрицы нагрузок и подробно описано в Главе 2, посвященной МГК. Далее, счета, полученные для новых образцов, умножаются на вектор регрессионных коэффициентов и получаются оценки искомой величины **Y**.

Число ГК является важным параметром модели и от его выбора зависит точность прогнозирования. Если для построения модели использовать недостаточное количество ГК, то при этом можно не учесть всю дисперсию в данных, связанную с изменением концентрации определяемого вещества. В этом случае модель будет недообученной. Если использовать избыточно большое число ГК, то, помимо дисперсии, связанной с изменением концентрации, модель начнет принимать в расчет в том числе и случайный шум в данных, что ухудшит качество прогнозирования. Для выбора оптимального числа ГК используют методы проверки моделей, описанные выше. Образцы разбиваются на градуировочный и проверочный наборы. С помощью градуировочных наборов рассчитывают регрессионные коэффициенты для разного

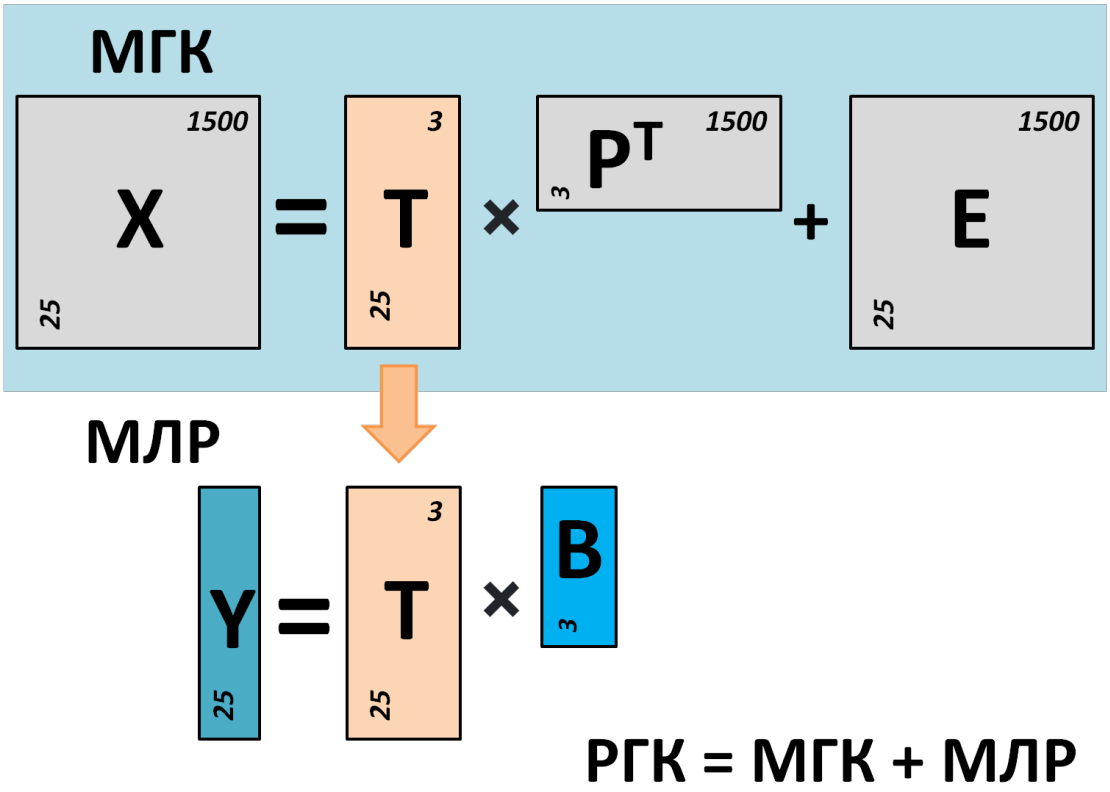


Рис. 4.10. Регрессия по главным компонентам.

числа ГК и вычисляют значения RMSEC. Проверочный набор используют для расчета RMSEP при тех же выбранных числах ГК. При этом получаются зависимости RMSEC и RMSEP от числа ГК (Рис. 4.11). Оптимальное значение должно соответствовать двум критериям: минимум RMSEP и близость значений RMSEP и RMSEC.

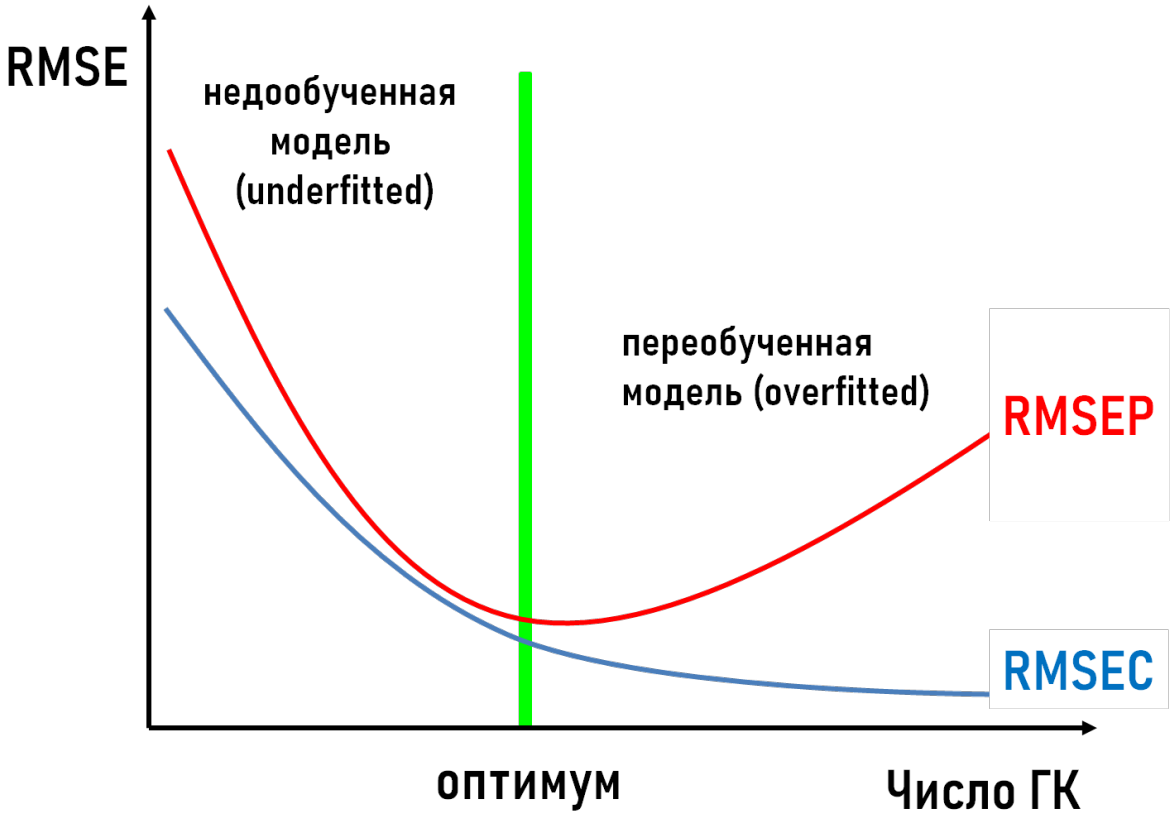


Рис. 4.11. Подбор оптимального количества ГК в регрессионных моделях.

Часто при недостаточно большом количестве образцов используют перекрестную проверку и вместо RMSEP используют значения RMSECV.

Безусловным преимуществом РГК является возможность построения моделей на гораздо большем числе переменных по сравнению с МЛР, что позволяет более полно и информативно использовать данные. Основной недостаток метода регрессии по главным компонентам заключается в том, что при МГК разложении учитываются все возможные источники дисперсии, которые необязательно могут быть связаны с изменением концентрации в градуировочных образцах. Если посторонние источники дисперсии (например, шумы) значительны, то при расчете регрессионных коэффициентов в модель попадет большое количество нерелевантной информации, что снизит точность прогнозирования.

Следующий алгоритм многомерного регрессионного моделирования, который мы рассмотрим, в значительной мере лишен этого недостатка, поскольку учитывает структуры данных и в **X**, и в **Y**.

4.6 Метод проекций на латентные структуры (ПЛС)

Метод *проекций на латентные структуры* (англ. *projections on latent structures*, или *partial least squares*, *PLS*) является одним из наиболее популярных методов многомерного регрессионного моделирования в хемометрике. Идея этого метода схожа с идеей РГК - в этом методе так же используется снижение размерности исходных данных для построения регрессионного уравнения, однако, в отличие от МГК, в методе проекций на латентные структуры ищется не просто наибольшая дисперсия, но только та наибольшая дисперсия, которая скоррелирована с изменением данных в **Y**.

Проиллюстрируем это на примере, приведенном на рис. 4.12. Пусть у нас есть три образца с условными концентрациями определяемого вещества 5, 10 и 15 (на рис. 4.12 они покрашены красным, синим и зеленым цветами соответственно). Для каждого образца измерены спектры, содержащие несколько линий, часть из которых не относится к определяемому соединению. В ходе моделирования алгоритм ПЛС рассчитает наибольшие регрессионные коэффициенты для тех переменных, значения которых скоррелированны с изменением значений концентрации в **Y**. В нашем примере это будут переменные в окрашенной серым области на рис. 4.12, где значения интенсивности меняются симбатно со значениями концентрации. Вклад других областей спектра в модель будет существенно меньше, поскольку там интенсивности меняются «вразнобой» с вектором концентраций.

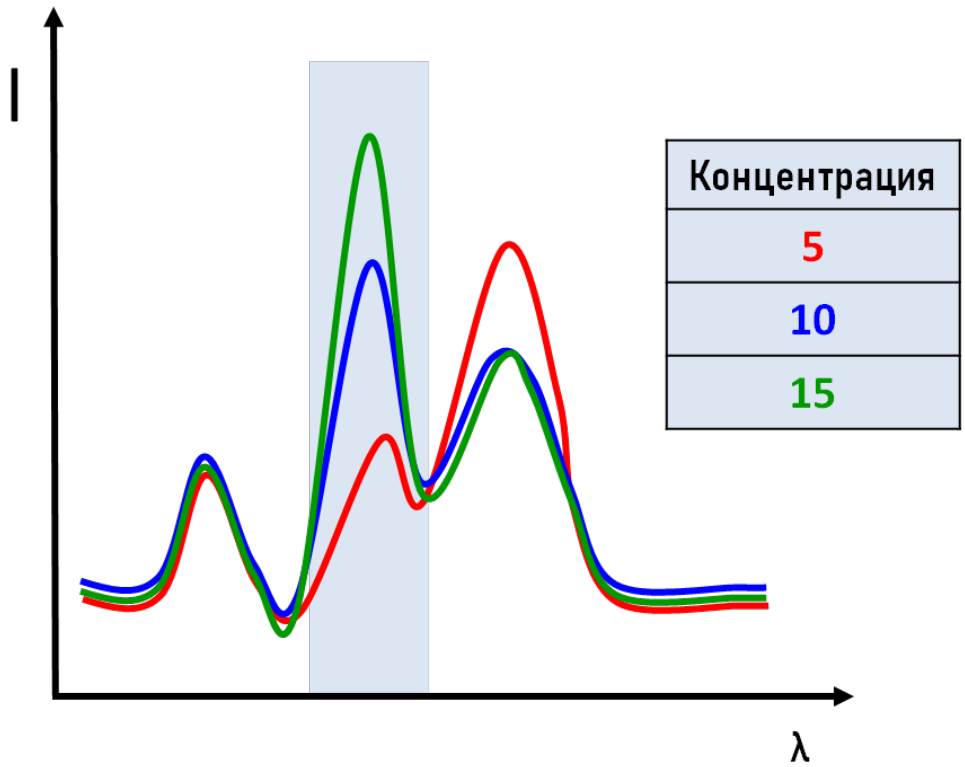


Рис. 4.12. Иллюстрация основной идеи ПЛС.

С математической точки зрения построение ПЛС модели происходит следующим образом. На первом этапе проводится разложение матриц X и Y на соответствующие счета и нагрузки согласно уравнениям:

$$\mathbf{X} = \mathbf{T}\mathbf{P}^{\mathrm{T}} + \mathbf{E} \tag{4.10}$$

$$\mathbf{Y} = \mathbf{U}\mathbf{Q}^{\mathrm{T}} + \mathbf{F} \tag{4.11}$$

В общем случае **Y** представляет собой матрицу, содержащую в столбцах концентрации различных интересующих соединений, поскольку алгоритм ПЛС позволяет проводить градуировку одновременно по нескольким веществам. На следующем этапе с условием максимизации ковариации между счетами для **X** и **Y** (**T** и **U** соответственно) рассчитывается новая матрица взвешенных нагрузок W, которая затем используется для расчета регрессионных коэффициентов:

$$\mathbf{W} = \max(\mathrm{cov}(\mathbf{T}, \mathbf{U})) \tag{4.12}$$

$$\mathbf{B} = \mathbf{W}(\mathbf{P}^{\mathrm{T}}\mathbf{W})^{-1}\mathbf{Q}^{\mathrm{T}} \tag{4.13}$$

где $\mathbf{P}^{\mathrm{T}}$ и **Q**T – транспонированные нагрузки для **X** и **Y** соответственно.

Для понимания работы ПЛС давайте еще раз вспомним, как проводится прогноз в РГК модели:

$$\mathbf{Y}^{\mathrm{pred}} = \mathbf{T}\mathbf{B}^{\mathrm{T}} \tag{4.14}$$

где **T** — это матрица счетов, полученная с помощью МГК разложения. Например, если известна матрица нагрузок, **P**, то матрицу **T** можно получить, проецируя исходные данные из **X** на вектора нагрузок:

$$\mathbf{T} = \mathbf{X}\mathbf{P} \tag{4.15}$$

Так вот, ПЛС использует точно такой же подход за одним исключением — счета в ПЛС вычисляются с помощью проекции на другой набор векторов, который представлен в виде матрицы **W**:

$$\mathbf{T} = \mathbf{X}\mathbf{W} \tag{4.16}$$

которая в свою очередь определяется по уравнению Equation 4.12.

Другими словами, чем сильнее переменная из **X** скоррелирована с **Y**, тем больший вес эта переменная будет иметь в матрице **W**. Тут надо заметить, что выражение Equation 4.16 немного упрощено для облегчения понимания метода, но, по сути, оно верно.

На практике ПЛС разложение проводят с помощью алгоритмов *NIPALS* (*nonlinear iterative partial least squares*) или *SIMPLS*. В случае, когда **Y** представляет собой вектор и модель строится для определения содержания одного компонента (так называемый метод *ПЛС1*), результаты вычисления регрессионных коэффициентов по этим двум алгоритмам практически не отличаются. Если **Y** – это матрица, и одна модель строится для определения сразу нескольких веществ (разновидность *ПЛС2*), то алгоритм *SIMPLS* более предпочтителен, поскольку он лучше максимизирует ковариацию между **X** и **Y** в случае многомерного **Y**.

На практике, даже в случае многомерного **Y** *ПЛС2* применяется крайне редко, обычно просто строят несколько *ПЛС1* моделей для каждого прогнозируемого вещества по отдельности.

Также, как и в РГК, параметром ПЛС модели является число компонент, участвующих в разложении **X** и **Y**. В случае ПЛС, однако, это число обычно называется числом *скрытых переменных* (англ. *latent variables*), поскольку здесь этот термин имеет несколько другой смысл. Оптимизация числа скрытых переменных осуществляется на основе графика типа Рис. 4.11, полученного с помощью тестового набора, либо перекрестной проверки.

Важным инструментом изучения ПЛС моделей являются регрессионные коэффициенты. Рассмотрим это на примере спектров рентгеновской флуоресценции (РФА), измеренных для образцов сложных многокомпонентных смесей, содержащих европий – определяемый элемент в этой задаче (Рис. 4.13).

В верхней части Рис. 4.13 изображены спектры нескольких образцов смесей. Ниже показаны РФА линии европия. Видно, что сигналы европия в измеренных спектрах не имеют четкой структуры и сильно перекрываются с линиями других компонентов. Кроме того, из-за малых концентраций спектры имеют низкое соотношение сигнал/шум. Одномерная регрессия по характеристической линии европия, традиционно применяемая в РФА, дает в этой задаче неудовлетворительные результаты. Применение ПЛС регрессии позволило обойти эти проблемы и получить модели для количественного определения европия.

В нижней части Рис. 4.13 синим цветом приведены регрессионные коэффициенты такой ПЛС модели. Видно, что их форма близка к форме РФА линий европия. Алгоритм ПЛС смог вычленить в сложной структуре спектров смесей вклад определяемого элемента и «уловил» важность конкретных областей спектра для прогнозирования концентрации европия, что позволяет судить об адекватности модели. Если бы форма регрессионных коэффициентов существенно отличалась от формы линий спектра чистого европия, это являлось бы свидетельством того, что модель держится на случайных корреляциях. Кроме того регрессионные коэффициенты позволяют проводить оптимизацию модели с точки зрения выбора значимых переменных. Очевидно, что включать в модель переменные с регрессионными коэффициентами, близкими к нулю, не имеет большого смысла, поскольку они несут только нерелевантный шум. Перестроение модели с использованием только значимых областей спектра, регрессионные коэффициенты для которых существенно отличаются от нуля, как правило, позволяет улучшить качество моделирования.

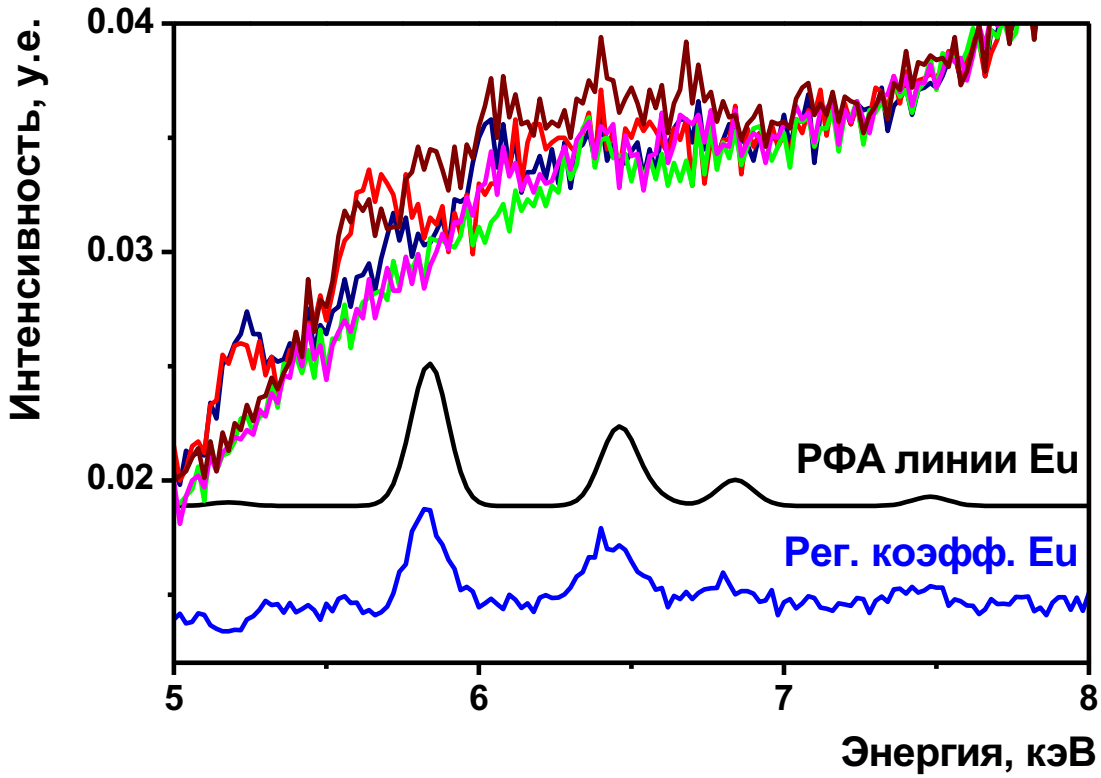


Рис. 4.13. РФА спектры сложных смесей, содержащих европий, РФА линии европия и регрессионные коэффициенты ПЛС модели для определения европия.

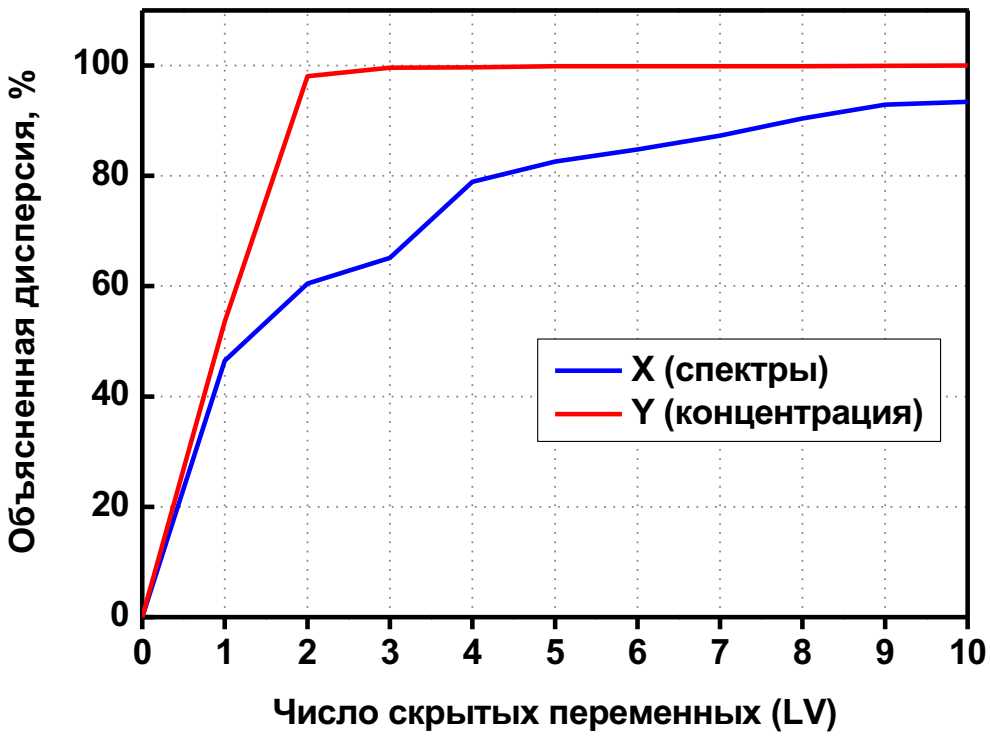


Рис. 4.14. График объясненной дисперсии ПЛС модели.

Для понимания структуры данных интересно посмотреть на график объясненной дисперсии. На Рис. 4.14 приведен такой график для модели из предыдущего примера по определению европия. Видно, что объясненная дисперсия в матрице спектров **X** начинает приближаться к 90% только на девятой скрытой переменной, в то время как объясненная дисперсия для Y достигает практически 100% уже на второй скрытой переменной. Это означает, что значительная часть дисперсии в матрице **X** не скоррелирована с изменением концентрации европия (вектор **Y**). Тем не менее, даже с такими данными алгоритм ПЛС позволяет успешно строить модели для количественного прогнозирования целевых параметров. Этим обусловлена широкая популярность метода в решении задач количественного анализа.

Использование ПЛС в R

Для построения ПЛС регрессии в R необходимо имплементировать алгоритмы NIPALS, или SIMPLS. Однако, так как это довольно популярный метод, имеется большое число пакетов, где эти и многие другие алгоритмы уже реализованы, так что мы воспользуемся готовой реализацией. Вы можете попробовать самостоятельно имплементировать алгоритмы в качестве тренировки.

В примерах ниже мы будем использовать уже известный вам пакет *mdatools*. Однако, вы можете попробовать и другие пакеты, например *pls* или *Chemometrics*.

Попробуем решить задачу, уже рассмотренную нами выше в разделе про МЛР: построить регресионную модель, которая будет прогнозировать концентрацию компонента *C3* из набора *Simdata* по спектральным данным. Однако, в отличие от МЛР, в данном случае мы можем использовать все спектры целиком, так как число переменных, как и внутренние корреляции, не являются проблемой для ПЛС.

Начнем с построения модели без валидации. Код ниже показывает, как подготовить данные, построить модель и показать ее производительность.

```
library(mdatools)
data(simdata)
Xc <- simdata$spectra.c
yc <- simdata$conc.c[, 3]

m <- pls(Xc, yc, 10)
summary(m)
```

```
PLS model (class pls) summary
------------------------------
Info:
Number of selected components: 10
```

```
Cross-validation: none

    X cumexpvar Y cumexpvar    R2  RMSE Slope Bias   RPD
Cal       99.98      99.914 0.999 0.001 0.999    0 34.24
```

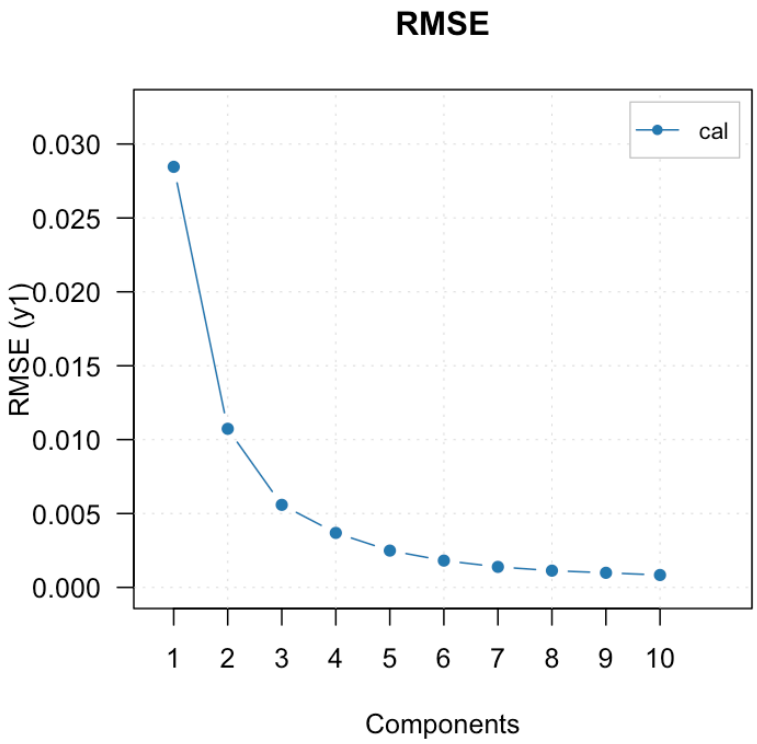
Число 10, указанное выше при вызове функции `pls()` — это максимальное число компонент, которое нужно использовать. Как можно видеть из информации о модели, все 10 компонент были использованы и статистика показана именно для этого числа компонент в модели.

Таблица показывает следующие характеристики модели:

- `X cumexpvar` — процент дисперсии $\mathbf{X}$ которая объясняет эта модель
- `Y cumexpvar` — процент дисперсии $\mathbf{y}$ которая объясняет эта модель
- `R2` — коэфициент детерминации
- `RMSE` — среднеквадратичная ошибка
- `Slope` — угол наклона прямой для предсказанных и измеренных значений $\mathbf{y}$
- `Bias` — среднее значение ошибки предсказания
- `RPD` — *residual prediction deviation*, отношение стандратного отклонения откликов к стандартному отклонению ошибок

Если посмотреть на RMSE и R2 значения, то можно заключить, что модель почти идельно прогнозирует значения концентраций. Однако, это не совсем так. Давайте посмотрим на кривую зависимости RMSE от числа компонент в модели:

```
plotRMSE(m)
```



Мы видим классический паттерн, показаный также на рисунке 4.12 — чем больше число компонент в ПЛС модели, тем лучше предсказания, сделанные для калибровочного набора. Для того, чтобы найти оптимальное число компонент и оценить реальную производительность модели нужна валидация.

Так как в данном случае в данных имеетеся тестовый набор, воспользуемся им:

```
Xt <- simdata$spectra.t
yt <- simdata$conc.t[, 3]

m <- pls(Xc, yc, 10, x.test = Xt, y.test = yt)
summary(m)
```

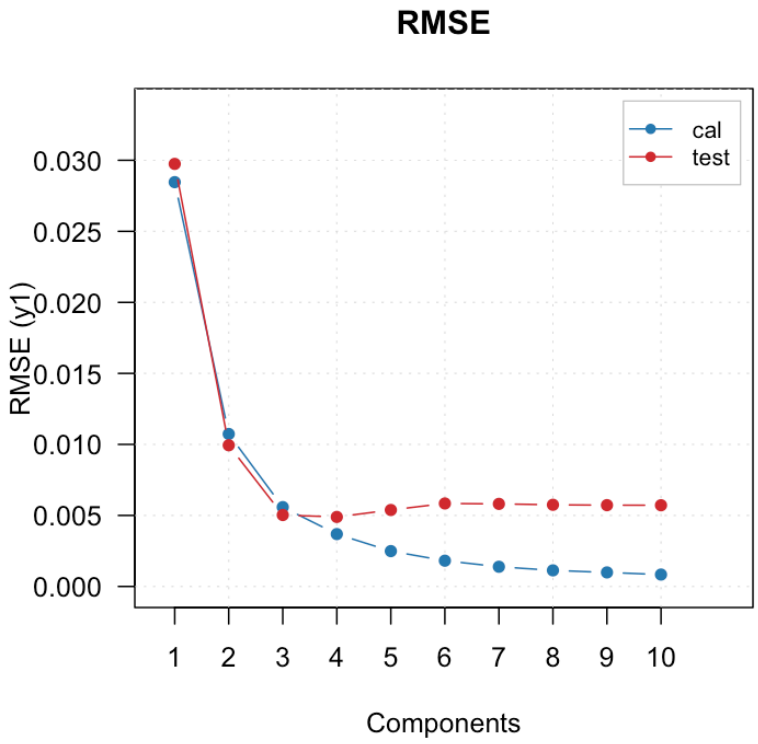
```
PLS model (class pls) summary
------------------------------
Info:
Number of selected components: 4
Cross-validation: none
```

```
     X cumexpvar Y cumexpvar    R2  RMSE Slope  Bias  RPD
Cal       99.977      98.335 0.983 0.004 0.983 0e+00 7.79
Test      99.983      97.478 0.972 0.005 0.986 1e-04 6.03
```

Как можно заметить, в данном случае таблица состоит из двух строк. Первая строка — это характеристики модели, расчитанные для градуировочного набора (так же как и в предыдущем примере). Вторая строка — это характеристики, рассчитанные для предсказаний, сделанных для тестового набора. Кроме того, можно отметить, что число выбранных компонент в этом случае уже не 10, а 4. Почему четыре?

Посмотрим еще раз на график RMSE:

```
plotRMSE(m)
```



Очевидно, что ошибка для тестового набора выглядит примерно так, как это показано на рисунке 4.12 — вначале идет вниз с ростом числа компонент в модели, а, начиная с какого-то числа, либо идет немного вверх, либо не меняется. В нашем случае после четырех компонент ошибка немного растет, поэтому модель и определила четыре, как оптимальное число.

Однако, на графике видно, что разница между 3 и 4 компонентами незначительная. Поэтому нет нужды использовать более сложную модель в данном случае. Это можно получить в числовом виде, если применить функцию `summary()` не ко всей модели, а только к результатам тестового набора (все результаты находятся "внутри" объекта `m`):

```
summary(m$res$test)
```

```
PLS regression results (class plsres) summary
Info: test set validation results
Number of selected components: 4
        X expvar X cumexpvar Y expvar Y cumexpvar     R2  RMSE  Slope    Bias
Comp 1    99.587      99.587    6.845       6.845 -0.038 0.030 -0.011  0.0018
Comp 2     0.248      99.836   82.752      89.598  0.884 0.010  0.927 -0.0017
Comp 3     0.147      99.983    7.738      97.336  0.970 0.005  0.974 -0.0003
Comp 4     0.000      99.983    0.142      97.478  0.972 0.005  0.986  0.0001
Comp 5     0.000      99.983   -0.531      96.947  0.966 0.005  0.994  0.0003
Comp 6     0.000      99.983   -0.542      96.405  0.960 0.006  1.001  0.0002
Comp 7     0.000      99.983    0.036      96.441  0.960 0.006  1.006  0.0001
Comp 8     0.000      99.983    0.078      96.519  0.961 0.006  1.010  0.0001
Comp 9     0.000      99.983    0.031      96.550  0.962 0.006  1.016  0.0000
Comp 10    0.000      99.983    0.008      96.558  0.962 0.006  1.015 -0.0002
         RPD
Comp 1  0.99
Comp 2  3.01
Comp 3  5.88
Comp 4  6.03
Comp 5  5.49
Comp 6  5.05
Comp 7  5.07
Comp 8  5.13
Comp 9  5.15
Comp 10 5.16
```

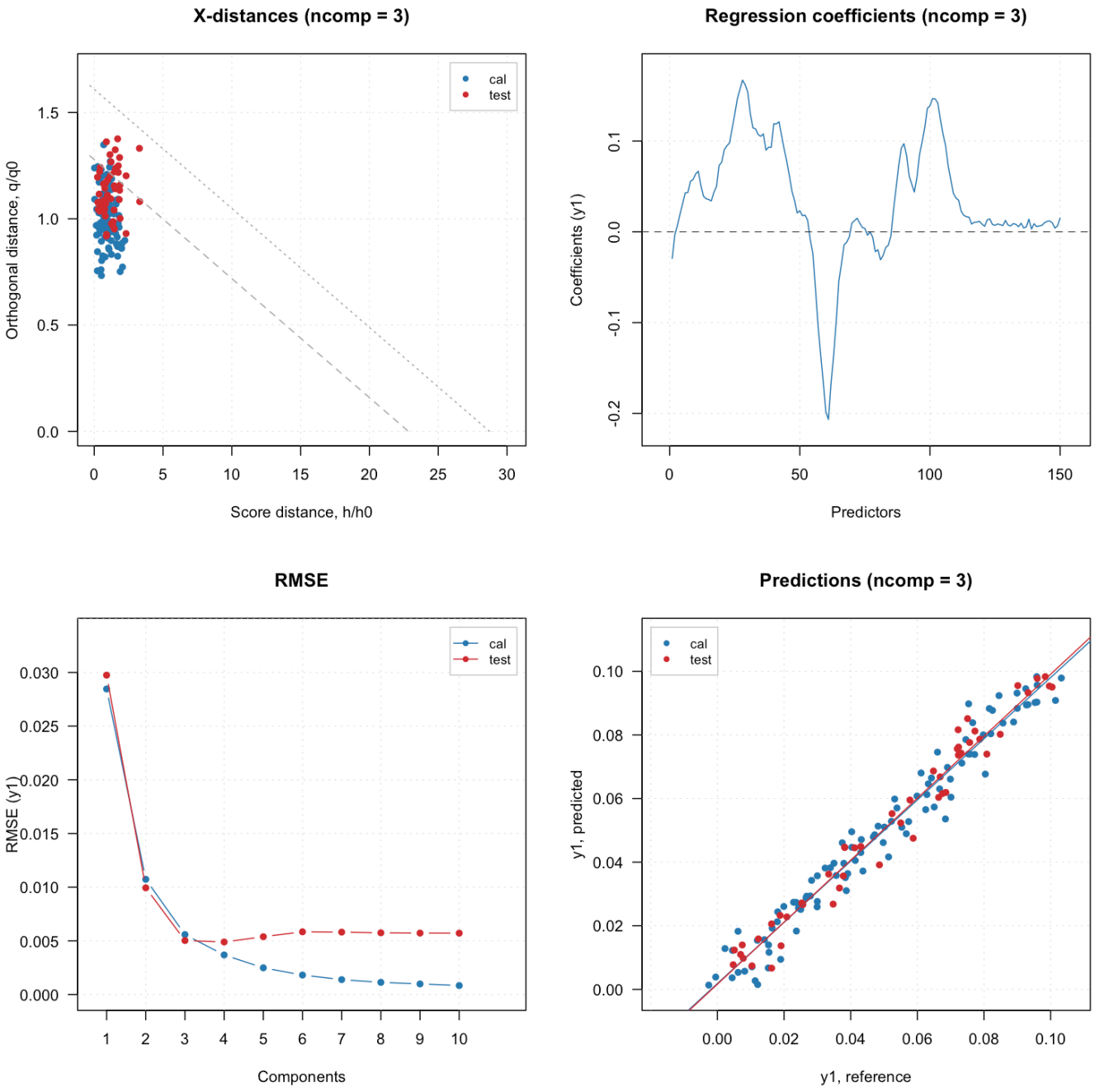
Так, для модели с 3 компонентами коэффициент детерминации составляет 0.970, а для модели с 4 компонентами — 0.972. Это довольно небольшая разница, чтобы оправдать наличие лишней компоненты, поэтому в этом случае нужно "сказать" модели, чтобы она использовала число 3 в качестве оптимального:

```
m <- selectCompNum(m, 3)
summary(m)
```

```
PLS model (class pls) summary
------------------------------
Info:
Number of selected components: 3
Cross-validation: none

     X cumexpvar Y cumexpvar    R2  RMSE Slope   Bias  RPD
Cal       99.977      96.180 0.962 0.006 0.962  0e+00 5.14
Test      99.983      97.336 0.970 0.005 0.974 -3e-04 5.88
```

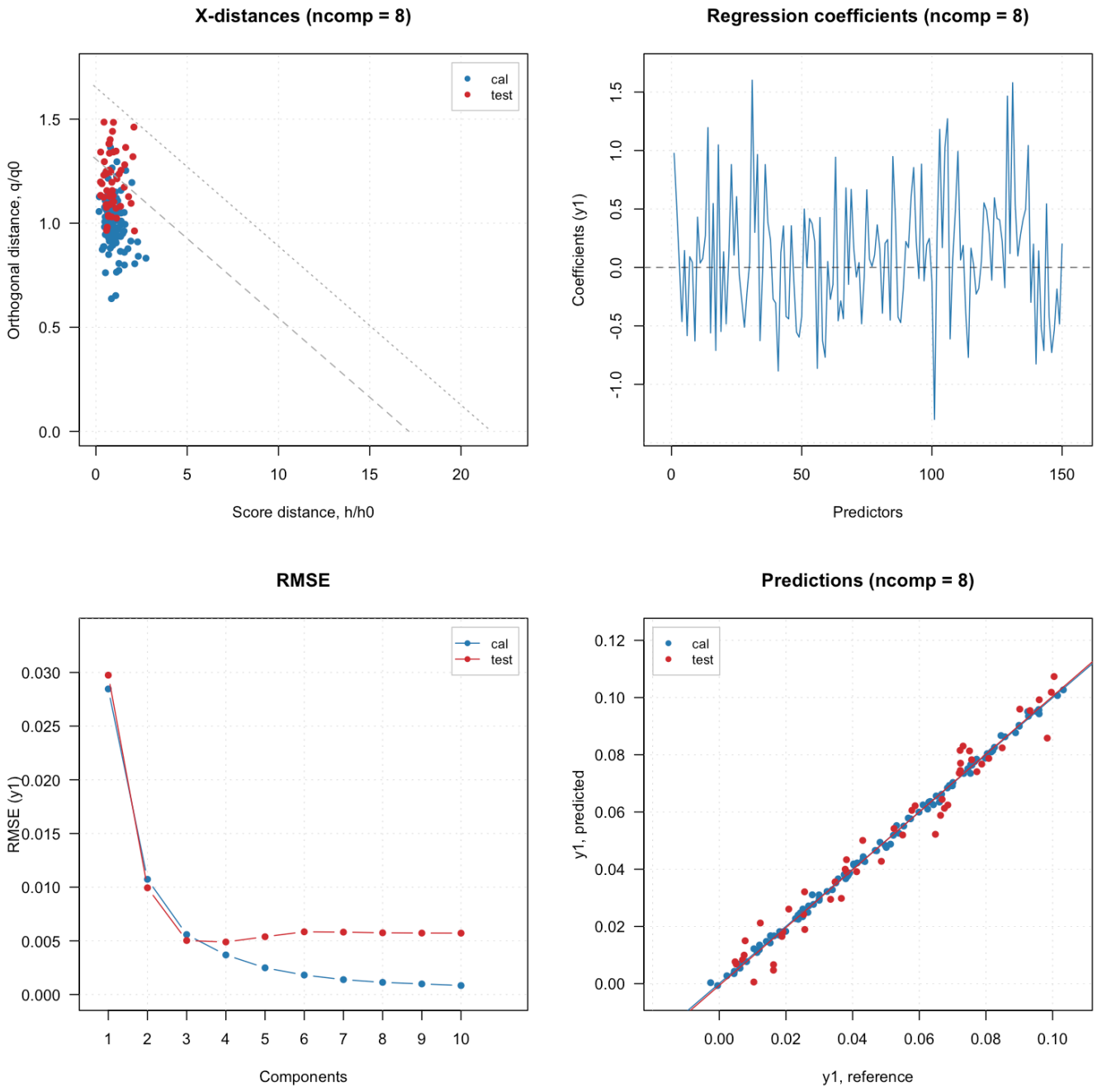
Далее вы можете исследовать модель на наличие выбросов, посмотреть на регресионные коэффициенты, на счета и нагрузки, объясненную дисперсию, и так далее. Подробный список всех возможных графиков и действий можно найти в справке для функции `pls()`. Самым простым действием будет построение четырех основных графиков: график расстояний, график реггресионных коэффициентов, график ошибок (уже показанный нами ранее отдельно) и график “введено-посчитано”. Все четыре графика вместе можно показать простой командой:

```
plot(m)
```



По умолчанию графики показываются для оптимального числа компонент. Вы можете это поменять если указать желаемое число в качестве дополнительного аргумента:

```
plot(m, ncomp = 8)
```



На этом графике хорошо видно, что модель переопределена — прогнозы для калибровочного набора (синие точки) почти идеальные, в то время как тестовый набор предсказывается гораздо хуже. Кроме того, регресионные коэффициенты выглядят довольно зашумленными.

В заключение покажем, как валидировать модель с помощью кросс-валидации в *mdatools*. Для этого нужно вместо тестового набора задать значение для аргумента `cv`, которое в общем виде выглядит как: `list("type"`

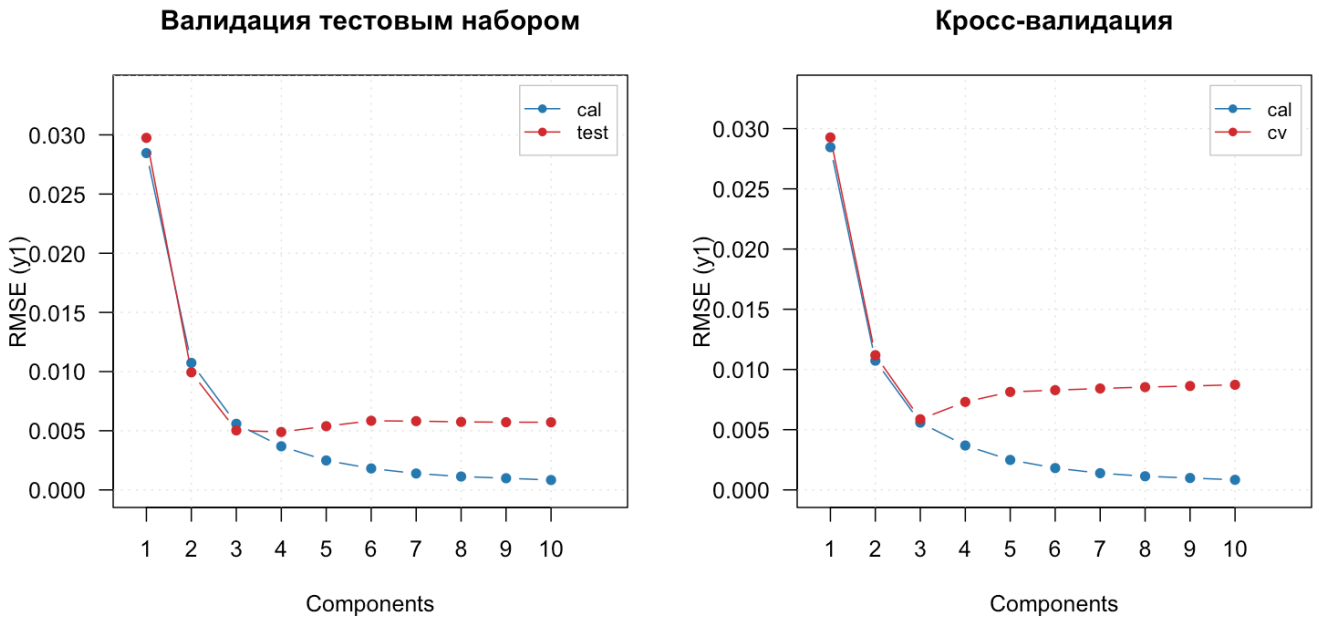
`, nseg, nrep)`. Здесь `"type"` это название способа разбиения, поддерживаются следующие типы: `"full"` — полная кросс-валидация, `"rand"` — кросс-валидация со случайным разбиением на сегменты и `"ven"` — venetian blinds, кросс-валидация с систематическим разбиением.

В случае полной кросс-валидации дополнительных параметров указывать не нужно, т.е. код будет выглядеть как `cv = list("full")`. Для других двух типов нужно указать число сегментов, например `cv = list("rand" , 10)` означает кросс-валидацию со случайным разбиением строк на 10 сегментов.

В случае случайного разбиения результаты кросс-валидации будут отличаться, если выполнить построение модели несколько раз подряд. Для того, чтобы сгладить этот эффект используют кросс-валидацию с повторением: процесс разбиения и подсчета характеристик будет повторен несколько раз и значения характеристик усреднены. Например `cv = list("rand" , 10, 10)` означает кросс-валидацию со случайный разбиением на 10 сегментов и с 10 повторениями. Нужно иметь в виду, что использование повторений потребует дополнительного времени на рассчеты.

Посмтрим ПЛС модель с использованием систематического разбиения на 5 сегментов и сравним графики RMSE для кросс-валидации и для тестовой валидации:

```
mcv <- pls(Xc, yc, 10, cv = list("ven", 5))

par(mfrow = c(1, 2))
plotRMSE(m, main = "Валидация тестовым набором")
plotRMSE(mcv, main = "Кросс-валидация")
```



Графики ошибок довольно похожи, но в случае с кросс-валидацией выбор трех компонент более очевиден, как и характерный паттерн для переопределенной модели.

5 Методы многомерной классификации

5.1 Основные понятия

Задачей *классификации* является разделение образцов на группы (*классы*) в соответствии с их свойствами, или характеристиками. При этом свойства образцов могут быть практически любыми. Так, в хемометрике под свойствами можно понимать, например, ИК-спектры образцов, их хроматограммы, различные физико-химические свойства (температура плавления, энтальпия парообразования, плотность, и т.п.).

Методы классификации используют для определения принадлежности нового образца к одному, или нескольким известным классам, а также для установления общности свойств образцов в различных классах. Такие задачи очень широко распространены в аналитической химии, например, при проверке качества (не выходят ли характеристики образца за границы, предписанные стандартом?), аутентичности (действительно ли образец происходит из географической области, задекларированной продавцом?), при решении задач медицинской диагностики (пациент относится к классу «норма», или «патология» по совокупности результатов клинических исследований?).

С точки зрения задачи разбиения образов на группы, можно выделить два подхода: *кластеризация* и классификация. Принципиальное различие между этими подходами заключается в том, что в случае кластеризации для разбиения набора образцов на группы используются только свойства образцов без априорной информации о тех классах, к которым они действительно принадлежат. Такие методы иногда называют методами *классификации без учителя* (англ. *unsupervised classification*). В случае классификации задача стоит несколько иначе: необходимо на основании свойств образцов с известной классовой принадлежностью (обучающий набор) вывести правило, согласно которому новые неизвестные образцы можно отнести к тому, или иному классу (*классификационное правило*). Эти методы иногда называют *классификацией с учителем* (англ. *supervised classification*).

5.2 Геометрическое представление данных

Для понимания основных принципов классификации удобно воспользоваться геометрическим представлением данных, которое мы подробно рассмотрели во второй главе. Пусть у нас есть набор образцов, описанных двумя переменными (x_1 и x_2), которые принадлежат двум различным классам. Эти переменные описывают какие-либо свойства образцов, например, концентрацию сульфата и хлорида в образцах минеральной воды, добытых в двух разных источниках. Каждый из этих образцов можно

представить точкой в двухмерном пространстве с координатами (x_1; x_2), как показано на рисунке 5.1. Такое представление позволяет легко визуализировать данные и сделать предварительные выводы о возможности создания классификационного правила для разделения образцов классов. В случае приведенного примера такое классификационное правило может быть основано на уравнении прямой, разделяющей точки, принадлежащие разным классам (серый пунктир на рис. 5.1). Новые образцы будут относиться к тому, или иному классу, в зависимости от того, с какой стороны от прямой они находятся.

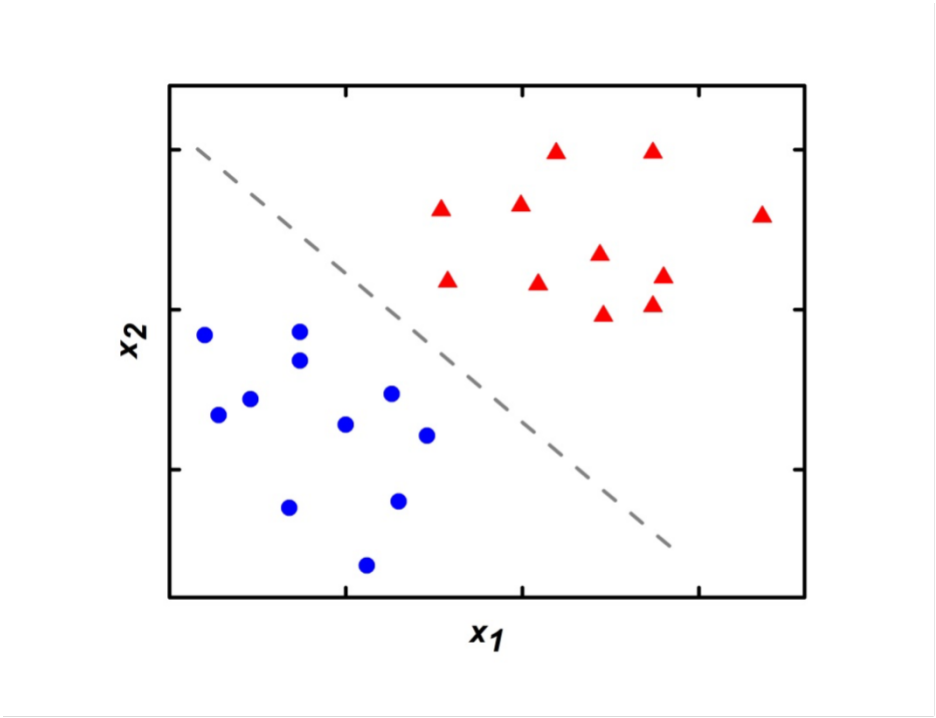


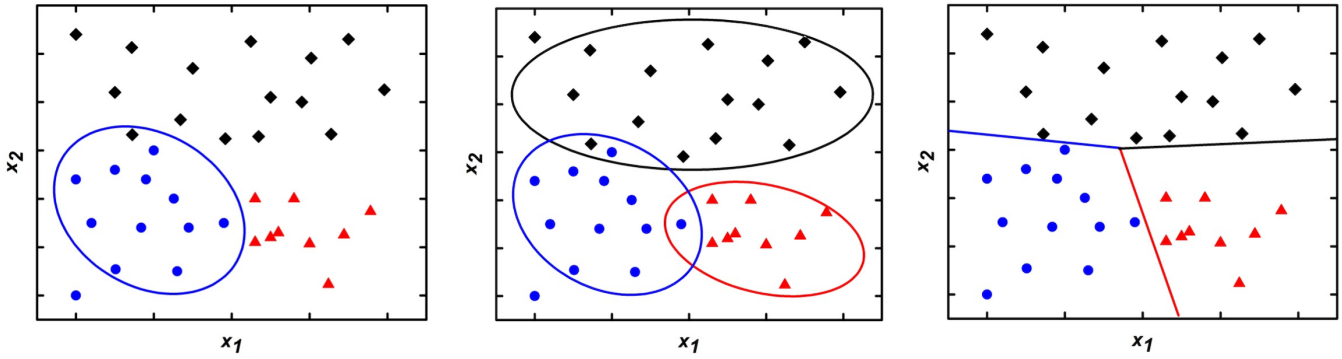
Рис. 5.1. Графическое представление образцов двух классов и разделяющей прямой.

В случае большого числа переменных, характеризующих образцы, подобная визуализация затруднена. В этом случае часто прибегают к предварительному сжатию данных с помощью метода главных компонент, либо применяют специальные многомерные методы, о которых мы поговорим ниже.

Различают два основных типа классификации: *одноклассовая* и *многоклассовая*. При одноклассовой классификации исследуемый образец может либо принадлежать, либо не принадлежать одному заранее описанному классу. Типичным примером одноклассовой классификации являются задачи аутентификации (да, этот образец вина был произведен из винограда, выращенного в таком-то конкретном регионе – нет, этот образец – что-то другое). Мерой схожести образцов между собой при таком представлении является, как правило, *расстояние* между образцами. Чем дальше образцы друг от друга, тем больше они различаются по свойствам x_1 и x_2. Во многих метода классификации расстояния используются для создания классификационного правила. Способ оценки расстояния будет сильно влиять на результаты классификации. Наиболее часто используются Евклидово расстояние и расстояние Махаланобиса (глава 2).

Для создания классификационного правила в этом случае нужно собрать репрезентативную выборку образцов одного конкретного класса. Все остальные образцы будут считаться не принадлежащими классу, безотносительно их конкретной классовой принадлежности. Если имеются репрезентативные образцы для нескольких классов, то можно построить одноклассовый классификатор для каждого из них и применять их вместе для классификации новых образцов (многоклассовая классификация). В этом случае образец может быть классифицирован как принадлежащий одному классу, нескольким классам, или ни одному классу вообще, как проиллюстрировано на рисунке 5.2.

Вид классификации, при котором каждый образец может относиться к одному и только одному классу и не может оказаться не принадлежащим к какой-то группе, называется *дискриминацией*. Как и ранее, эти различия в идеологии методов удобно проиллюстрировать графически.



а) одноклассовая классификация **б)** многоклассовая классификация **в)** дискриминация

Рис. 5.2. Классификация и дискриминация.

На рис. 5.2 представлены образцы трех различных классов, отмеченные разным цветом. Рис. 5.2а показывает пример одноклассового классификатора. Модель, задающая границы «синего» класса, обозначена синим эллипсом. Образцы, расположенные внутри эллипса при этом считаются образцами, принадлежащими классу, а образцы, находящиеся снаружи – не принадлежащими, без дальнейших уточнений относительно того, какому именно другому классу они принадлежат. Обратите внимание, что один из образцов «черного» класса при этом ошибочно включен в модель, а один из образцов «синего» класса ошибочно не включен в модель. Пример многоклассовой классификации приведен на рис. 5.2б. В этом случае задаются граничные условия для каждого класса – они представлены эллипсами соответствующих цветов. При этом отдельные образцы могут принадлежать сразу нескольким классам (образцы на пересечении границ «синего» и «черного» классов, «синего» и «красного» классов), могут принадлежать только одному классу (большинство образцов на рисунке), либо могут не принадлежать ни одному из классов (синий и черный образцы за границами эллипсов). Рис. 5.2в демонстрирует пример дискриминации. В этом случае границы классов заданы так, что все образцы принадлежат к какому-то конкретному классу и при этом только к одному.

5.3 Оценка качества классификации

Как видно из рис. 5.2, при классификации возможны два вида ошибок: отнесение объекта к «чужому» классу, или непринятие объекта «своим» классом. С точки зрения статистики каждый класс может быть формально описан, как статистический критерий с нулевой гипотезой: «объект принадлежит классу». В этом случае, если модель ошибочно отклоняет объект, принадлежащий классу, то это будет являться *ошибкой первого рода* (англ. *Type I error*), которая обычно обозначается как *FN* (*false negative*, ложно отрицательный образец). Например, на рис. 5.2а это синий образец, оказавшийся за границей «синего» класса. Количество таких ошибок характеризует способность классификационной модели «узнавать» образцы, принадлежащие описываемому классу, и связано с параметром *чувствительность* (англ. *sensitivity*):

$$Sensitivity = \frac{TP}{TP + FN} \tag{5.1}$$

где *TP* (*true positive*, истинно положительный) — количество образцов правильно отнесенных классификационной моделью к моделируемому классу. По сути, чувствительность показывает относительную долю образцов, принадлежащих классу, правильно распознанных моделью. Ошибочное отнесение к моделируемому классу образца, который на самом деле ему не принадлежит, является *ошибкой второго рода* (англ. *Type II error*) и обозначается как *FP* (*false positive*, ложно положительный). На рис. 5.2а это черный образец, оказавшийся внутри «синего» класса. Количество таких ошибок отражает способность модели отклонять «чужие» образцы и связано с параметром *селективности* (англ. *specificity*):

$$Specificity = \frac{TN}{TN + FP} \tag{5.2}$$

где *TN* (*true negative*, истинно отрицательный) — количество образцов правильно не отнесенных классификационной моделью к моделируемому классу. Другими словами, селективность — это относительная доля образцов, не принадлежащих классу, правильно отвергнутых моделью.

В зависимости от решаемой задачи ценность каждого из параметров для выбора оптимальной модели различается. Так, например, если вы используете классификационную модель для выявления какого-либо заболевания на основе совокупности различных анализов (т. е. модель распознаёт больных пациентов, как членов класса) и нулевая гипотеза заключается в отсутствии у пациента заболевания, то для вас важнее селективность (важнее не пропустить заболевшего, чем ошибочно признать здорового заболевшим). Если вы работаете судьей, и нулевая гипотеза заключается в том, что подсудимый не виновен (презумпция невиновности), то для вас из соображений гуманности и справедливости важнее чувствительность (важнее не осудить невиновного, чем отпустить виновного).

Следует отметить, что в дискриминации ошибки первого рода для одного класса автоматически становятся ошибками второго рода для другого класса и наоборот.

Общее качество классификационной модели часто оценивают параметром *точность* (англ. *accuracy*):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.3}$$

По сути, это доля правильно классифицированных образцов (как правильно распознанных, так и правильно отвергнутых) от общего числа образцов.

Все три описанные метрики принято выражать в процентах. Они используются для сравнения между собой различных алгоритмов классификации, а также для оптимизации параметров моделирования. Для наглядного представления результатов классификации часто используется так называемая *матрица ошибок* (англ. *confusion matrix*), приведенная на рис. 5.3.

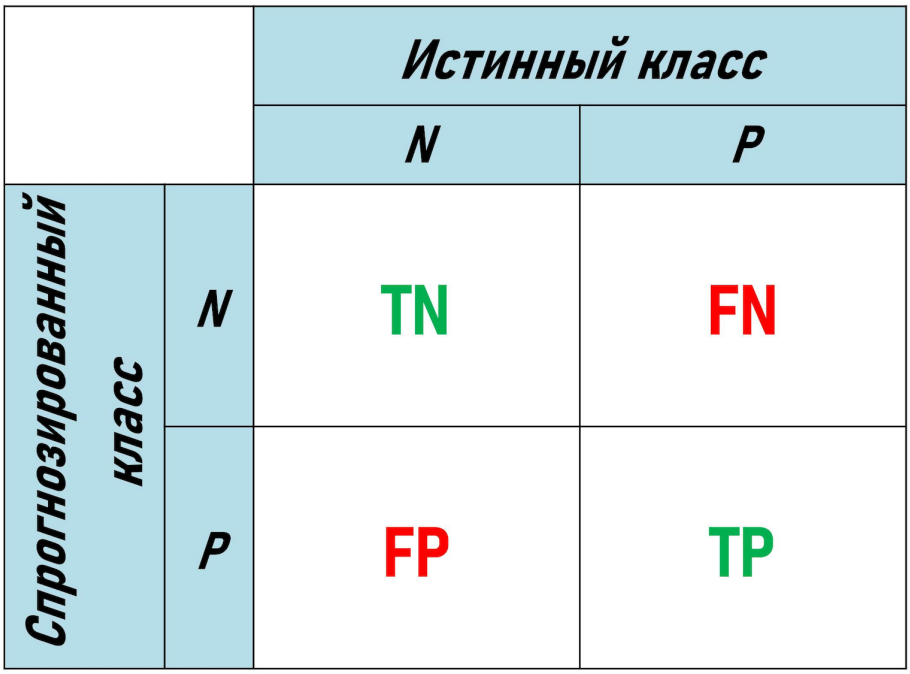


Рис. 5.3. Матрица ошибок.

В матрице ошибок образцы разделены на четыре группы (*TN*, *TP*, *FN*, *FP*) в зависимости от того, к какому классу они отнесены моделью. На основе матрицы ошибок удобно рассчитывать чувствительность, специфичность и точность классификационной модели.

Как и в других методах моделирования, для построения классификационной модели необходим обучающий набор образцов (англ. *training set*), а для оптимизации и проверки ее качества — проверочный набор (англ. *test set*). Принципы выбора образцов в эти наборы совпадают с изложенными выше требованиями к созданию обучающих и проверочных наборов для регрессионного моделирования, главный из которых заключается в том, что в обоих наборах образцы должны репрезентативно

представлять все возможные свойства образцов в классе. Матрицу ошибок можно рассчитывать как для обучающего, так и для проверочного набора, однако, для сравнения моделей используют метрики, рассчитанные для проверочных наборов.

Общая схема построения и применения классификационной модели аналогична случаю регрессионного моделирования:

1. разбиение образцов на обучающий и проверочные наборы;
2. построение классификационной модели;
3. валидация модели;
4. оптимизация параметров (в случае, если у модели есть параметры);
5. оценка качества итоговой модели;
6. применение для классификации новых образцов.

Далее мы рассмотрим несколько наиболее популярных методов кластеризации и классификации данных, которые могут работать как с данными небольшой размерности, так и с многомерными данными.

5.4 Кластеризация с помощью метода k-средних

Метод k-средних, пожалуй, является самым простым и в тоже время самым распространенным методом кластеризации данных. Как и большинство других подобных методов он основан на оценке близости между образцами и «пытается» объединить близкие объекты в кластеры. Существует множество модификаций этого метода, которые используют различные метрики близости. Здесь мы рассмотрим его классический вариант, когда в качестве такой метрики используется расстояние Евклида в пространстве переменных. Для начала напомним, что в задаче кластеризации у нас нет калибровочного набора для каждого класса или кластера, т.е. мы заранее не знаем к какому классу принадлежат образцы. Представьте себе двадцать незнакомых людей, которых нужно разделить на две группы, так, чтобы им было интересно общаться друг с другом. Заранее мы не знаем, как это сделать, но измеряя различные индивидуальные характеристики (какие книги эти люди читают, какие фильмы смотрят, на какие ссылки переходят) можно постепенно посчитать меру близости между индивидами и объединить их в группы, или кластеры соответствующим образом. Именно так работают алгоритмы кластеризации в современных социальных сетях, когда вы видите различные рекомендации.

Таблица. 5.1. Содержание оксида натрия в 20 образцах стекла и предварительная кластеризация образцов.

#	Na2O, w/w %	Расстояние до C1	Расстояние до C2	Кластер
1	10.7	1.6	2.9	1
2	10.4	1.9	3.2	1
3	9.3	3.0	4.3	1
4	10.4	1.9	3.2	1

#	Na2O, w/w %	Расстояние до C1	Расстояние до C2	Кластер
5	11.1	1.2	2.5	1
6	10.3	2.0	3.3	1
7	9.9	2.4	3.7	1
8	13.2	0.9	0.4	2
9	11.4	0.9	2.2	1
10	10.9	1.4	2.7	1
11	12.5	0.2	1.1	1
12	13.0	0.7	0.6	2
13	11.9	0.4	1.7	1
14	14.2	1.9	0.6	2
15	14.6	2.3	1.0	2
16	13.5	1.2	0.1	2
17	11.7	0.6	1.9	1
18	11.5	0.8	2.1	1
19	14.3	2.0	0.7	2
20	13.3	1.0	0.3	2

В науке кластеризация используется также для разбиения на группы (или, скорее, объединения в кластеры) образцов, знаний о которых недостаточно, или они являются неполными. Так, например, в 2021 году, отслеживались мутации вируса COVID-19 на основе его генетического кода. Как только появлялось достаточное количество проб с вирусами, ДНК которых были похожи и, одновременно, отличались от ДНК других ранее изученных образцов, эти пробы объединяли в отдельный кластер и определяли как новый вариант.

Для иллюстрации принципов работы метода k-средних будем использовать самый простой, одномерный пример, данные для которого показаны в таблице 5.1. Эти данные (второй столбец в таблице) представляют собой содержание оксида натрия в 20 кусочках стекла, найденных на месте автомобильной аварии (в процентах от массы образца). К примеру, мы бы хотели проверить откуда это стекло — это часть машины, или там есть куски из других источников (например, стеклотары для напитков), которые химически отличаются. На первых взгляд трудно определить наличие каких-то групп в этих данных.

Для того, чтобы разбить данные на кластеры, методу k-средних нужно знать предполагаемое число таких кластеров. Предположим, нам известно заранее, что таких кластеров должно быть два. Теперь основная идея очень проста. Нужно найти положения (координаты) центров двух кластеров так, чтобы все образцы были однозначно ассоциированы с одним из центров, а сами центры были достаточно устойчивы.

Для начала нужно определить исходное положение центров, или, другими словами, инициализировать

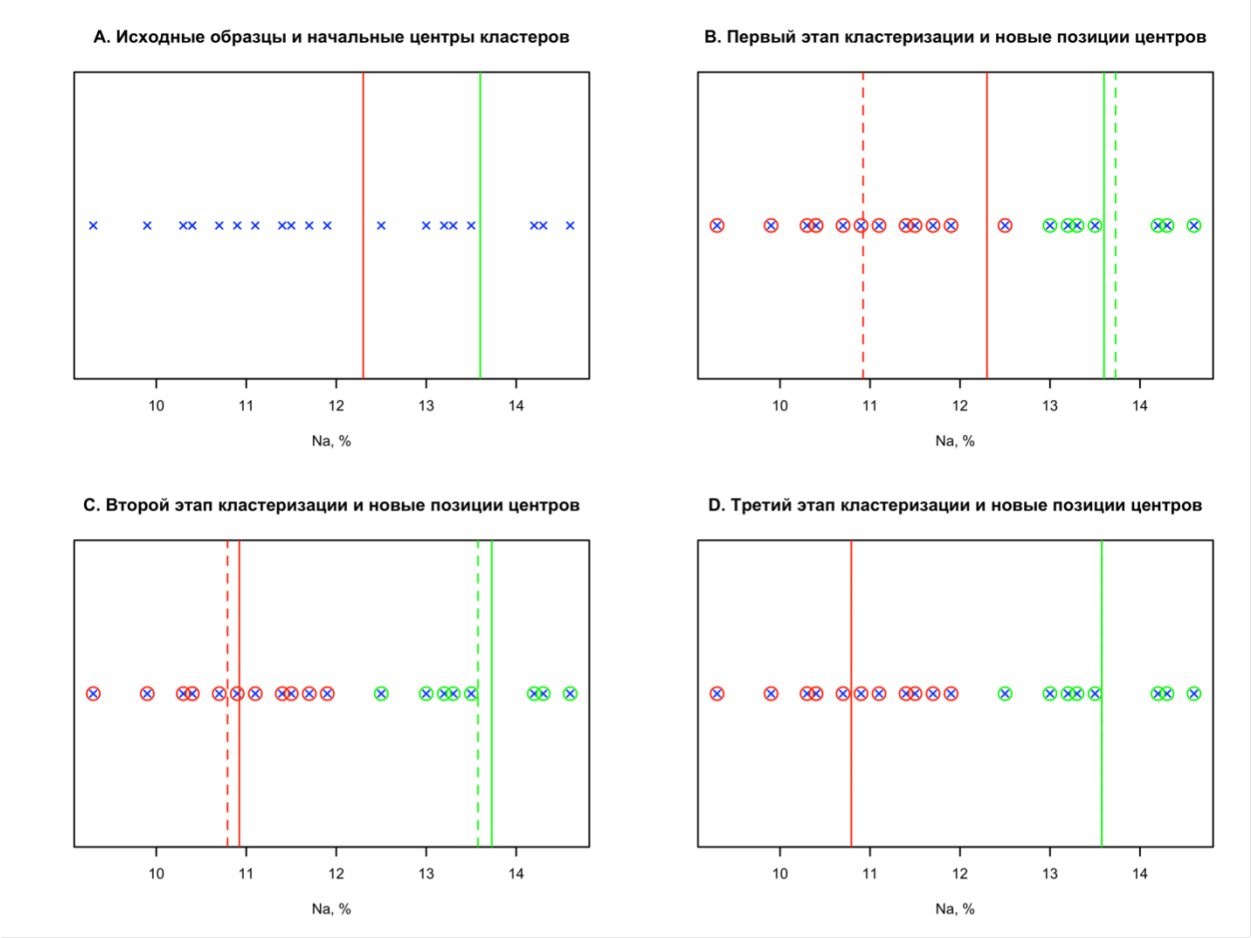


Рис. 5.4. Основные этапы кластеризации методом к-средних.

алгоритм. Так как у нас нет никаких априорных знаний, чтобы это сделать, то выберем их с помощью простого генератора случайных чисел. В данном случае у нас получилось 12.3 для первого центра и 13.6 для второго. График A на рисунке 5.4 показывает исходные данные в виде набора синих точек, а выбранные случайно центры в виде красной и зеленой линий. Очевидно, что выбор получился довольно неудачным — центры расположены близко друг к другу и оба смещены вправо. Посмотрим, справится ли метод k-средних с таким выбором.

Далее нужно проделать простой набор действий, а именно:

1. Вычислить расстояния от каждой точки до центра кластера 1
2. Вычислить расстояние от каждой точки до центра кластера 2
3. Разбить точки на кластеры по принципу наименьшего расстояния

Результаты этих действий представлены в таблице 5.1 (столбцы 3–6). Например, в первом образце содержание оксида натрия составляет 10.7%. Т.е. расстояние до первого кластера равно $\sqrt{(10.7-12.3)^2} = 1.6$, а расстояние до второго — $\sqrt{(10.7-13.6)^2} = 2.9$. Очевидно, что данный образец ближе к центру первого кластера, как мы его, соответственно, и помечаем.

График B на рисунке 5.4 показывает изначальное разделение образцов на два кластера с помощью цветных меток. Образцы, которые были ближе к центру первого кластера отмечены красными кружками, а образцы, которые ближе ко второму — зелеными.

Далее нам нужно скорректировать позиции центров с помощью очень простой операции — мы вычисляем их как средние значения. Т.е. сначала вычисляем среднее значение содержания оксида натрия для образцов, помеченных красными кругами, получаем новое значение центра первого кластера. Далее делаем тоже самое для «зеленых» образцов — получаем координату новой позиции центра второго кластера. Собственно, отсюда и происходит название метода — центры кластеров определяются как средние значения параметров образцов, которые находятся в этом кластере.

Новые позиции центров показаны на этом же графике с помощью пунктирных линий. Как можно заметить, новая позиция центра первого кластера (красная пунктирная линия) довольно сильно поменялась и теперь находится на левой половине графика. Центр второго кластера (зеленая пунктирная линия) сместился вправо, но не очень сильно. Ну и новые центры теперь находятся довольно далеко друг от друга.

После этого мы повторяем наши действия, а именно:

1. Перегруппируем образцы, вычисляя расстояния до новых центров
2. Вычисляем новые позиции центров по результатам перегруппировки

Алгоритм продолжает свою работу до тех пор, пока разница между предыдущей и новой позицией центров не станет меньше какого-то порогового значения, т.е. позиции центров внутри кластеров *стабилизируются*. Графики C и D на рисунке показывают результаты второго и третьего шагов алгоритма. Видно, что уже на

третьем шаге новые и старые центры кластеров не сильно отличаются друг от друга, а на четвертом шаге они совпадают, так как новой перегруппировки уже не потребовалось.

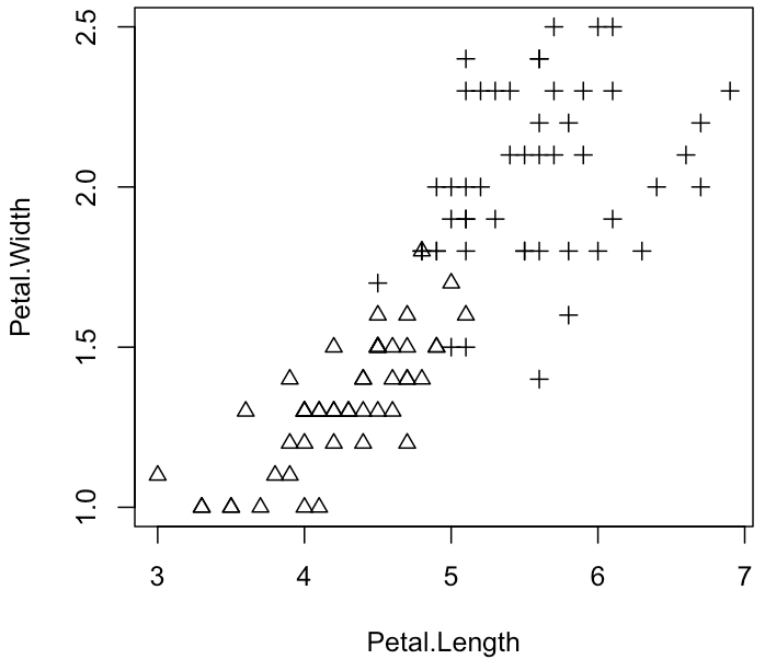
Метод k-средних легко работает и в пространствах размерности 2 или более, все, что ему нужно — это метрика для расстояний между точками и возможность вычислить среднее значение для каждой переменной. Однако, для многомерных данных есть один важный нюанс. Как мы уже отмечали ранее, расстояние Евклида в многомерном пространстве предполагает, что все переменные независимы. Если же есть взаимная корреляция между переменными, то либо следует использовать расстояние Махаланобиса, либо снижать размерность исходного пространства. Например, применить метод главных компонент и использовать k-средних в пространстве главных компонент, работая со счетами образцов.

Метод k-средних в R

Метод k-средних реализован в базовом наборе функций в R, т.е. вам не нужно устанавливать какие-то сторонние библиотеки для этого. В самом простом случае вам лишь нужно передать набор данных в виде матрицы, или фрейма с данными и число кластеров. Если нужно, то вместо числа можно указать начальные координаты центров (если вы можете определить их с помощью каких-то априорных знаний). Метод возвращает множество разных значений и метрик, наиболее важные — это матрица с координатами центров и вектор с номерами кластеров для каждой строки с данными.

Чтобы проиллюстрировать его работу, воспользуемся широко известным набором данных Фишера о геометрических размерах трех видов цветов ириса https://ru.wikipedia.org/wiki/Ирисы_Фишера. Для простоты ограничимся двумя классами, *virginica* и *versicolor* и двумя переменными, *Petal Width*, *Petal Length*. Блок кода ниже показывает, как подготовить данные и визуализировать их с помощью простой диаграммы рассеивания. Так как в данном случае истинный класс образцов нам известен, будем использовать это знание для проверки качества кластеризации.

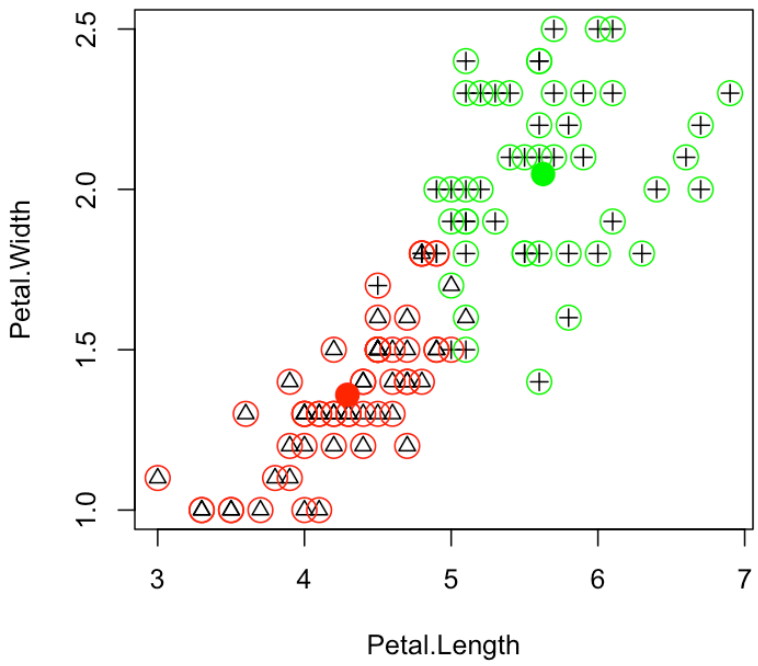
```
data(iris)
d <- subset(iris, Species == "versicolor" | Species == "virginica")
X <- subset(d, select = c(Petal.Length, Petal.Width))
plot(X, pch = as.numeric(d$Species))
```



Как можно видеть из графика, образцы, соответствующие двум видам ирисов, довольно сильно перекрываются (для простоты разные виды показаны на графике разными маркерами). Применим метод k-средних и покажем результаты кластеризации с помощью цветных окружностей, сделанных поверх исходных данных, как ранее в этой главе. Вот блок кода, который позволит это сделать.

```
# применить метод к-средних
r <- kmeans(X, 2)

# показать новый график для исходных данных
plot(X, pch = as.numeric(d$Species))

# показать центры кластеров
points(r$centers[, 1], r$centers[, 2], col = c("red", "green"), cex = 2, pch = 16)

# показать красные кружки вокруг образцов из первого кластера
points(X[r$cluster == 1, 1], X[r$cluster == 1, 2], pch = 1, cex = 2, col = "red")

# показать красные кружки вокруг образцов из второго кластера
```

```
points(X[r$cluster == 2, 1], X[r$cluster == 2, 2], pch = 1, cex = 2, col = "green")
```



Как можно заметить, большинство образцов *versicolor*, которые показаны на графиках в виде треугольников, были определены как элементы зеленого кластера, а образцы *virginica* (показаны в виде плюсов) как красный. Только три образца *versicolor* и два образца *virginica* отнесены к неправильному кластеру, что довольно неплохо, учитывая, насколько эти два класса образцов перекрываются в этом пространстве координат.

В качестве тренировки предлагаем запустить программу, указывая самим начальные положения центров, и исследовать, как разные начальные значения влияют на результат алгоритма.

5.5 Метод ближайших соседей (kNN)

Метод ближайших соседей (англ. *k nearest neighbors*, *kNN*) — один из самых простых, интуитивно понятных и при этом весьма эффективных методов классификации многомерных данных. Суть его можно выразить одной фразой: «скажи мне, кто твой сосед (или твои соседи), и я скажу кто ты». Действительно, схожие по свойствам образцы в многомерном пространстве свойств будут располагаться рядом (будут являться соседями).

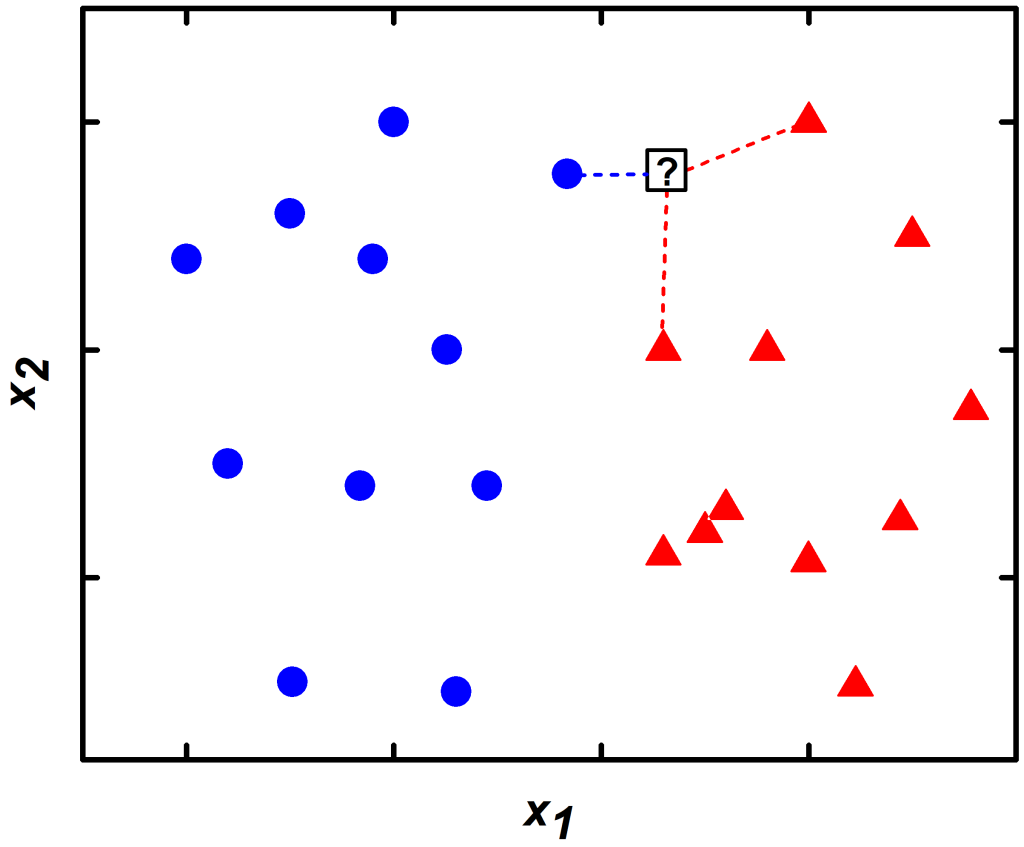


Рис. 5.5. Метод ближайших соседей.

Графическая иллюстрация метода ближайших соседей приведена на рис. 5.5. Этот метод является параметрическим и имеет параметр k — число ближайших соседей, которое принимается в расчет при установлении классовой принадлежности нового неизвестного образца, который обозначен знаком вопроса на рис. 5.5. Возьмем для начала $k = 1$. Образец, который наиболее близок к неизвестному, в данном случае принадлежит синему классу, следовательно, результатом классификации при $k = 1$ будет являться принадлежность неизвестного образца к синему классу. Если взять $k = 3$, то среди трех ближайших соседей неизвестного образца окажется два красных и один синий образец. Голосование большинством даст результат классификации в виде принадлежности неизвестного образца к красному классу.

Реализация метода kNN довольно проста. На первом этапе рассчитываются расстояния от всех образцов обучающей выборки до неизвестного образца. В качестве расстояния в зависимости от задачи могут быть использованы различные метрики, например, Евклидово, или расстояние Махаланобиса. Далее расстояния сортируются по возрастанию и определяются ближайшие к неизвестному образцу k соседей. Подсчитывается число соседей в каждом из классов и на основании этого принимается решение о принадлежности к тому, или иному классу.

Рассмотрим процедуру проведения расчетов по методу kNN на простом примере. Пусть у нас есть два класса в каждом из которых по три образца (“Класс 1” и “Класс 2” на рис. 5.6). Каждый образец описан двумя переменными – x_1 и x_2. Также имеется неизвестный образец (на рис. 5.6 обозначен знаком “?”). Давайте проведем расчеты, что бы понять к какому классу принадлежит неизвестный образец. Для этого рассчитаем Евклидово расстояние от неизвестного образца до всех образцов обоих классов и отсортируем эти расстояния по возрастанию (столбец r на рис. 5.6 — разными цветами показаны расстояния от неизвестного образца до образцов разных классов). При $k = 3$ среди ближайших соседей неизвестного образца оказалось два синих (большинство) и один красный, следовательно, при $k = 3$ неизвестный образец будет отнесен ко второму классу. При $k = 5$ среди ближайших соседей оказывается большинство образцов красного класса, результат классификации меняется. Попробуйте сами произвести эти несложные расчеты.

Очевидно, что выбор конкретного значения k является критически важным и непосредственно влияет на результат классификации. Выбор оптимального параметра k осуществляется на этапе оптимизации модели. Для этого проводят классификацию на тестовом наборе при различных значениях k и определяют качество модели по какому-либо критерию (например, число неправильно классифицированных образцов). Значение k при котором модель дает наилучшие результаты является оптимальным. При небольшом количестве обучающих образцов можно воспользоваться перекрестной проверкой, в ходе которой каждый образец считается неизвестным и классифицируется при заданном значении k. Следует упомянуть, что k лучше выбирать нечетным.

Метод ближайших соседей можно применять и для многоклассовой классификации, в этом случае решение о принадлежности неизвестного образца определяется также по результатам «голосования» большинства ближайших образцов из разных классов.

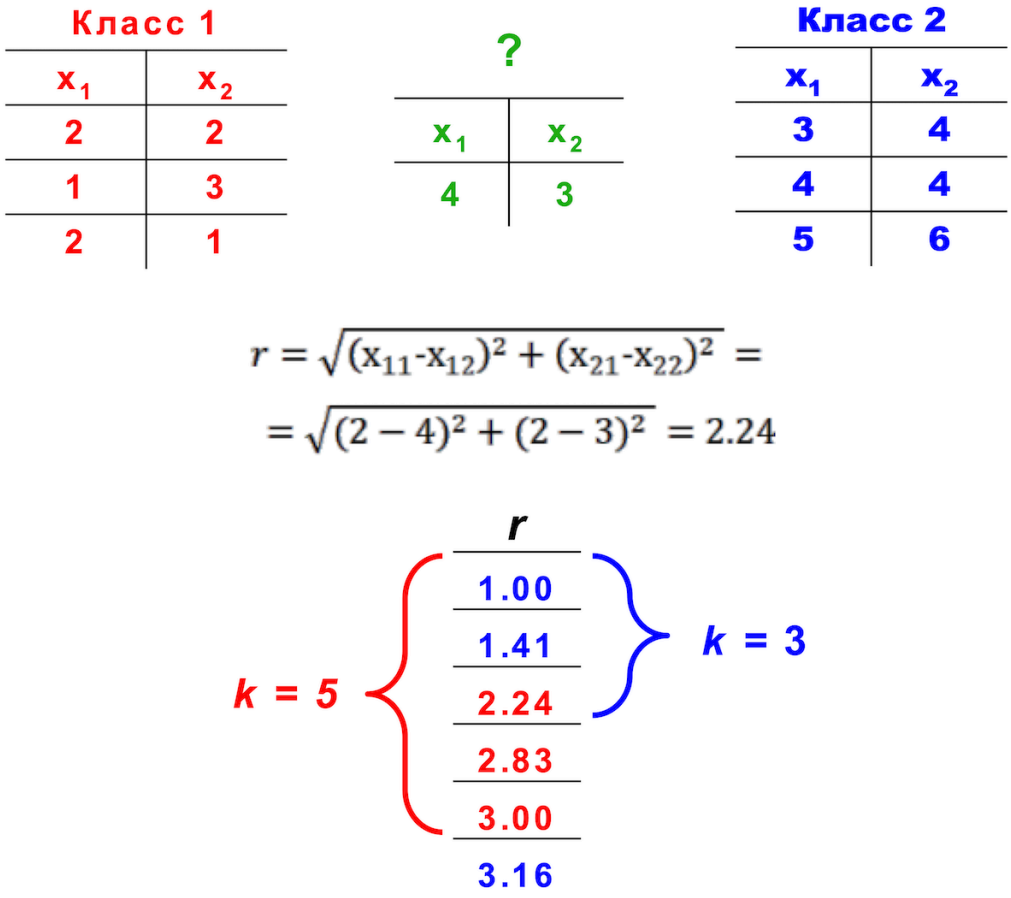


Рис. 5.6. Пример классификации методом ближайших соседей.

В случае данных высокой размерности, содержащих большое число переменных, описывающих образец, часто прибегают к использованию МГК (Chapter 2) и проводят kNN моделирование на матрице счетов. При этом появляется еще один параметр — число ГК в модели, который также необходимо оптимизировать.

Реализация в R

Для реализации kNN в R имеется следующая функция из пакета `class`:

```
knn(train, test, cl, k = 3)
```

- `train` – матрица, или фрейм, содержащий образцы обучающего набора
- `test` – матрица, или фрейм, содержащий образцы проверочного набора
- `cl` – вектор, содержащий информацию о классовой принадлежности образцов обучающей выборки
- `k` – число ближайших соседей, принимаемых в расчет

Давайте построим классификатор для набора данных по ирисам Фишера, который мы уже использовали для примера с кластеризацией. При числе ближайших соседей $k = 3$. Фрейм с этими данными входит в базовую установку R и доступен по команде `iris`.

Для начала разобьем наши данные на обучающую (`trainSet`) и проверочную (`testSet`) выборку. 70% образцов войдут в обучающую выборку, остальные в проверочную:

```
tr.index <- sample(1:nrow(iris), nrow(iris) * 0.7)
trainSet <- iris[tr.index,]
testSet <- iris[-tr.index,]
```

Строим классификационную модель по алгоритму kNN с $k = 3$, результаты классификации для проверочного набора записываем в переменную `k3`:

```
library(class)
k3 <- knn(trainSet[1:4], testSet[1:4], trainSet$Species, k = 3)
```

Посмотрим на результаты классификации. Это вектор, содержащий наименования класса, приписанные моделью каждому образцу из проверочного набора:

```
show(k3)
```

```
 [1] setosa     setosa     setosa     setosa     setosa     setosa
 [7] setosa     setosa     setosa     setosa     setosa     setosa
[13] setosa     setosa     setosa     setosa     setosa     versicolor
[19] versicolor versicolor versicolor versicolor versicolor versicolor
```

```
[25] versicolor versicolor versicolor versicolor versicolor virginica
[31] virginica  virginica  virginica  virginica  versicolor virginica
[37] virginica  virginica  virginica  virginica  virginica  virginica
[43] virginica  virginica  virginica
Levels: setosa versicolor virginica
```

Построим матрицу ошибок для этого классификатора, сравнив спрогнозированные значения классов с их истинными значениями для проверочного набора:

```
table(k3, testSet$Species)
```

```
k3           setosa versicolor virginica
  setosa         17          0         0
  versicolor      0         12         1
  virginica       0          0        15
```

Обратите внимание, что ваши результаты могут отличаться от приведенных здесь, поскольку они зависят от конкретного разбиения образцов на обучающую и проверочную выборки, которое в нашем примере выполнено случайным образом.

Предлагаем вам самостоятельно изучить результаты классификации при различных значениях k. В случае небольшого числа образцов, когда невозможно их разделить на обучающий и проверочный наборы без ущерба для качества модели, можно воспользоваться перекрестной проверкой, которая реализована функцией `knn.cv`:

```
knn.cv(train, cl, k = 3)
```

- `train` – матрица, или фрейм, содержащий образцы обучающего набора
- `cl` – вектор, содержащий информацию о классовой принадлежности образцов обучающей выборки
- `k` – число ближайших соседей, принимаемых в расчет

На примере данных `iris` перекрестную проверку можно реализовать следующем образом:

```
kcv <- knn.cv (iris[,1:4], iris$Species, k = 7)
table(kcv, iris$Species)
```

```
kcv          setosa versicolor virginica
```

```
setosa         50          0         0
versicolor      0         46         1
virginica       0          4        49
```

5.6 Линейный и квадратичный дискриминантный анализ

Линейный дискриминатный анализ (англ. *linear discriminant analysis*, *LDA*) – один из первых способов классификации, предложенный Рональдом Фишером для многомерных данных в 1936 году. Классификация по методу ЛДА основана на поиске гиперплоскости, разделяющей классы в пространстве переменных. В случае двух переменных такая гиперплоскость задается уравнением прямой. В случае трех переменных — уравнением плоскости, при большем числе переменных — гиперплоскость. ЛДА, так же как и kNN, является методом дискриминации, то есть каждый классифицируемый объект будет принадлежать к одному и только одному классу.

Для применения ЛДА необходимо выполнение двух важных условий: 1) распределение образцов в пространстве переменных для каждого класса должно подчиняться нормальному закону; 2) дисперсия в разных классах должна быть одинакова.

Рассмотрим принципы ЛДА на примере двух классов, образцы которых описываются двумя переменными x_1 и x_2 (рис. 5.7). Образцы обоих классов имеют нормальное распределение по каждой из переменных. Дисперсии для каждой из переменных одинаковы в обоих классах, а распределения отличаются только средними значениями (центрами распределений). При этом для переменных x_1 и x_2 дисперсии могут различаться.

Так как в данном случае значения отдельных переменных распределены нормально (это одно из условий применения ЛДА), то центр этих значений для одной переменной определяется ее средним, а их разброс — дисперсией, или стандартным отклонением. Теоретическое распределение значений задается известной кривой в форме колокола. График слева на рис. 5.7. показывает теоретические кривые распределения плотности вероятностей значений переменной x_2. Для удобства представления график повернут на 90 градусов. Цвет кривой соответствует конкретному классу. Как можно видеть, условие равенства дисперсий приводит к тому, что кривые одинаковы по своей форме, но смещены по отношению друг к другу, так как классы имеют разное среднее значение для x_2.

График снизу показывает теоретические распределения плотности вероятностей для переменной x_1. Здесь мы можем наблюдать похожую картину, за одним исключением – степень перекрытия этих двух кривых гораздо меньше, по сравнению с переменной x_2. Другими словами, переменная x_1 играет более важную роль в различии между этими двумя классами.

График посередине — это классическое представление данных уже в двумерном пространстве. Так, центр каждого класса на этом графике задается пересечением средних значений для каждой переменной. Вокруг центров мы можем задать набор эллипсов, которые будут показывать точки, удаленные от центра

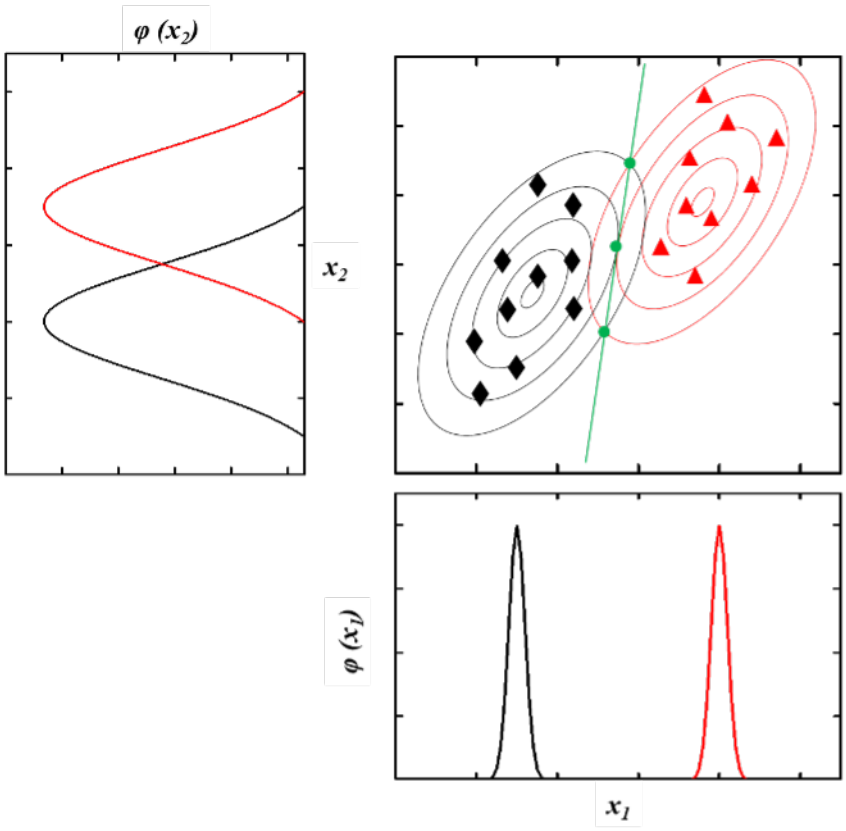


Рис. 5.7. Линейный дискриминатный анализ

на определенную величину стандартного отклонения. Скажем, если построить такой эллипс для двух стандартных отклонений, то 95% точек, принадлежащих классу, будут лежать внутри него. По сути, контуры эллипса задают точки, равноудаленные от центра по расстоянию Махаланобиса.

Так как дисперсии точек, принадлежащих разным классам, для отдельно взятой переменной одинаковы, то форма эллипсов (направление и величина их осей, а также эксцентриситет) также идентична и эллипсы отличаются друг от друга лишь смещением в пространстве. Если провести прямую через точки пересечения эллипсов — это и будет искомая разделительная прямая.

Стоит заметить, что разделительная прямая почти перпендикулярна оси x_1. Это связано с тем, что как мы заметили, x_1 важнее для разделения образцов между классами.

Если обобщить эти рассуждения на пространство более высокой размерности, то эллипсы превратятся в гиперэллипсоиды, а разделительная прямая в гиперплоскость, однако по сути ничего не поменяется.

Рис. 5.8. Пример расчетов в линейном дискриминантном анализе.

Пример расчетов в методе ЛДА приведен на рис. 5.8. Воспользуемся теми же исходными данными, на которых мы рассматривали kNN рис. 5.6. Для того, чтобы рассчитать уравнение прямой необходимо

провести следующие действия. Для начала рассчитываются векторы средних значений для каждого класса (m_A и m_B). Матрицы данных обоих классов центрируются на соответствующие значения средних. По центрированным матрицам рассчитывается матрица ковариации **S** (в формуле ковариации на рис. 5.8, n_1 и n_2 – число образцов в первом и втором классе соответственно). Уравнение прямой, разделяющей классы, можно записать в матричном виде:

$$\mathbf{x}\mathbf{w}_1^{\mathrm{T}} - \mathbf{v}_1 = \mathbf{x}\mathbf{w}_2^{\mathrm{T}} - \mathbf{v}_2 \tag{5.4}$$

Где $\mathbf{x}\mathbf{w}_1^{\mathrm{T}} - \mathbf{v}_1 = \mathbf{f}_1$ и $\mathbf{x}\mathbf{w}_2^{\mathrm{T}} - \mathbf{v}_2 = \mathbf{f}_2$ называются счетами ЛДА, с помощью которых можно легко установить принадлежность неизвестного образца к классу — если для нового образца $f_1 > f_2$, то он принадлежит классу 1, и наоборот. На рис. 5.8 приведен расчет коэффициентов разделяющей прямой для нашего простого примера. Графически классификационная модель представлена на рис. 5.9.

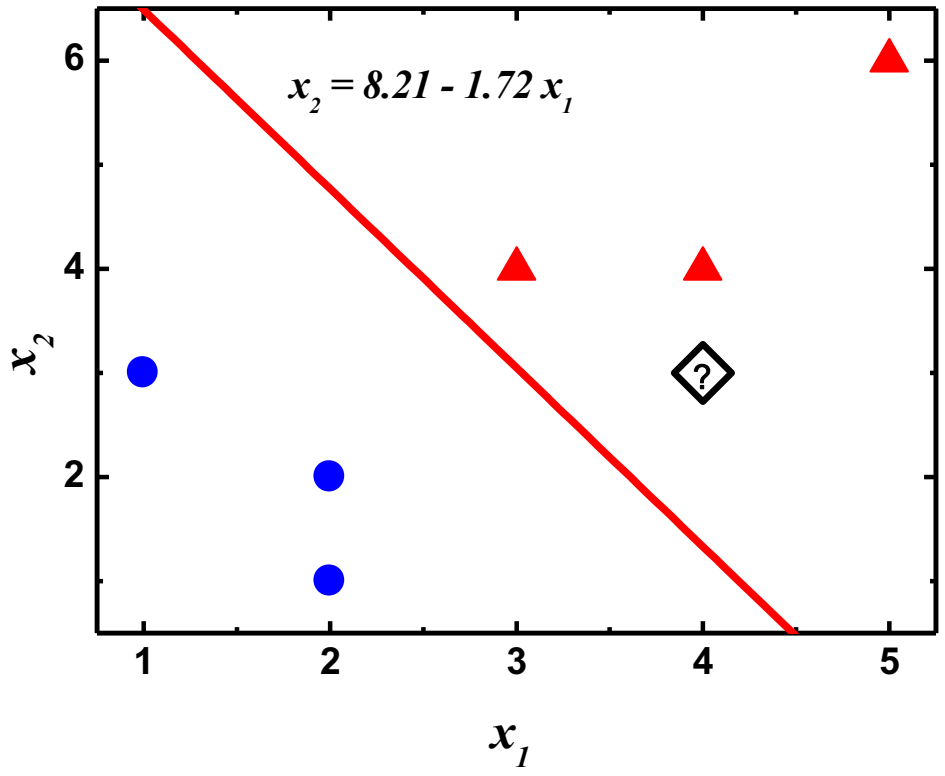
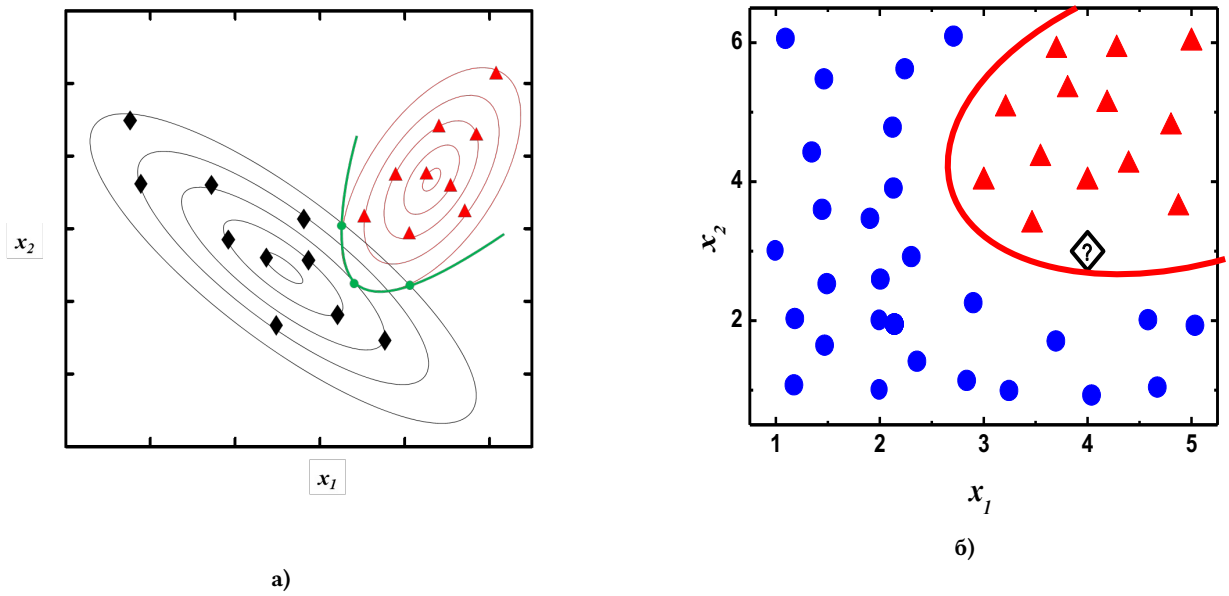


Рис. 5.9. Граница между классами в линейном дискриминантном анализе. Новый образец помечен знаком вопроса, по результатам ЛДА он будет отнесен к красному классу.

Квадратичный дискриминантный анализ (англ. *quadratic discriminant analysis*, *QDA*) также основан на гипотезе о нормальном распределении плотности вероятности, но дисперсия в классах может различаться.

Это приводит к тому, что эллипсы на рис. 5.7 будут иметь разный эксцентриситет и направление осей (рис. 5.10а). В этом случае для разделения классов используется более сложная функция – парабола (отсюда и название — квадратичный дискриминантный анализ) — рис. 5.10.



а) б)

Рис. 5.10. Квадратичный дискриминантный анализ (а) и пример разделяющей кривой в квадратичном дискриминатном анализе.

Реализация в R

Для построения ЛДА модели в R можно воспользоваться функцией `lda()` из пакета `MASS`. Обратимся к уже рассмотренным нами ранее данным — ирисам Фишера. Разбиение на обучающую и проверочную выборки проведем также как и ранее в предыдущих двух разделах.

Построим ЛДА модель на обучающем наборе и запишем ее в переменную `l`:

```
library(MASS)
l <- lda (formula = Species ~ . , data = trainSet, prior = c(1, 1, 1)/3)
```

В этой функции мы используем следующие аргументы: `formula` — задает правило разбиения фрейма на предикторы и названия классов. Запись `Species ~ .` означает, что названия классов находятся в переменной `Species`, а в качестве предикторов используются все оставшиеся переменные фрейма. Аргумент `data` задает название фрейма, который используется для построения модели. Аргумент `prior` задает априорные вероятности принадлежности образцов к классу, если его не указывать то функция `lda()` рассчитает эти значения самостоятельно исходя из количества образцов соответствующих классов в обучающем наборе. Запись `c(1,1,1)/3` означает, что априорно мы считаем принадлежность образца к

любому из трех классов равновероятной.

Для прогнозирования принадлежности к классам новых образцов используется функция predict. Спрогнозируем классы для образцов нашего проверочного набора и запишем результаты прогнозирования в переменную `pred`:

```
pred <- predict (object = l, newdata = testSet)
```

Аргумент `object` задает переменную в которой хранится модель – в нашем случае `l`; аргумент `newdata` – название фрейм с проверочным набором. Построим матрицу ошибок:

```
table(pred$class, testSet$Species)
```

```
            setosa versicolor virginica
setosa         17          0         0
versicolor      0         12         0
virginica       0          0        16
```

5.7 Логистическая регрессия

Логистическая регрессия (ЛР) —- еще один популярный метод многомерной дискриминации. Метод основан на построении регрессионного уравнения, связывающего набор переменных, характеризующих образец, с его классовой принадлежностью. Это метод бинарной классификации и одна модель позволит разделить объекты только двух классов. ЛР использует уравнение, в котором зависимая переменная y (кодирующая принадлежность к классу) связана с независимыми переменными x (описывающими свойства образца). Переменная y может изменяться в интервале от нуля до единицы, где ноль означает принадлежность к первому классу, а единица — ко второму. Общий вид уравнения логистической регрессии приведен ниже (уравнение 5.5). Такая функция называется сигмоидальной. Регрессионные коэффициенты b_j уравнения 5.5 находятся в процессе обучения модели, которое осуществляется с помощью обучающей выборки образцов с достоверно известной классовой принадлежностью.

$$y(x_1, x_2, ..., x_m) = \frac{1}{(1 + e^{-(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_m x_m)})} \tag{5.5}$$

Найденные регрессионные коэффициенты в дальнейшем используются для прогнозирования классовой принадлежности новых образцов. Рассмотрим работу ЛР на примере одной независимой переменной x_1. Пусть у нас есть два класса, первый из которых характеризуется низкими значениями этой переменной, а второй — высокими. Закодируем первый класс, как $y = 0$, а второй $y = 1$. Образцы двух классов изображены

на рис. 5.11 символами разной формы и цвета. На этом же рисунке показано уравнение сигмоиды для случая одной независимой переменной, а красной линией обозначен сам вид логистической функции. Читатель легко может проверить, что значение функции стремится к нулю при малых значениях x_1 и стремится к единице при больших значениях x_1.

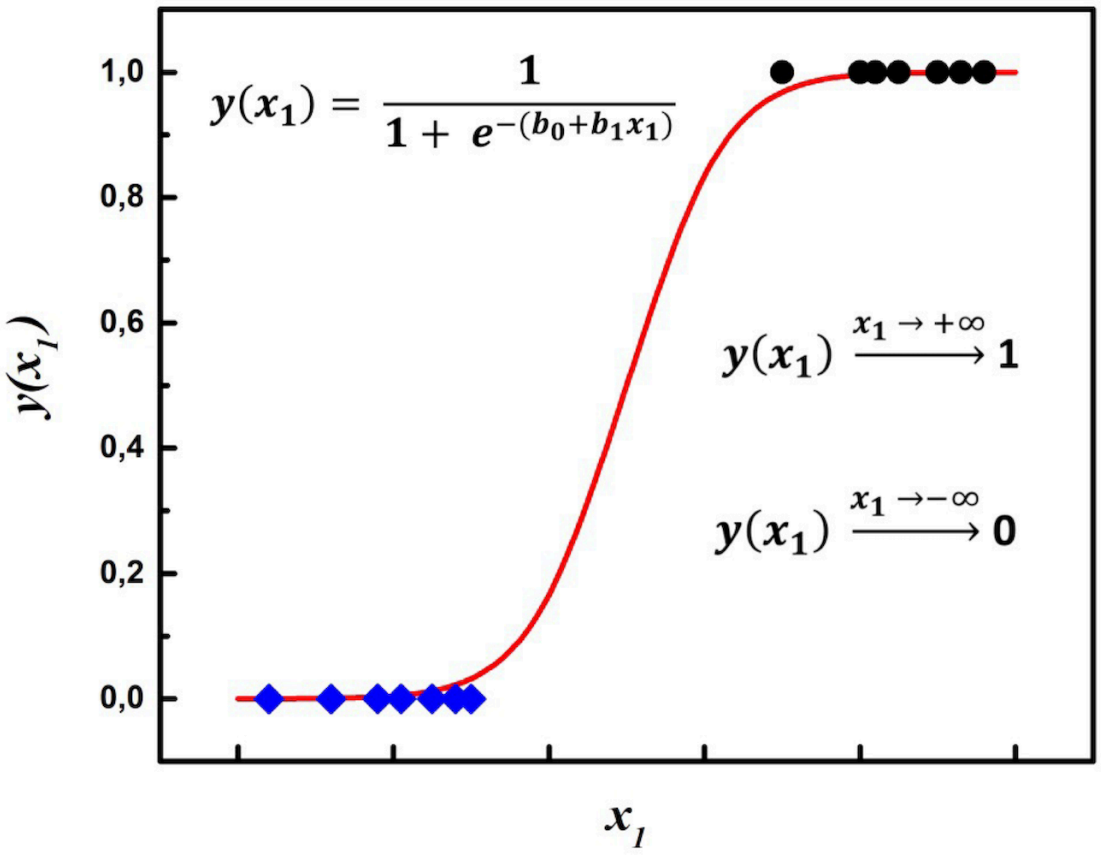


Рис. 5.11. Логистическая регрессия.

Вид сигмоиды зависит от коэффициентов b_0 и b_1. Задача обучения модели сводится к нахождению таких коэффициентов, которые обеспечат прохождение сигмоиды с минимальным отклонением от обучающих образцов. Коэффициенты находят традиционными математическими способами решения нелинейных уравнений, например, методами градиентного спуска, или максимального правдоподобия.

После нахождения коэффициентов можно прогнозировать классовую принадлежность новых образцов. Для этого их параметр x_1 подставляется в уравнение сигмоиды и вычисляется значение y. Это значение для новых образцов не обязательно будет равняться нулю, либо единице, а может принимать любое

значение в этом диапазоне. Общий принцип отнесения к классу заключается в сравнении вычисленного значения y с пороговым значением (англ. *decision boundary*, *граница принятия решения*). В простейшем случае это пороговое значение принимается за 0.5. При $y < 0.5$ образец относят к первому классу, при > 0.5 ко второму. На практике конкретное значение границы принятия решений выбирается на основе результатов прогнозирования в проверочном наборе (образцов с известной классовой принадлежностью, не использовавшихся для обучения). На рис. 5.12 приведен пример графика «введено – найдено» для классификационной модели на основе логистической регрессии, построенный для проверочных образцов. Синими ромбами показаны образцы первого класса, черными кругами – второго.

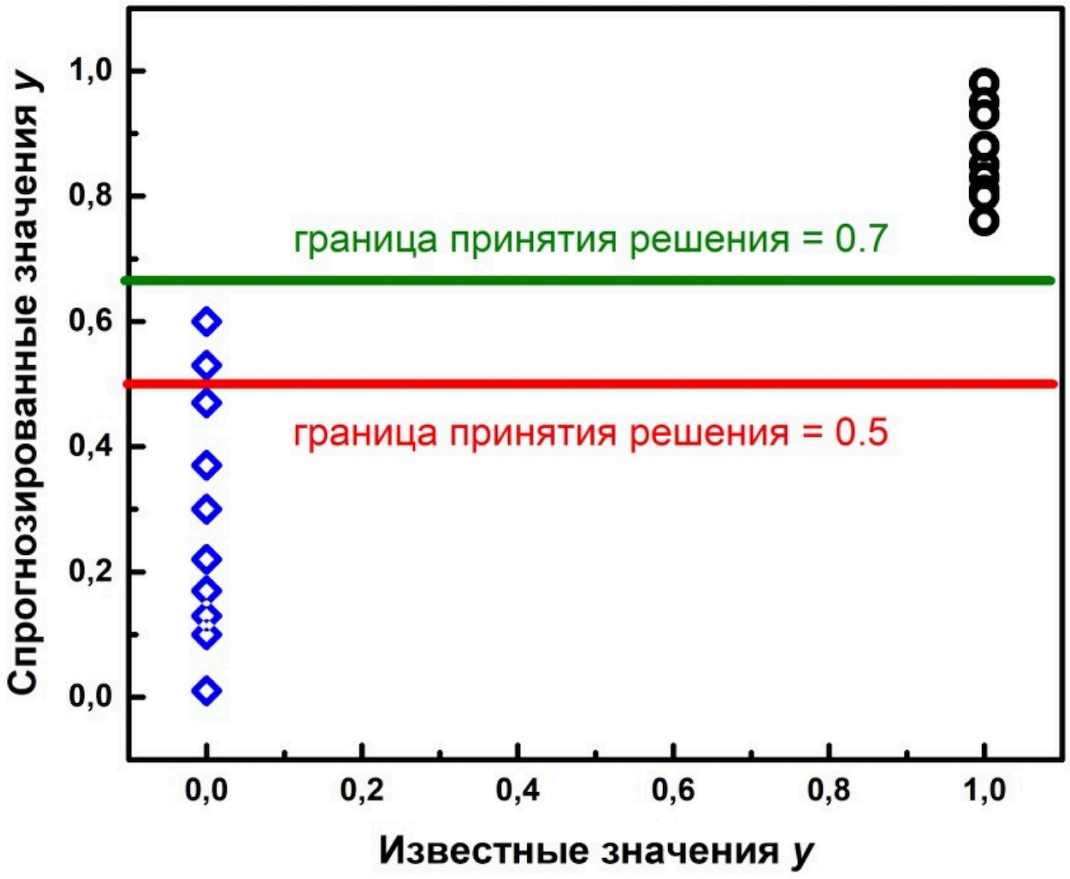


Рис. 5.12. График известной и спрогнозированной принадлежности к классу для логистической регрессии.

Видно, что при использовании границы принятия решения равной 0.5 два образца из первого класса будут ошибочно отнесены ко второму, поскольку расположены выше этого значения. В данном случае оптимальной границей принятия решения будет величина 0.7, показанная зеленой линией. При этом все образцы из проверочного набора будут классифицированы верно. В качестве критериев оптимизации

модели (поиска оптимального значения границы принятия решения) используются рассмотренные ранее метрики чувствительности, специфичности и точности модели, рассчитанные для проверочных образцов.

Методологию логистической регрессии легко распространить на большее число переменных x, в этом случае для расчета показателя экспоненты в знаменателе уравнения сигмоиды используется больше слагаемых (уравнение 5.5).

Не смотря на то, что ЛР является методом бинарной классификации, его можно успешно применять и в многоклассовых задачах. Для этого используется подход «один против всех» (англ. *one vs. all*). При этом, в случае, например, трех классов будут построены три ЛР модели. В каждой из них образцы одного из трех классов кодируются нулями, а образцы двух оставшихся — единицами. Прогнозирование новых образцов также проводится всеми тремя моделями, каждая из которых будет показывать принадлежность образцов к «своему» классу.

Основным ограничением метода логистической регрессии является невозможность использования большого числа независимых переменных x, потому что метод максимального правдоподобия, который применяется для оценки регрессионных коэффициентов уравнения 5.5. плохо справляется с такими задачами. Поскольку для современных измерительных приборов (например, спектрометры, хроматографы) характерны данные с большим числом переменных x, то для решения химических задач метод логистической регрессии применятся сравнительно редко. Кроме того для применения логистической регрессии необходимо иметь достаточно большие выборки обоих моделируемых классов. В противном случае точность классификатора снизится.

Реализация в R

Ниже приведена реализация логистической регрессии в R на примере уже известного нам набора данных – `iris`. Так как логистическая регрессия является двухклассовым методом классификации, то мы воспользуемся только частью данных — классами *setosa* и *versicolor*. Для этого введем специальный фрейм `irisLR`, в котором будут первые 100 образцов:

```
irisLR <- iris[1:100,]
```

Переобозначим столбец `Species`, содержащий название классов, в числовом виде. После этой операции `setosa = 0`, а `versicolor = 1`:

```
irisLR$Species <- as.numeric(irisLR$Species) - 1
```

Разобьем наши данные на обучающую (`trainSet`) и проверочную (`testSet`) выборку. 70% образцов войдут в обучающую выборку, остальные в проверочную:

```
tr.index <- sample(1:nrow(irisLR), nrow(irisLR)*0.7)
trainSet <- irisLR[tr.index, ]
testSet <-irisLR[-tr.index, ]
```

Строим регрессионную модель, используя команду `glm` из пакета `stats`:

```
lr <- glm(trainSet[,5] ~ ., data = trainSet[,1:4])
```

Прогнозируем маркеры классов для проверочного набора и заносим результат в переменную `predictions`:

```
predictions <- plogis(predict(lr, testSet))
```

Переменная `predictions` содержит значение функции логистичекесой регрессии для проверочных образцов. Для определения принадлежности этих образцов к классу, необходимо ввести границу решения. Для начала проверим результаты классификации с границей решения 0.5. Результаты классификации будут записаны в переменную `pred`:

```
pred <- predictions
pred[pred>0.5] <- 1
pred[pred<0.5] <- 0

# матрица ошибок
table(pred, testSet$Species)
```

```
pred  0  1
   0  5  0
   1  9 16
```

Видно, что значительная часть образцов класса 0 (`setosa`) при данной границе классифицируется неверно (9 образцов из 15 прогнозируются как класс *versicolor*). Попробуем изменить границу принятия решения, используем значение 0.6:

```
pred <- predictions
pred[pred<0.6] <- 0
pred[pred>0.6] <- 1
table(pred, testSet$Species)
```

```
pred  0  1
   0 14  0
   1  0 16
```

Теперь все образцы классифицированы верно.

5.8 SIMCA

Как мог заметить читатель, рассмотренные в предыдущих разделах методы классификации работают в исходном пространстве переменных, и, как следствие, подвержены «проклятию» размерности. Когда классифицируемые образцы описываются большим набором переменных, удобно пользоваться проекционными методами, чтобы избежать этой проблемы. Метод мягкого независимого моделирования классовых аналогий (англ. *soft independent modeling of class analogy*, *SIMCA*) является одним из наиболее популярных классификационных методов этого типа. Метод предложил в 1977 году Сванте Вольд и с тех пор этот алгоритм прочно вошел в арсенал хемометрики. В основе SIMCA лежит метод главных компонент.

Рассмотрим SIMCA в одноклассовом варианте, т.е. задача классификации заключается в том, чтобы для неизвестного образца предсказать принадлежит ли он данному классу, или нет. На первом этапе на основе обучающих образцов данного класса строится МГК модель (глава 2). Неизвестные образцы проецируются на эту модель и для каждого из них рассчитываются две метрики h (расстояние от проекции образца до начала координат) и q (расстояние от образца до модели). Проецирование на модель осуществляется путем умножения матрицы данных для новых образцов на матрицу нагрузок, полученную при построении МГК модели для обучающей выборки (глава 2). Рассчитанные значения h и q для новых образцов сравниваются с граничными значениями этих величин, рассчитанных для обучающей выборки, и на основе этого принимается решение о принадлежности новых образцов к классу. Значения h и q для i-го образца рассчитываются по следующим формулам:

$$h_i = \sum_{a=1}^{A} \frac{t_{ia}^2}{\lambda_a} \tag{5.6}$$

$$q_i = \sum_{j=1}^{J} e_{ij}^2 \tag{5.7}$$

В этих формулах A — это число ГК в модели, описывающей класс; t_{ia} — счета i-го образца, спроецированного на компоненту a, λ_a – нормировочный множитель для расчета расстояния Махаланобиса (собственное значения этой компоненты, см. главу по МГК); e_{ij} – элемент матрицы остатков для строки i и столбца j, получаемой после проецирования нового образца на модель класса с А ГК; J – число переменных.

Графическая интерпретация h и q приведена на рис. 5.13. Пусть исходное пространство данных **X** задано тремя переменными (x_1, x_2, x_3). Закрытыми точками обозначены образцы обучающей выборки, на основе которых построена модель МГК (серая плоскость), заданная двумя главными компонентами ($A = 2$). Новый образец, классовую принадлежность которого нам нужно определить, проецируется на эту плоскость. В этом варианте q будет представлять собой расстояние от неизвестного образца до плоскости ГК (расстояние до модели), а h — расстояние от проекции неизвестного образца до центра модели.

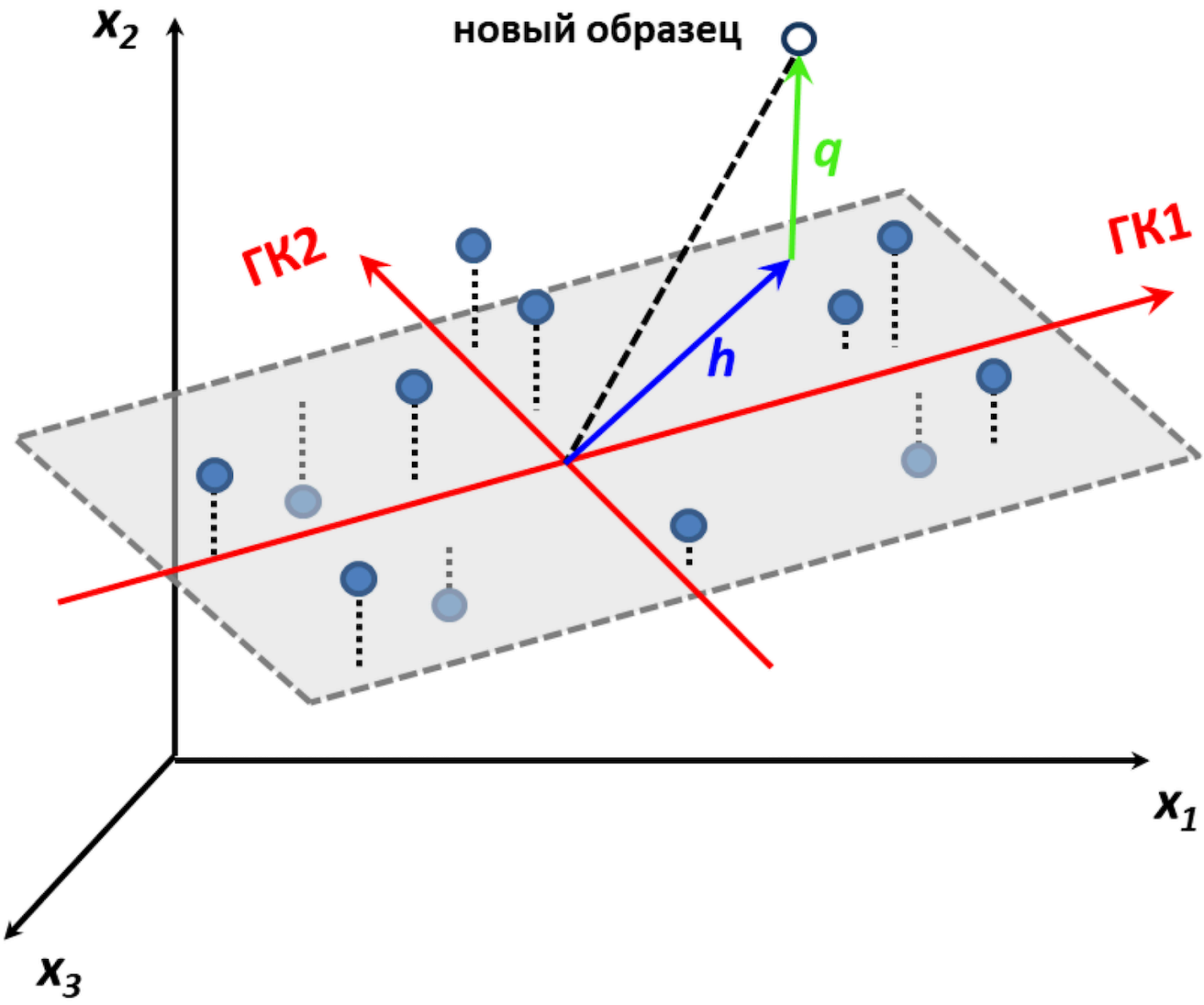


Рис. 5.13. Классификация по методу SIMCA.

Величины h и q, как видно из рисунка и формул, зависят от числа ГК (A), которое является основным параметром SIMCA. С помощью подбора числа ГК можно оптимизировать модель, уменьшая число неверно классифицированных образцов в проверочном наборе. Граничные значения h и q рассчитываются

следующим образом: для всех образцов обучающей выборки рассчитывают величины этих параметров; предполагается, что они подчиняются распределению χ^2; на основании этого распределения при заданном уровне значимости α (как правило, выбирают $\alpha = 0.05$) выбираются критические (граничные) значения h и q.

Существует несколько вариантов принятия решений о принадлежности неизвестного образца к классу на основе критических значений параметров. В наиболее простом варианте h и q неизвестных образцов сравниваются с критическими. Образец принадлежит классу только в том случае, если и h, и q будут меньше критических. Если хотя бы одно из этих условий не выполнено, то образец считается не принадлежащим классу (рис. 5.14). Другие варианты принятия решений подробно описаны в работе DOI: 10.1002/cem.3250.

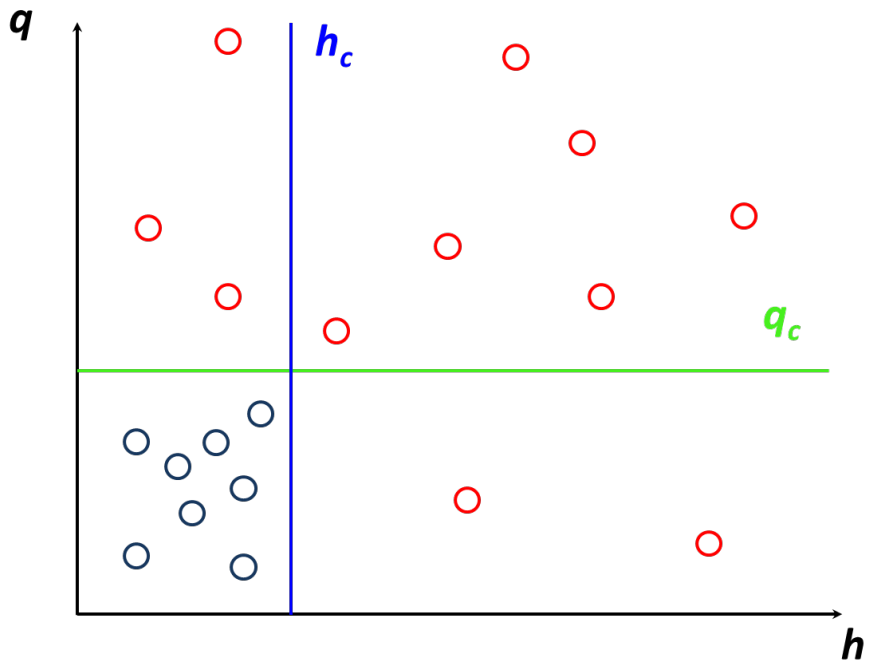


Рис. 5.14. Правило принятия решений в SIMCA. Красным цветом отмечены образцы не принадлежащие классу.

Метод SIMCA легко обобщается на большее число классов. В этом случае для каждого моделируемого класса строится своя модель МГК, рассчитываются свои значения h и q, с которыми сравниваются параметры новых неизвестных образцов. Стоит отметить, что при таком подходе каждый новый образец может либо принадлежать одному классу, либо принадлежать сразу нескольким классам, либо не принадлежать ни одному классу. В случае двух классов для визуализации результатов классификации удобно пользоваться графиком Кумана (Рис. 6.10).

На этом графике откладываются расстояния от новых образцов до модели первого класса и до модели второго класса. На рисунке можно выделить четыре области: 1 – образцы, принадлежащие первому классу;

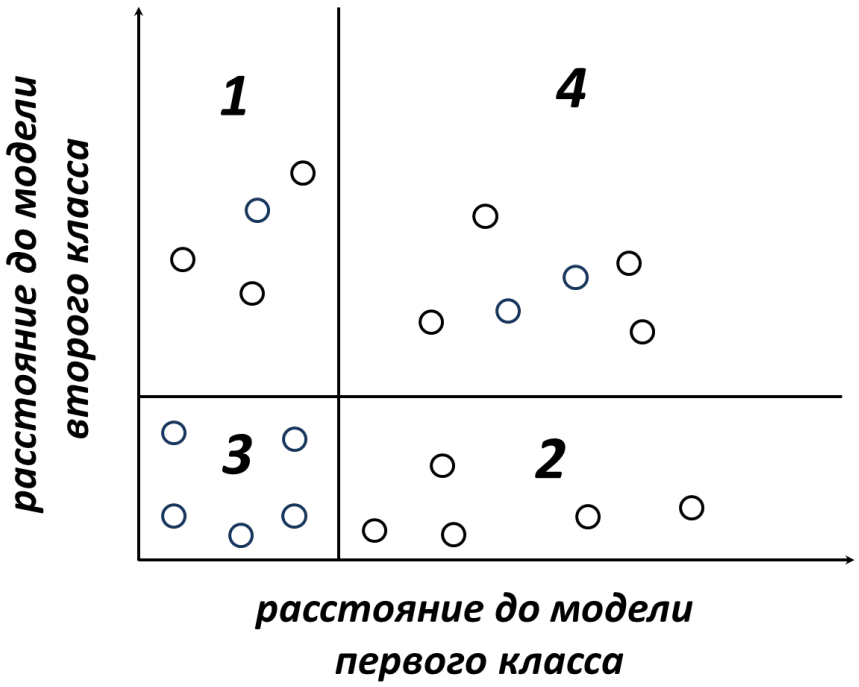


Рис. 5.15. График Кумана для двух классов.

2 – образцы, принадлежащие второму классу; 3 – образцы, принадлежащие обоим классам; 4 – образцы, не принадлежащие ни одному классу.

Примеры реализации в R

Для начала рекомендуем еще раз посмотреть все примеры кода из главы 2. Это будет полезно, так как SIMCA — это по сути МГК с небольшой дополнительной надстройкой, и все основные нюансы, в том числе и то, как выбирать критические расстояния мы рассмотрели в главе про МГК.

Ниже мы рассмотрим пример реализации SIMCA в пакете `mdatools`. В качестве тренировки мы предлагаем вам реализовать этот метод самостоятельно, используя код, написанный нами для МГК.

Для построения модели SIMCA нужно воспользоваться функцией `simca()`. Она почти полностью повторяет функционал функции `pca()` из этого же пакета, но несколько параметров требуют дополнительного внимания в этом случае.

Во-первых, в качестве второго аргумента нужно задать имя класса. Это обычный текст, но важно, что если вы собираетесь дальше тестировать модель с помощью тестового набора с известными классами, то название класса, которое вы даете при построении модели, должно совпадать с названием этого класса, который будет находится в векторе с метками классов тестового набора.

Во-вторых, нужно определить параметр `alpha`, который задает уровень значимости для поиска выбросов в

МГК, в случае SIMCA он регулирует положение границы принятия решений. При `alpha = 0.05` эта граница будет выбрана так, чтобы обеспечить чувствительность около 95%. Другими словами 5% образцов из класса, для которого мы строим модель, будут отвергнуты этой моделью (станут False Negatives).

Ну и, наконец, параметр `lim.type` задает способ построения границы принятия решений. Если необходима прямоугольная граница, описанная в этой главе, то нужно использовать `lim.type = "chisq"`. В этом случае каждое из двух расстояний, q и h, будут "работать" независимо друг от друга. Альтернативным решением будет использование треугольной границы, о которой мы рассказывали в главе по МГК. В этом случае нужно использовать `lim.type = "ddmoments"` и это значение для этого аргумента по умолчанию.

В примере ниже будем использовать уже хорошо знакомый набор данных с ирисами. Начнем с того, что загрузим эти данные и разобьем их на калибровочный и тестовый наборы.

```
data(iris)

# сгенерируем индексы так чтобы каждая вторая стока оказалась в тестовом наборе
idx <- seq(1, nrow(iris), by = 2)

# зададим матрицу с измерениями и вектор с названиями классов для калибровки
Xc <- iris[idx, 1:4]
cc <- iris[idx, 5]

# и тоже самое для теста
Xt <- iris[-idx, 1:4]
ct <- iris[-idx, 5]
```

После этого в каждом из двух наборов у нас содержится 75 измерений — по 25 на каждый класс. Выделим теперь измерения для каждого класса в отдельный фрейм, но только для калибровочного набора:

```
X.set <- Xc[1:25, ]
X.ver <- Xc[26:50, ]
X.vir <- Xc[51:75, ]
```

Теперь построим SIMCA модель для класса *versicolor* и покажем информацию о модели. В коде ниже мы явно задаем тип границы и уровень значимости (`lim.type = "ddmoments"` и `alpha = 0.05`), однако, в этом случае их можно было опустить, так как мы используем значения по умолчанию. Мы также задаем максимальное число компонент равное трем.

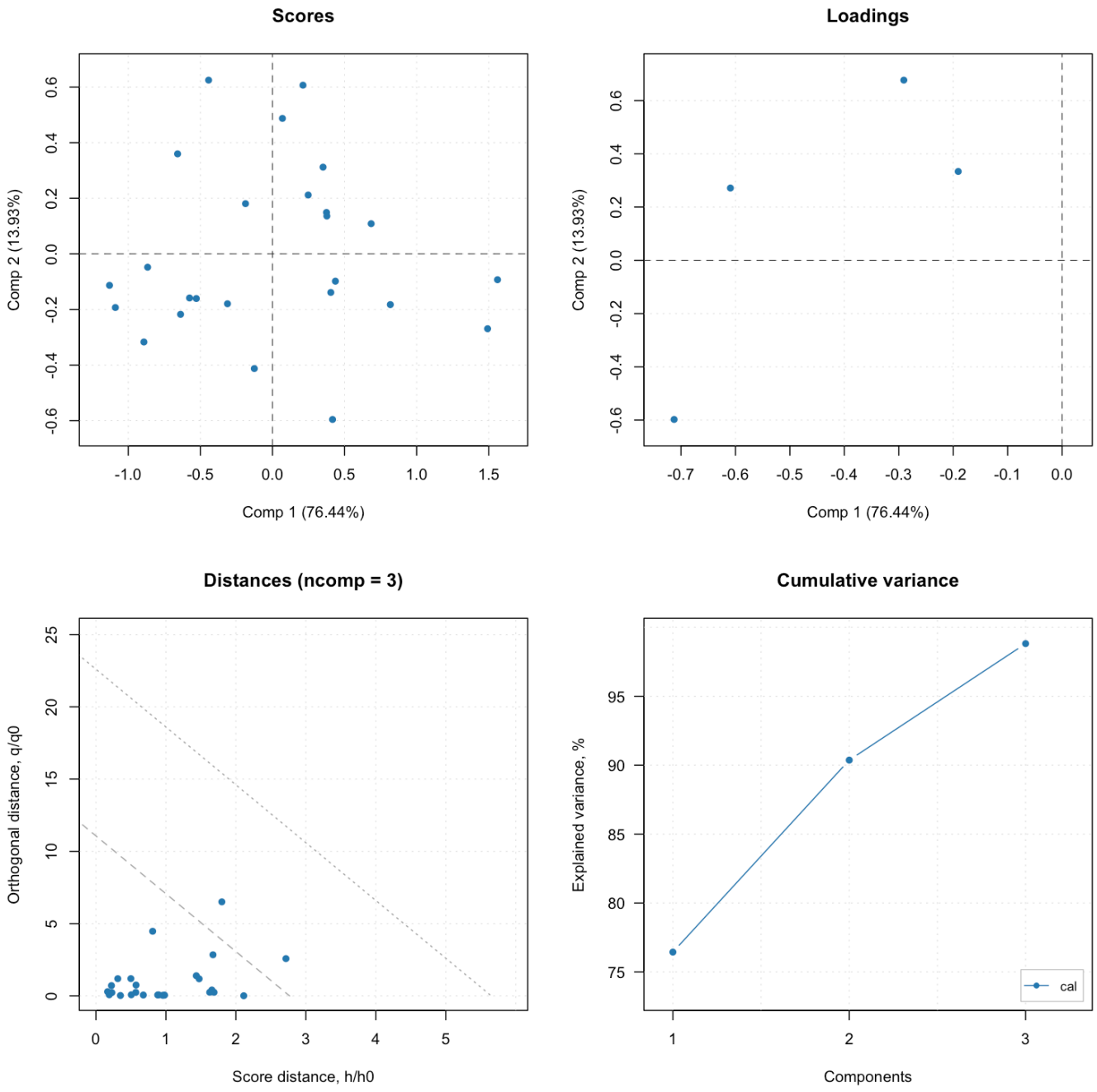
```
library(mdatools)
m <- simca(X.ver, "versicolor", ncomp = 3, lim.type = "ddmoments", alpha = 0.05)
summary(m)
```

```
SIMCA model for class 'versicolor' summary


Number of components: 3
Type of limits: ddmoments
Alpha: 0.05
Gamma: 0.01

    Expvar Cumexpvar TP FP TN FN Spec. Sens. Accuracy
Cal   8.45     98.82 23  0  0  2    NA  0.92     0.92
```

Как можно видеть из таблицы, число правильно распознанных членов класса составило 23, в то время как 2 были ложно отвергнуты моделью, что в результате дало чувствительность (*Sens*) равную 92%. Отметим, что в качестве селективности (*Spec*) указано значение *NA*, так как мы не можем рассчитать эту характеристику, не имея образцов из других классов.

Посмотрим теперь на графический обзор модели:

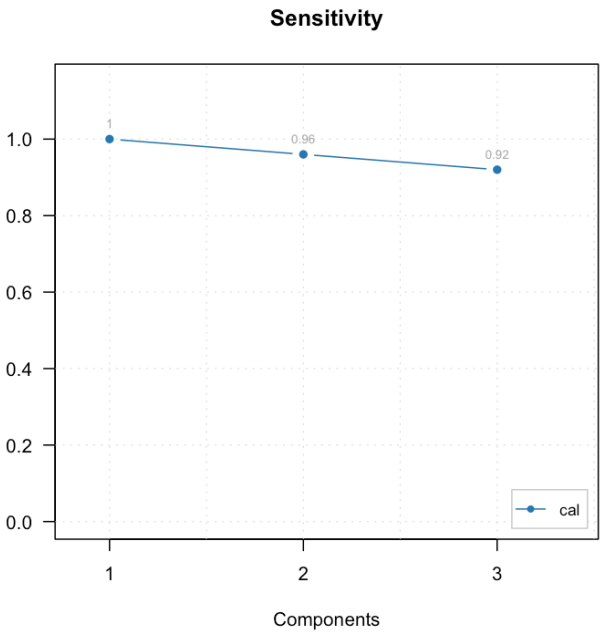
```
plot(m)
```

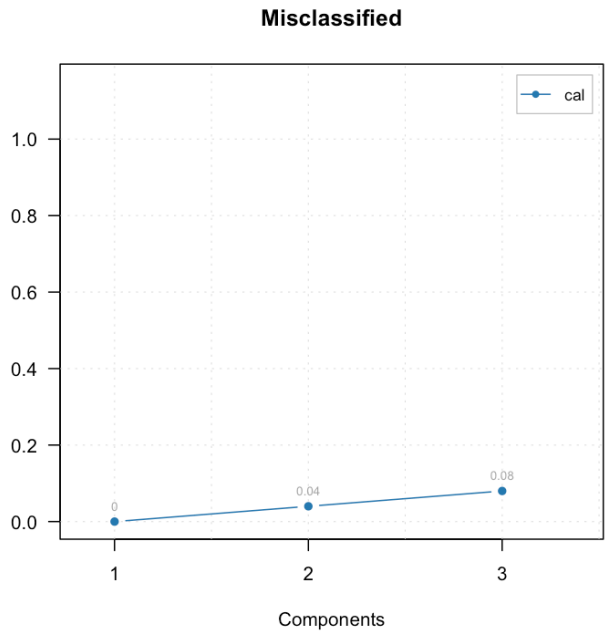


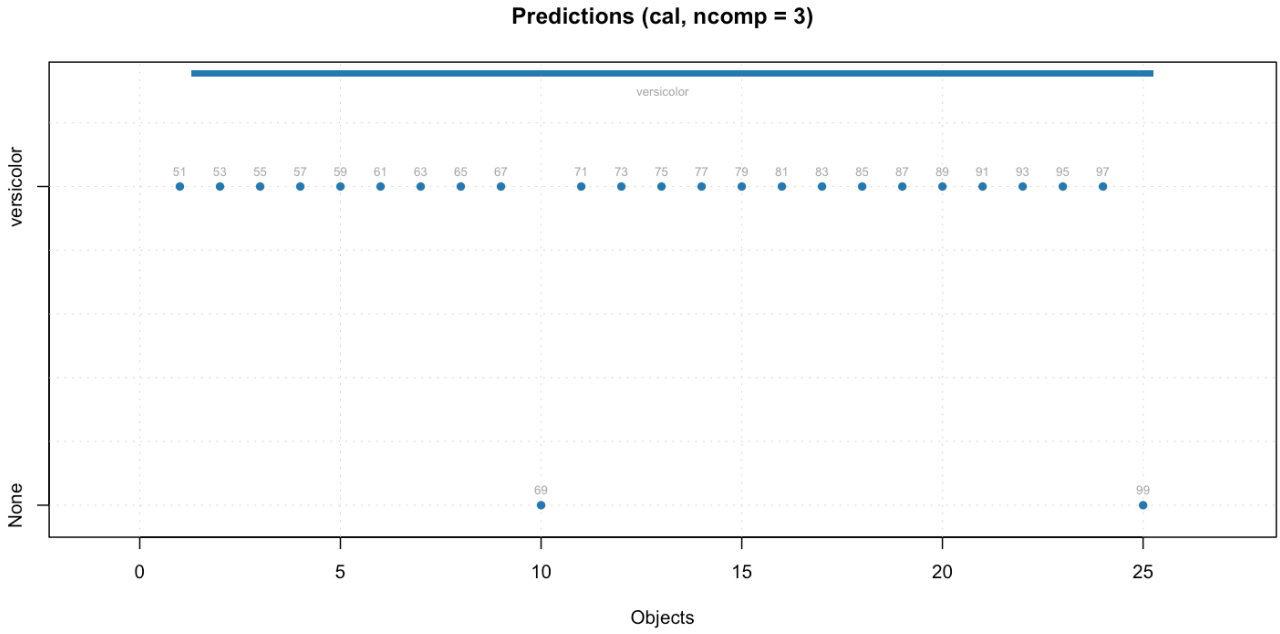
Как можно видеть, он очень похож на такой же график для МГК модели

В дополнение ко всем графикам, доступным для МГК моделей, в этом случае мы также можем изучить результаты классификации. Пример ниже показывает три графика: график чувствительности и числа неправильно классифицированных образцов в зависимости от числа компонент, а также график с результатами классификации:

```
layout(matrix(c(1, 3, 2, 3), ncol = 2))
plotSensitivity(m, show.labels = TRUE)
plotMisclassified(m, show.labels = TRUE)
plotPredictions(m, show.labels = TRUE)
```







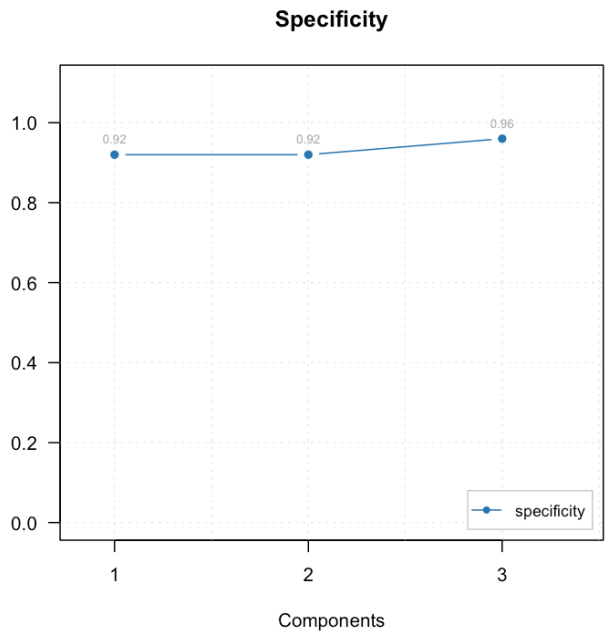
Очевидно, что в этом случае оптимальное число компонент равно двум. С одной компонентой чувствительность получается идеальная, но в нашем случае мы ожидаем ее значение равное 0.95, так что использование двух компонент выглядит более разумным в этом случае.

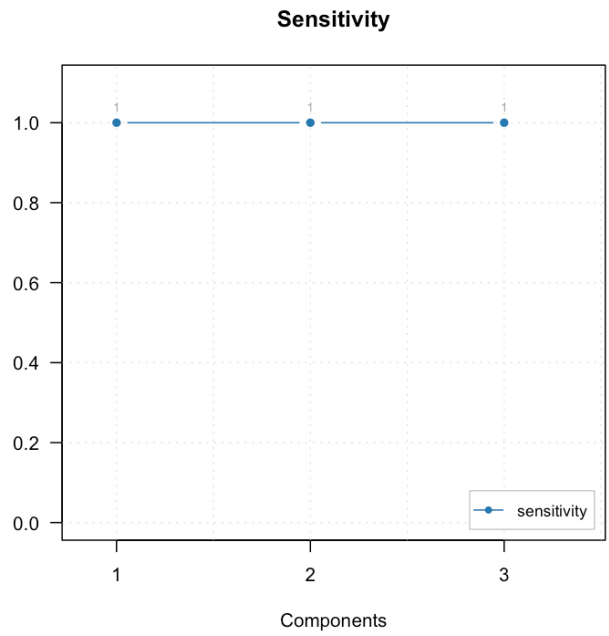
Решение об оптимальном числе компонент трудно сделать без проверки модели, к счастью у нас есть тестовый набор, которым мы сейчас воспользуемся:

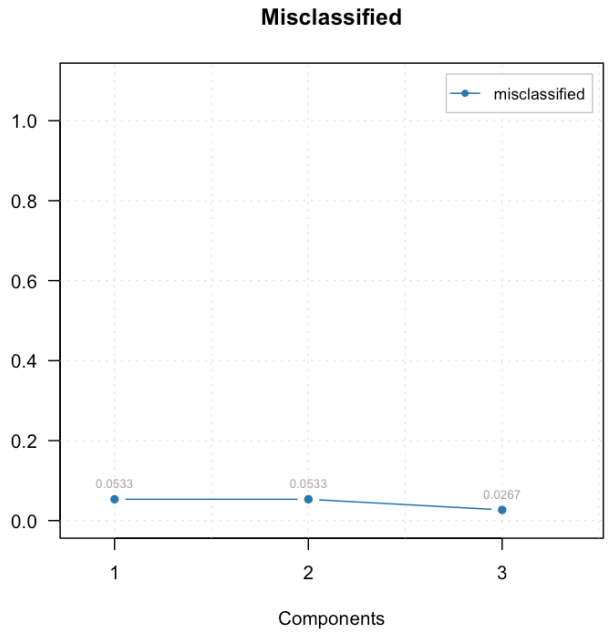
```
Summary for SIMCA one-class classification result

Class name: versicolor
Number of selected components: 3

       Expvar Cumexpvar TP FP TN FN Spec. Sens. Accuracy
Comp 1  64.27     64.27 25  4 46  0  0.92     1    0.947
Comp 2   1.67     65.95 25  4 46  0  0.92     1    0.947
Comp 3  32.45     98.40 25  2 48  0  0.96     1    0.973
```

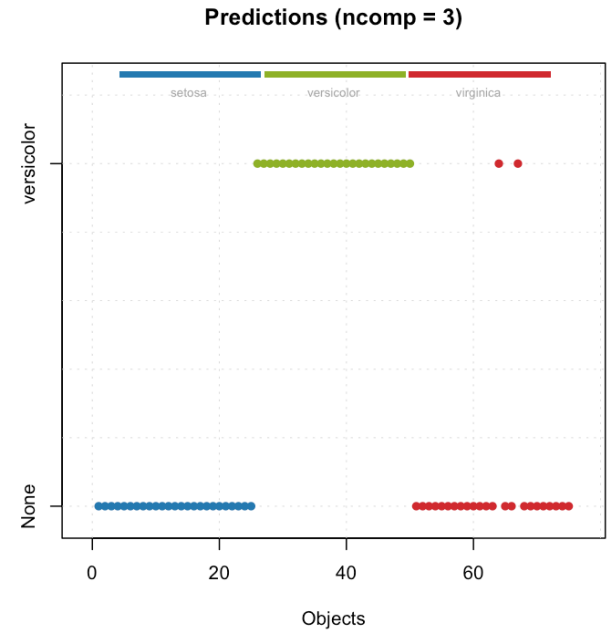
Результаты применения модели для предсказания класса образцов из тестового набора демонстрируют наилучшие характеристики для трех компонент. Эти результаты можно также показать на графиках, в этом случае мы также можем показать график для специфичности:

```
par(mfrow = c(2, 2))
plotSpecificity(res, show.labels = TRUE)
plotSensitivity(res, show.labels = TRUE)
plotMisclassified(res, show.labels = TRUE)
plotPredictions(res)
```









Мы также можем показать результат в виде матрицы:

```
           versicolor None
versicolor         25    0
setosa              0   25
virginica           2   23
```

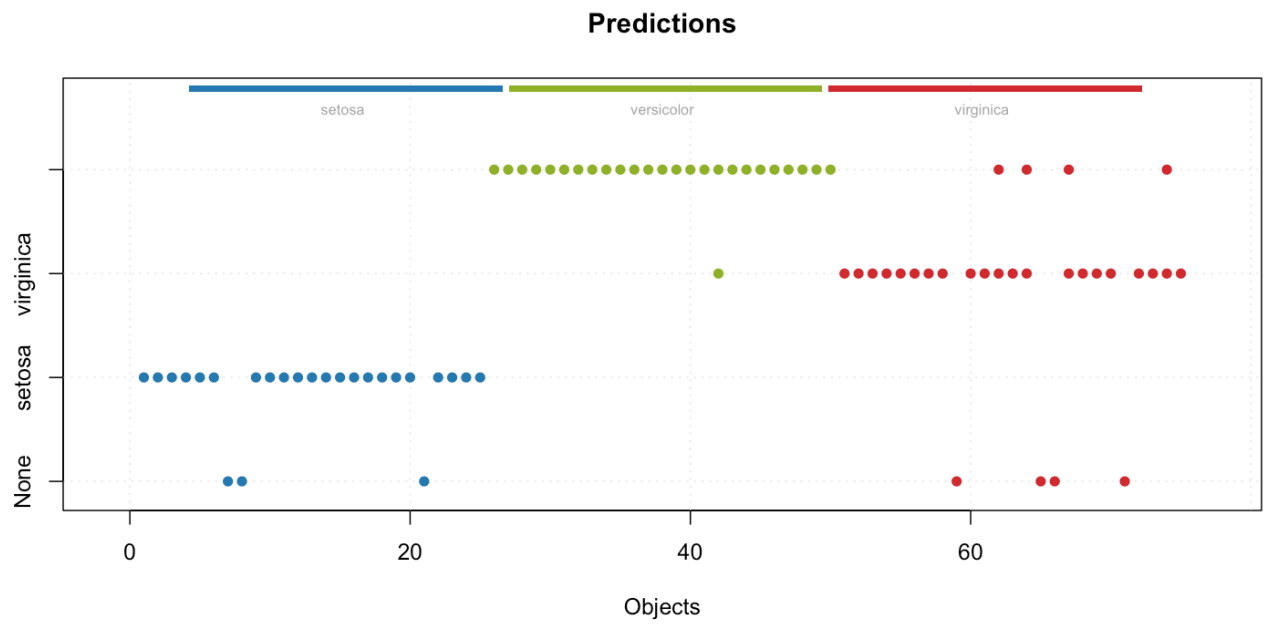
Существует также возможность объединения нескольких SIMCA моделей вместе, как это показано на примере ниже. Для начала построим индивидуальные модели для каждого из трех классов.

Теперь объеденим их вместе в специальный объект `simcam`:

```
SIMCA multiple classes classification (class simcam)

Number of classes: 3
Info:

Summary for calibration results
           Ncomp TP FP TN FN Spec. Sens. Accuracy
setosa         1 24  0 50  1  1.00  0.96    0.987
virginica      2 22  3 47  3  0.94  0.88    0.920
versicolor     1 25  3 47  0  0.94  1.00    0.960
```

Как можно видеть, теперь таблица показывает информацию о каждой модели. Применим эту объединенную модель для предсказания классов образцов из тестового набора и покажем график с результатами классификации:

```
res <- predict(mm, Xt, ct)

par(mfrow = c(1, 1))
plotPredictions(res)
```



Так же как и в случае с `summary()`, на графике видны результаты классификации для каждой модели (строки с точками). Однако, это все еще одноклассовая классификация, модели "работают" независимо друг от друга, поэтому один образец может быть распознан как "свой" одновременно несколькими моделями.

Мы также можем показать результат классификации "изнутри"

```
head(res$c.pred[20:30, 1, ])
```

```
   setosa virginica versicolor
40      1        -1         -1
42     -1        -1         -1
44      1        -1         -1
46      1        -1         -1
48      1        -1         -1
50      1        -1         -1
```

Метод `getConfusionMatrix()` также работает в этом случае.

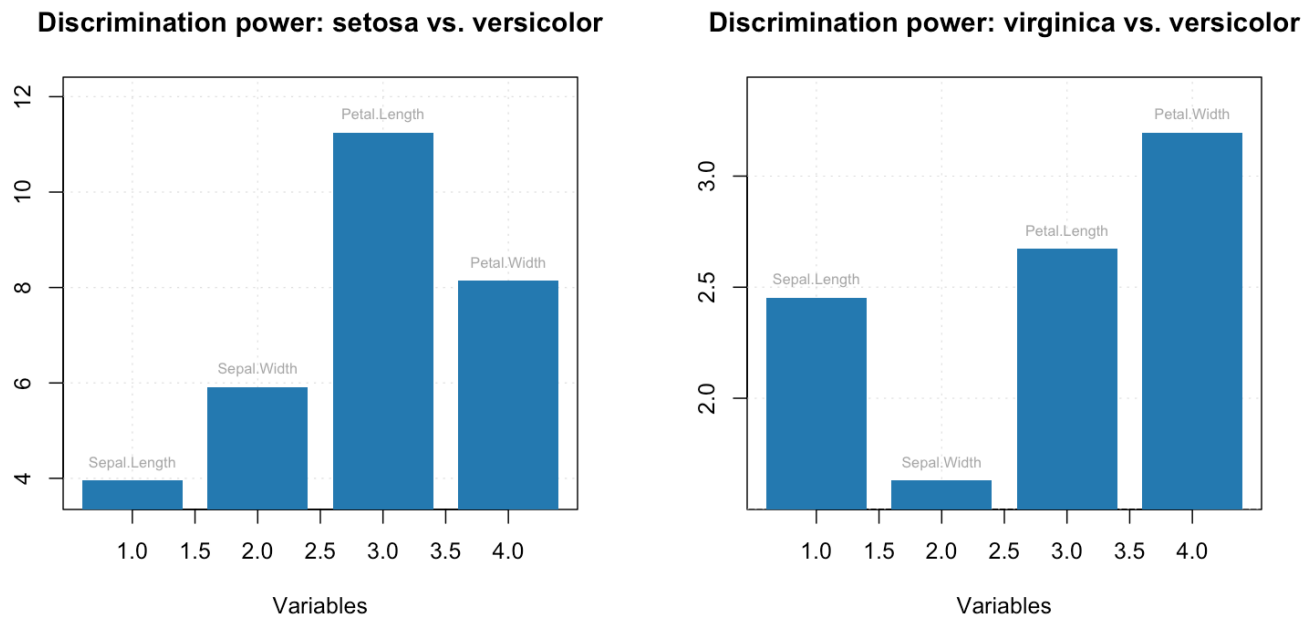
```
show(getConfusionMatrix(res))
```

```
          setosa virginica versicolor None
```

```
setosa         22         0          0    3
virginica       0        21          4    4
versicolor      0         1         25    0
```

Для объединенных SIMCA моделей доступны два дополнительных графика. Во-первых, график дискриминационной способности переменных, который для каждой пары классов показывает, какие именно переменные делают их непохожими друг на друга.

```
par(mfrow = c(1, 2))
plotDiscriminationPower(mm, c(1, 3), show.labels = TRUE)
plotDiscriminationPower(mm, c(2, 3), show.labels = TRUE)
```



И, наконец, график Кумана, рассмотренный нами ранее в теоретической части.

```
par(mfrow = c(1, 2))
plotCooman(mm, c(1, 3), show.labels = TRUE)
plotCooman(mm, c(2, 3), show.labels = TRUE)
```

Здесь вертикальные линии показывают границы принятия решений для расстояния q.

5.9 Дискриминантный анализ на основе ПЛС (PLS-DA)

В главе 4 мы познакомились с методом ПЛС, который позволяет строить регрессионные модели для количественного прогнозирования различных параметров в образцах. Этот же метод без каких-либо модификаций самого алгоритма ПЛС часто применяется и для решения задач классификация. Такой метод называется дискриминантным анализом на основе проекций на латентные структуры (ПЛС-ДА, англ. *PLS-DA*, *partial least squares – discriminant analysis*). В PLS-DA в качестве значений независимой прогнозируемой переменной y используются закодированные нулями и единицами классовые принадлежности образцов, аналогично рассмотренному выше методу логистической регрессии.

Как и в других рассмотренных нами случаях, регрессионная модель строится на основе образцов обучающей выборки с известной классовой принадлежностью. Спрогнозированные по этой модели значения y для новых образцов используются для определения принадлежности этих образцов к тому, или иному классу. Рис. 5.16 показывает пример PLS-DA модели в координатах «введено – найдено». На оси абсцисс отложена известная кодировка обучающих образцов (нули для первого класса и единицы для второго), а на оси ординат – спрогнозированные значения для этих же образцов. Обратите внимание, что спрогнозированные значения y могут отличаться от нуля и единицы, и, в принципе, принимать практически любые значения, однако, для хорошей модели PLS-DA их отличие от нуля и единицы будет незначительным.

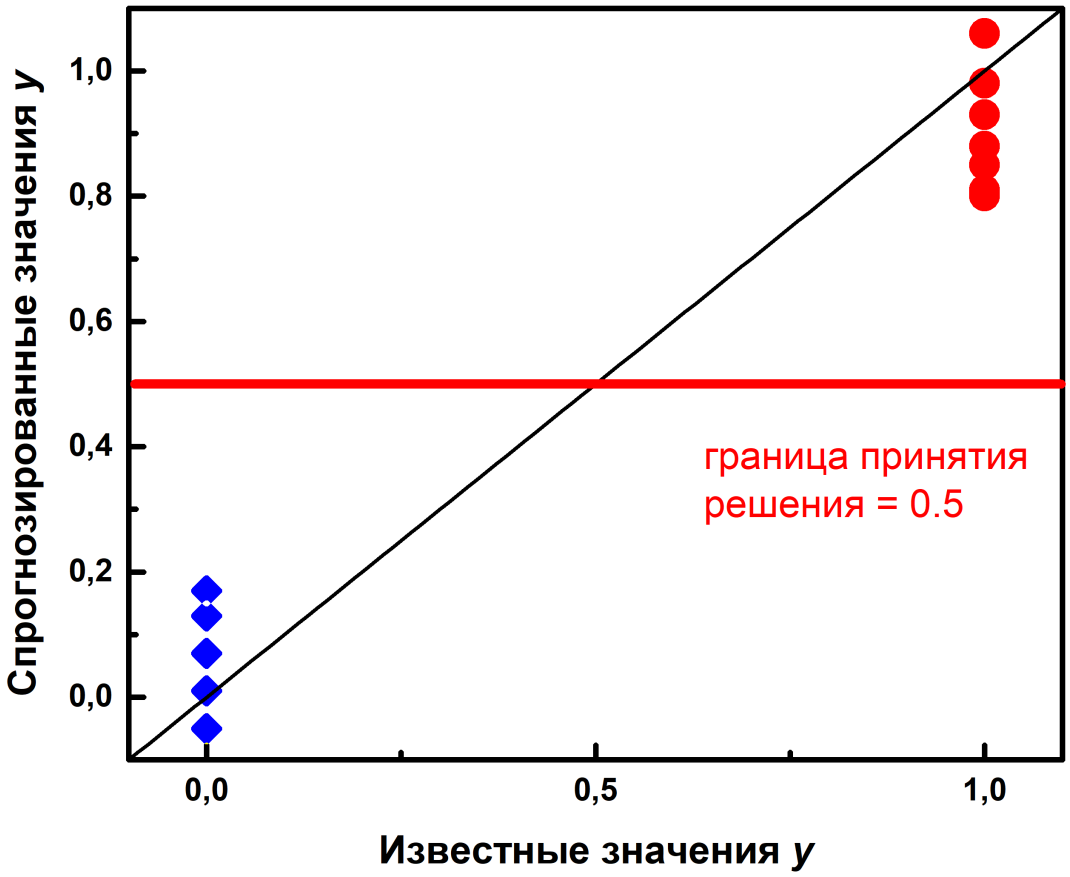


Рис. 5.16. График «введено – найдено» для PLS-DA модели.

В случае неизвестных образцов решение о классовой принадлежности принимают сравнивая спрогнозированное значение y с границей принятия решения (как правило, 0.5). Если $y < 0.5$, то образец считается принадлежащим к первому классу, в противном случае — ко второму.

PLS-DA, в отличие, например, от SIMCA, является строго двухклассовым методом. Если необходимо работать с числом классов больше двух, то PLS-DA используют по схеме «один против всех». При этом образцы одного из классов кодируют нулями, а все образцы любых других классов единицами. В случае трех классов необходимо будет построить три PLS-DA модели – отдельная модель для каждого класса.

Кроме этого, следует отметить, что PLS-DA — это именно метод дискриминации, то есть каждый образец обязательно будет принадлежать какому-либо классу, и при этом только одному. Следствием этого является требование: в обучающей выборке репрезентативно должны быть представлены оба класса. PLS-DA не используют для решения задач одноклассовой классификации, поскольку в этом случае нельзя собрать репрезентативную выборку всех образцов, не являющихся образцами моделируемого класса (их количество неизмеримо велико).

PLS-DA является параметрическим методом. В зависимости от числа скрытых переменных в модели (англ. *LV*, *latent variable*) и выбранного значения границы принятия решений будет меняться точность классификации. Выбор этих параметров осуществляется таким образом, чтобы уменьшить количество неверно классифицированных образцов в тестовом наборе.

Интересной особенностью PLS-DA является простая и удобная идентификация переменных, вносящих наибольший вклад в различия между классами. Это широко используется, например, для исследований по метаболомике при выявлении биомаркеров различных заболеваний. При постановке таких экспериментов отбирают образцы, например, биологических жидкостей, от двух групп людей — с исследуемым заболеванием, и без него. Проводят анализ этих жидкостей с целью наиболее полной расшифровки химического состава. Как правило, это делают с помощью высокоинформативных современных аналитических методов на основе хроматографии. В дальнейшем, после проведения необходимой предварительной обработки (выравнивание времен удерживания, коррекция базовой линии, нормализация и т.д.), используют метод PLS-DA для построения модели, которая на основании хроматограммы образца биологической жидкости способна спрогнозировать принадлежность человека к той, или иной группе (с заболеванием, либо без). Изучение графика регрессионных коэффициентов такой модели позволяет делать заключение о химических соединениях, вносящих наибольший вклад в различия между двумя классами.

На рис. 5.17 в качестве иллюстрации приведен гипотетический пример такого отбора переменных. Черной линией показана хроматограмма одного из образцов, а синей линией — регрессионные коэффициенты, полученные при PLS-DA моделировании. Видно, что наиболее высокие значения регрессионных коэффициентов получены для веществ, имеющих времена удерживания 150, 265 и 450 секунд. Регрессионные коэффициенты для остальных веществ незначительны. Отсюда можно

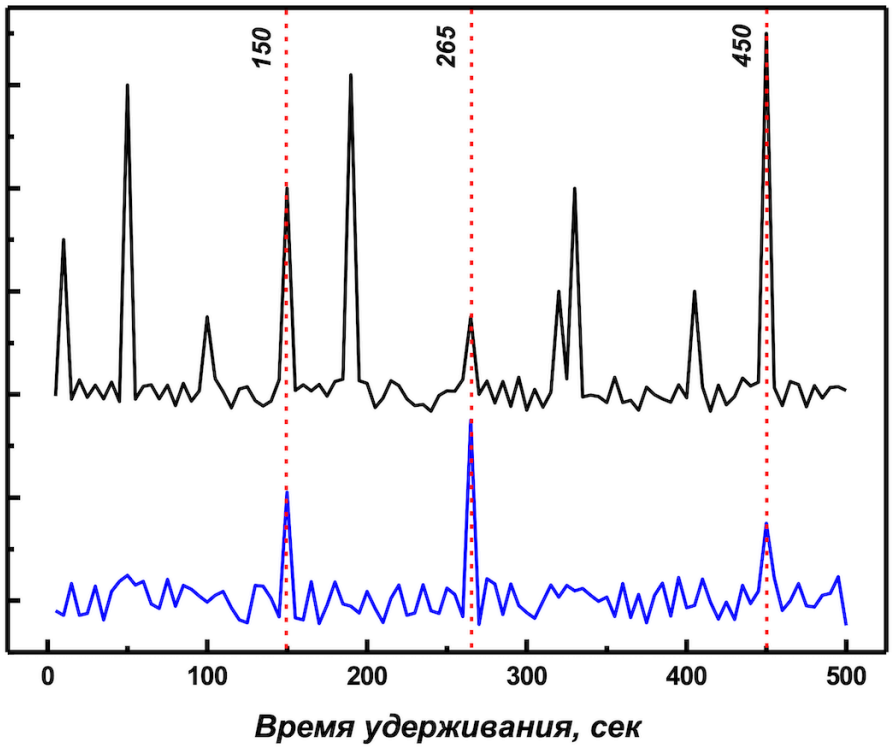


Рис. 5.17. Выявление значимых переменных в PLS-DA модели.

сделать вывод о том, что именно эти три соединения являются потенциальными маркерами изучаемого заболевания.

Метод PLS-DA нужно аккуратно использовать при большом числе переменных. Это связано с особенностями алгоритма, который «рад услужить» и ищет именно ту дисперсию в данных, которая связана с разницей между двумя классами. Проиллюстрируем этот тезис на следующем примере. Сгенерируем 100 наборов из 200 случайных чисел и разобьем их случайным образом на два класса, одному классу поставим в соответствие нули, другому – единицы. На рис. 5.18 приведен пример полученных таким образом данных.

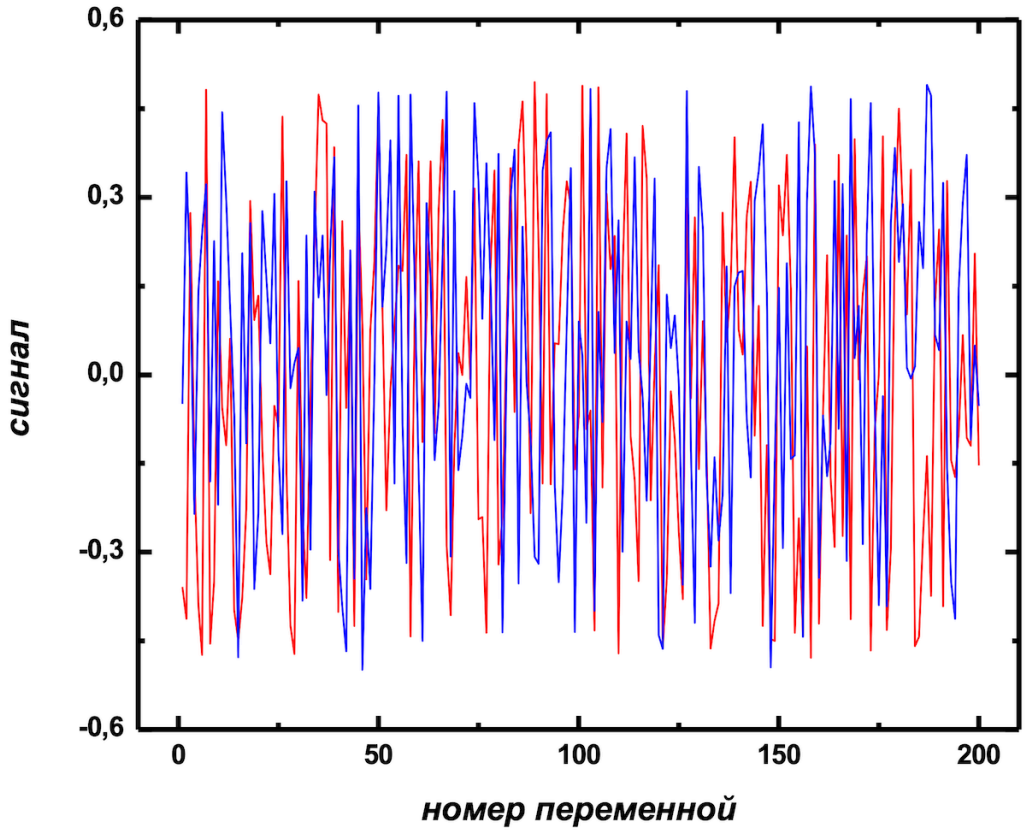


Рис. 5.18. Пример сгенерированных случайным образом данных для двух классов.

Даже на основе таких данных легко получить PLS-DA модель, которая, на первый взгляд, имеет хорошее качество (классы уверенно разделяются между собой, Рис. 5.19а). Проверка такой модели независимым тестовым набором (Рис. 5.19б) показывает ее полную непригодность для практического использования. Именно поэтому к валидации моделей ПЛС-ДА нужно подходить с особой тщательностью.

В методах классификации, где можно менять границу принятия решений (таких, как, например, PLS-DA, логистическая регрессия) для оценки качества модели часто используется ROC-кривая (англ. *receiver operat-*

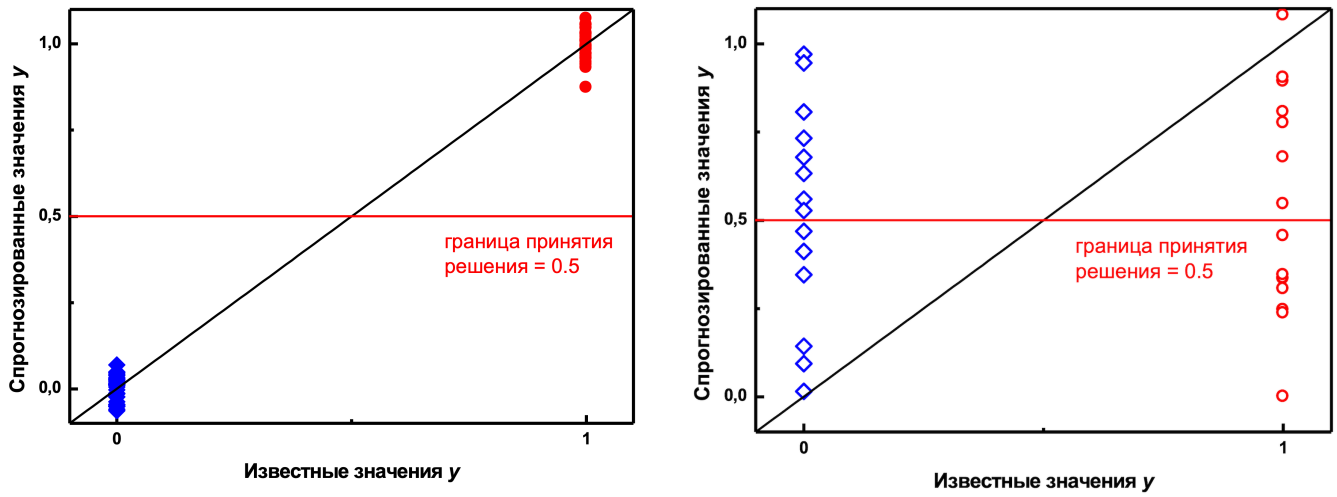


Рис. 5.19. Графики «введено – найдено» а) PLS-DA модели на обучающей выборке; б) проверки модели тестовым набором.

ing characteristics curve, рабочая характеристика приемника). Для построения этой кривой рассчитываются значения чувствительности и специфичности классификатора при разных границах принятия решения. По этим значениям строится ROC-кривая в координатах (1-специфичность) – чувствительность (рис. 5.20). Интегральным показателем качества модели служит величина AUC (area under the curve), которая представляет собой площадь под ROC-кривой соответствующей модели.

На рис. 5.20 приведены ROC-кривые для трех моделей. Пунктирной линией показан классификатор, который распределяет образцы между классами случайным образом, при таком варианте значение AUC = 0.5. Это означает, что модель не пригодна для классификации. Любая модель, график ROC для которой расположен выше пунктирной линии (AUC > 0.5), будет лучше, чем случайное гадание. Так, на графике синим и красным цветами показаны ROC-кривые для двух рабочих моделей. Очевидно, наилучшей является модель с AUC = 0.9. Наряду с матрицей ошибок ROC-кривая дает удобный инструмент для оценки качества классификационных моделей и сравнения их между собой.

Примеры реализации в R

Традиционно для примеров в этом разделе будем использовать реализацию метода из пакета *mdatools*. Однако, так же как SIMCA — это на 99 процентов МГК, так и PLS-DA — это на 99 процентов ПЛС-регрессия и вы можете реализовать этот метод самостоятельно.

Калибровка PLS-DA в *mdatools* очень похожа на калибровку PLS модели. Однако, вместо вектора с откликами вам нужно задать вектор с информацией о классах, которые требуется дискриминировать. Это можно сделать несколькими способами.

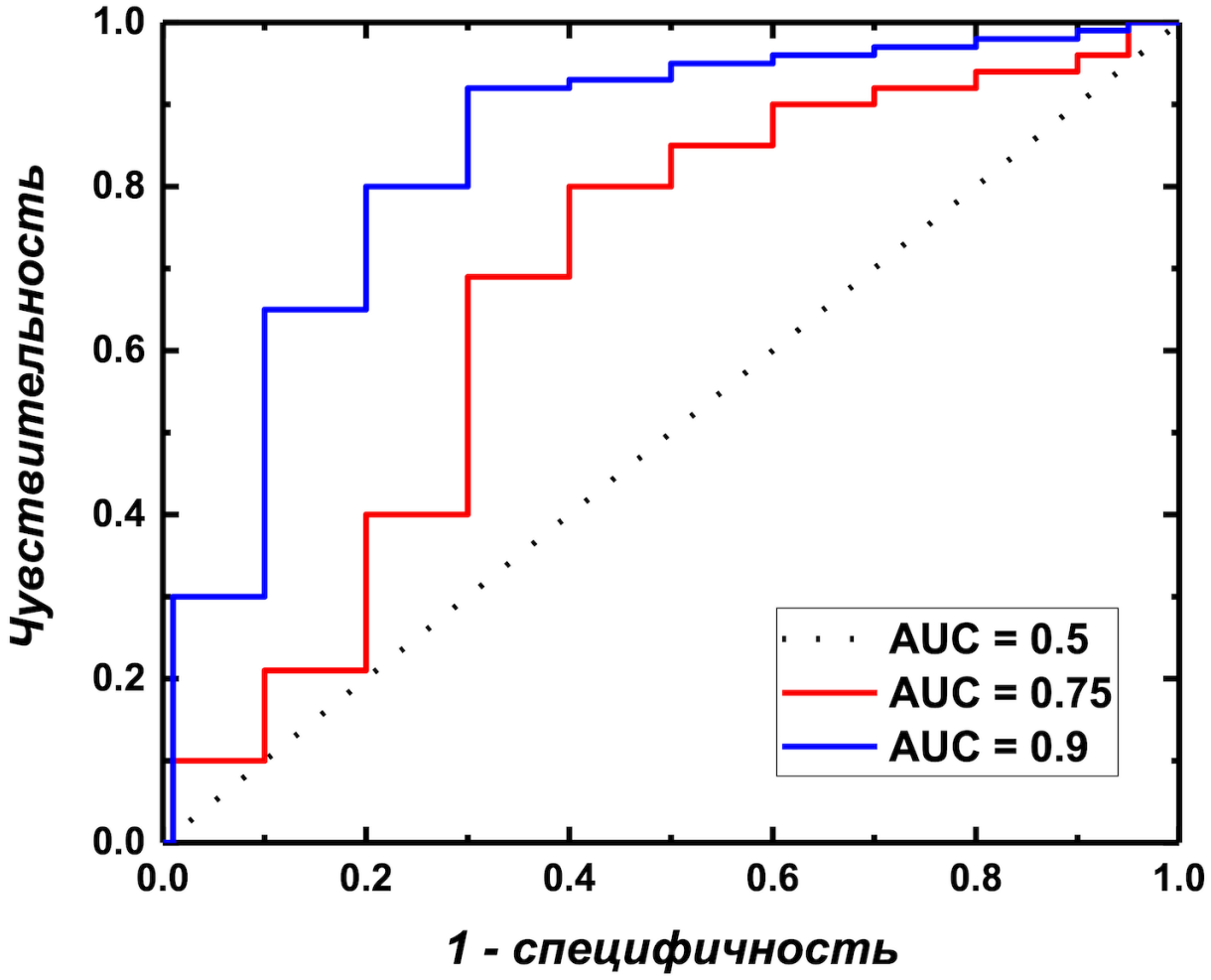


Рис. 5.20. ROC-кривые для разных моделей.

Если у вас несколько классов и вы хотите построить дискриминационную модель для каждого из них (т.е. по сути это будет три модели: члены класса против чужаков, для каждого из классов), то просто задайте вектор с метками классов в виде фактора. В этом случае метки будут автоматически использоваться в качестве имен классов. Имена меток очень важно задавать правильно, так, чтобы они были одинаковы и в калибровочном, и в тестовом наборе, если вы используете проверку тестовым набором.

Если же нужно сделать модель только для одной пары классов, то тогда этот вектор можно задать в виде логический значений (`TRUE` или `FALSE`), но в этом случае при построении модели нужно также задать имя класса. Этот случай мы рассмотрим чуть позже.

Давайте подготовим данные для моделирования на основе все того же набора с ирисами.

```
data(iris)

# выбираем первые 25 образцов из каждого класса для калибровочного набора
# и следующие 25 образцов поместим в тестовый набор

# для начала зададим индексы строк для каждого набора
cal.ind <- c(1:25, 51:75, 101:125)
val.ind <- c(26:50, 76:100, 126:150)

# теперь зададим фреймы с предикторами
Xc <- iris[cal.ind, 1:4]
Xv <- iris[val.ind, 1:4]

# и векторы с метками классов
cc.all <- factor(iris[cal.ind, 5])
cv.all <- factor(iris[val.ind, 5])
```

Заметим, что векторы с метками классов мы сделали факторами, это можно проверить:

```
show(cc.all)
```

```
 [1] setosa     setosa     setosa     setosa     setosa     setosa
 [7] setosa     setosa     setosa     setosa     setosa     setosa
[13] setosa     setosa     setosa     setosa     setosa     setosa
[19] setosa     setosa     setosa     setosa     setosa     setosa
[25] setosa     versicolor versicolor versicolor versicolor versicolor
[31] versicolor versicolor versicolor versicolor versicolor versicolor
```

```
[37] versicolor versicolor versicolor versicolor versicolor versicolor
[43] versicolor versicolor versicolor versicolor versicolor versicolor
[49] versicolor versicolor virginica  virginica  virginica  virginica
[55] virginica  virginica  virginica  virginica  virginica  virginica
[61] virginica  virginica  virginica  virginica  virginica  virginica
[67] virginica  virginica  virginica  virginica  virginica  virginica
[73] virginica  virginica  virginica
Levels: setosa versicolor virginica
```

Далее построим модель для дискриминации каждого класса. Для проверки будем использовать кросс-валидацию с систематическим разбиением на 5 сегментов:

```
PLS-DA model (class plsda) summary
-----------------------------------
Info:
Number of selected components: 1
Cross-validation: venetian blinds with 5 segments

Class #1 (setosa)
    X cumexpvar Y cumexpvar TP FP TN FN Spec. Sens. Accuracy
Cal       91.97       46.36 25  0 50  0     1     1        1
Cv           NA          NA 25  0 50  0     1     1        1

Class #2 (versicolor)
    X cumexpvar Y cumexpvar TP FP TN FN Spec. Sens. Accuracy
Cal       91.97       46.36  0  0 50 25  1.00     0    0.667
Cv           NA          NA  0  2 48 25  0.96     0    0.640

Class #3 (virginica)
    X cumexpvar Y cumexpvar TP FP TN FN Spec. Sens. Accuracy
Cal       91.97       46.36 24  5 45  1  0.90  0.96    0.920
Cv           NA          NA 24  4 46  1  0.92  0.96    0.933
```

Как можно заметить, информация о модели включает в себя три разных блока — по одному для каждого класса. Характеристики модели похожи на те, что мы видели на примерах для SIMCA, и, помимо объясненной дисперсии, содержат информацию о четырех группах образцов (true positives, false positives,

true negatives, false negatives), чувствительности, специфичности и точности классификации. Причем, как и в случае ПЛС-регрессии, эта информация дана и для тренировочного набора и для кросс-валидации.

И точно также вы можете отдельно показать информацию только для конкретных результатов — в этом случае вы также увидите, как качество модели зависит от числа компонент. Это можно сделать как для всех классов, так и для одного конкретного класса, просто укажите его номер в виде параметра с именем `nc`, как в примере ниже:

```
# показать информацию по результатам перекрестной проверки для
# третьего класса (virginica)
summary(m.all$cvres, nc = 3)
```

```
PLS-DA results (class plsdares) summary:
Number of selected components: 1

Class #3 (virginica):
       X expvar X cumexpvar Y expvar Y cumexpvar TP FP TN FN Spec. Sens.
Comp 1       NA          NA       NA          NA 24  4 46  1  0.92  0.96
Comp 2       NA          NA       NA          NA 20  6 44  5  0.88  0.80
Comp 3       NA          NA       NA          NA 24  4 46  1  0.92  0.96
       Accuracy
Comp 1    0.933
Comp 2    0.853
Comp 3    0.933
```

По причинам, описанным нами ранее, для кросс-валидации нет возможности точно посчитать объясненную дисперсию, поэтому в соответствующих столбцах стоят значения `NA`.

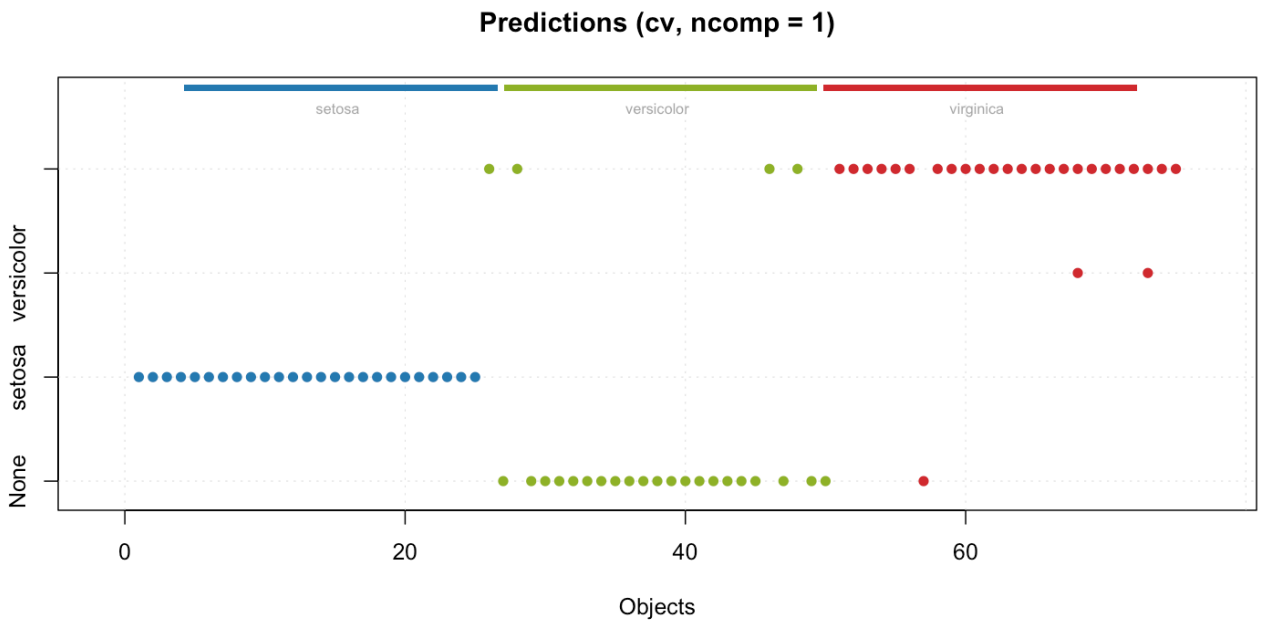
Мы также можем вывести матрицу предсказаний:

```
show(getConfusionMatrix(m.all$cvres))
```

```
           setosa versicolor virginica None
setosa         25          0         0    0
versicolor      0          0         4   21
virginica       0          2        24    1
```

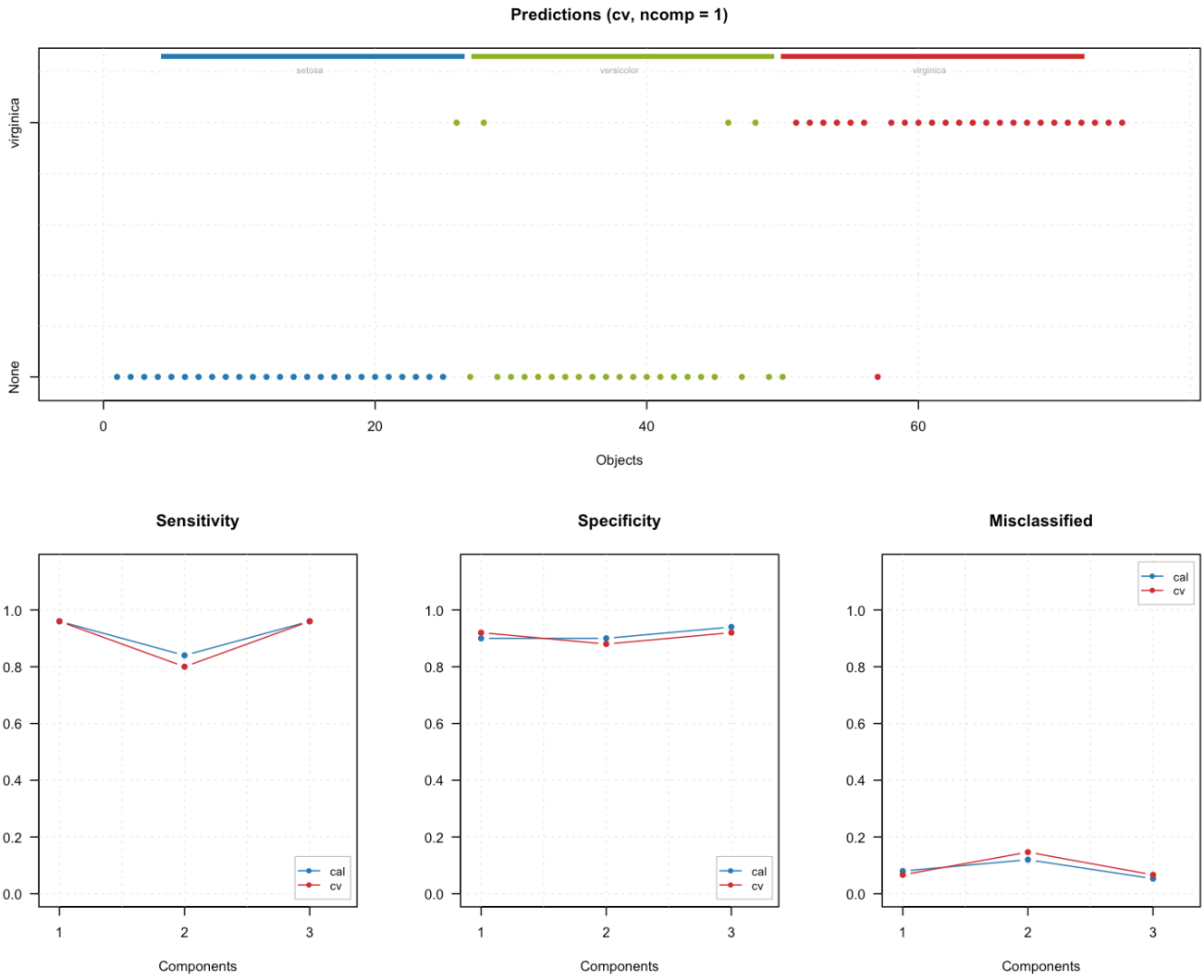
Все графики, показывающие визуально характеристики модели и результатов, а также результаты классификации, которые мы использовали для SIMCA, доступны также и для PLS-DA моделей:

```
plotPredictions(m.all)
```



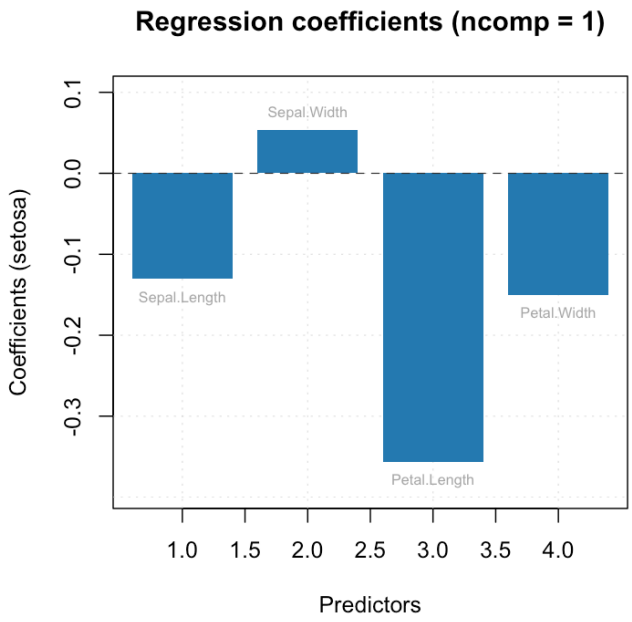
Однако, поскольку классов много, для некоторых графиков нужно указать номер, чтобы получить результат. Вот, например, набор всех графиков для третьего класса (*virginica*)

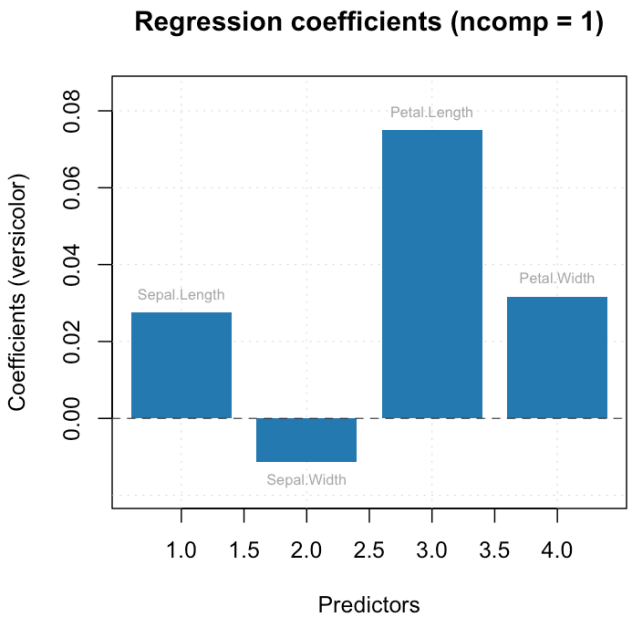
```
layout(matrix(c(1, 2, 1, 3, 1, 4), ncol = 3))
plotPredictions(m.all, nc = 3)
plotSensitivity(m.all, nc = 3)
plotSpecificity(m.all, nc = 3)
plotMisclassified(m.all, nc = 3)
```



Кроме этого, как мы уже отметили, вы можете пользоваться всем инструментарием, который предоставляет ПЛС-регрессия. Например, изучить график регрессионных коэффициентов, которые помогут определить, какие переменные вносят наибольший вклад в дискриминацию между классами. Надо лишь помнить, что в этом случае нужно использовать синтаксис ПЛС-регрессии, например, чтобы показать регрессионные коэфициенты для первого и второго классов, нужно задать не номер класса (аргумент `nc`), а номер отклика (аргумент `ny`), как это показано в примере ниже:

```
par(mfrow = c(1, 2))
plotRegcoeffs(m.all, ny = 1, show.labels = TRUE)
plotRegcoeffs(m.all, ny = 2, show.labels = TRUE)
```





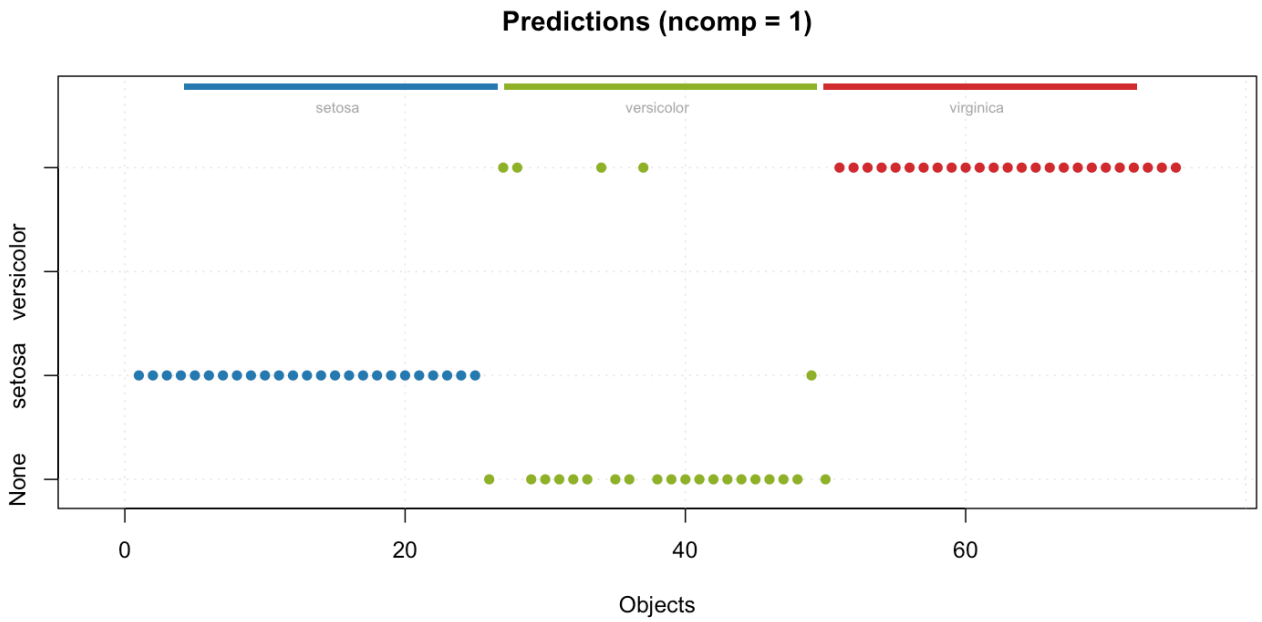
Классификация новых образцов делается также, как и в ПЛС-регрессии, только вместо вектора откликов нужно задавать вектор с метками классов (если они известны, например, для тестового набора):

```
res = predict(m.all, Xv, cv.all)
summary(res)
```

```
PLS-DA results (class plsdares) summary:
Number of selected components: 1

Class #1 (setosa):
       X expvar X cumexpvar Y expvar Y cumexpvar TP FP TN FN Spec. Sens.
Comp 1   92.924      92.924   42.703      42.703 25  1 49  0  0.98     1
Comp 2    4.560      97.484   11.216      53.920 25  0 50  0  1.00     1
Comp 3    1.790      99.274    1.717      55.637 25  0 50  0  1.00     1
       Accuracy
Comp 1    0.987
Comp 2    1.000
Comp 3    1.000
```

```
Class #2 (versicolor):
       X expvar X cumexpvar Y expvar Y cumexpvar TP FP TN FN Spec. Sens.
Comp 1   92.924      92.924   42.703      42.703  0  0 50 25  1.00   0.0
Comp 2    4.560      97.484   11.216      53.920 10  4 46 15  0.92   0.4
Comp 3    1.790      99.274    1.717      55.637 10  6 44 15  0.88   0.4
       Accuracy
Comp 1    0.667
Comp 2    0.747
Comp 3    0.720


Class #3 (virginica):
       X expvar X cumexpvar Y expvar Y cumexpvar TP FP TN FN Spec. Sens.
Comp 1   92.924      92.924   42.703      42.703 25  4 46  0  0.92  1.00
Comp 2    4.560      97.484   11.216      53.920 25  4 46  0  0.92  1.00
Comp 3    1.790      99.274    1.717      55.637 24  4 46  1  0.92  0.96
       Accuracy
Comp 1    0.947
Comp 2    0.947
Comp 3    0.933
```

```
par(mfrow = c(1, 1))
plotPredictions(res)
```



В конце приведем пример, где информация о классе задается не вектором с метками, а вектором с логическими значениями (`TRUE` для объектов, принадлежащих классу, и `FALSE` для объектов, не принадлежащих ему).

Построим модель, которая будет отличать один класс, *virginica*, от всех остальных. Т.е. нам нужно и для тренировочного набора, и для тестового подготовить информацию о классах в виде вектора с логическими значениями. Это можно сделать вот так:

```
cc.vir <- cc.all == "virginica"
cv.vir <- cv.all == "virginica"
```

В этом случае при построении модели нужно задать имя класса:

```
m.vir <- plsda(Xc, cc.vir, classname = "virginica", cv = list("ven", 5))
summary(m.vir)
```

```
PLS-DA model (class plsda) summary
-----------------------------------
Info:
Number of selected components: 3
Cross-validation: venetian blinds with 5 segments
```

```
Class #1 (virginica)
    X cumexpvar Y cumexpvar TP FP TN FN Spec. Sens. Accuracy
Cal       98.53       61.31 24  3 47  1  0.94  0.96    0.947
Cv           NA          NA 24  4 46  1  0.92  0.96    0.933
```

В остальном все действия будут такими же.

6 Анализ независимых компонент

6.1 Какие бывают компоненты

В прошлых главах мы рассмотрели подробно, как можно представить исходные данные (матрицу **X**) в виде линейной комбинации нескольких компонент, которые мы в общем случае назвали *латентными переменными*. Вы уже знакомы со множеством примеров такого представления, самым важным из которых для хемометрики, пожалуй, является закон Ламберта-Бера. Напомним еще раз следствие этого закона для спектров смесей: спектр смеси отдельных составляющих является линейной комбинацией чистых спектров этих составляющих, а их концентрации — весами, с которыми чистые спектры вносят вклад в общую форму спектра смеси.

Математически это можно записать в виде:

$$\mathbf{X} = \mathbf{C}\mathbf{S}^{\mathrm{T}} + \mathbf{E}$$

Строки матрицы **X** — это спектры смесей, которые мы измеряем с помощью спектрометра. Т.е. число строк в **X** соответствует общему числу измерений, а число столбцов — числу длин волн на которых были измерены спектральные отклики.

Матрица **S** содержит чистые спектры исходных составляющих (измеренных на том же самом спектрометре, если речь идет о реальных данных). Каждый спектр представлен значениями (например, поглощением на разных длинах волн) в виде вектора-столбца этой матрицы, т.е. отдельные спектры в **S** находятся в столбцах этой матрицы, а не в строках, как в **X**.

Матрица **C** содержит в себе вклад каждой компоненты в каждой смеси (строк матрицы **X**), в нашем случае — это концентрации исходных составляющих.

Как мы обсуждали ранее, чистые спектры (столбцы матрицы **S**) геометрически представляют собой векторы в пространстве длин волн, а спектры смесей — линейную комбинацию этих векторов в том же пространстве. Если матрицы **C** и **S** априори известны, то вычислить теоретические значения для спектров смесей **X** очень просто, нужно лишь найти скалярное произведение этих двух матриц.

Приведем простой пример. Пусть мы смешиваем этанол и метанол в концентрации 0.2M и 0.8M (первая смесь) и 0.8M и 0.2M (вторая смесь). Предположим также, что наш инфракрасный спектрометр измеряет

поглощение на трех волнах с частотами 1087, 1047 и 1020 cm^{-1}. Если мы измерим спектр чистого этанола, то получим следующие значения $[1.6, 2.0, 0.2]$, для метанола эти значения будут $[0.4, 1.0, 2.2]$.

Тогда матрица с чистыми спектрами, **S** будет выглядеть следующим образом:

$1/\lambda$, cm^{-1}	Этанол	Метанол
1087	1.6	0.4
1047	2.0	1.0
1020	0.2	2.2

Точнее так будет выглядеть таблица со спектральными значениями, а матрица будет выглядеть так:

$$\begin{bmatrix} 1.6 & 0.4 \\ 2.0 & 1.0 \\ 0.2 & 2.2 \end{bmatrix}$$

Матрица с концентрациями **C** будет выглядеть следующим образом:

$$\begin{bmatrix} 0.2 & 0.8 \\ 0.8 & 0.2 \end{bmatrix}$$

Тогда матрица со спектрами этих двух смесей **X** будет иметь две строки (по числу смесей) и три столбца (по числу измеренных спектральных откликов) и теоретически мы можем найти ее как:

$$\mathbf{X} = \mathbf{C}\mathbf{S}^{\mathrm{T}} = \begin{bmatrix} 0.2 & 0.8 \\ 0.8 & 0.2 \end{bmatrix} \begin{bmatrix} 1.6 & 2.0 & 0.2 \\ 0.4 & 1.0 & 2.2 \end{bmatrix} = \begin{bmatrix} 0.64 & 1.2 & 1.8 \\ 1.36 & 1.8 & 0.6 \end{bmatrix}$$

Т.е. в спектре первой смеси будет выделяться третий пик, который характерен для метанола, а в спектре второй смеси — первые два, которые характерны для этанола.

Экспериментальные спектры, измеренные для приготовленных смесей с помощью спектрометра, будут немного отличаться от теоретических наличием шума (случайных вкладов, связанных с несовершенством приборов и различными внешними факторами), который в нашем исходном соотношении задан матрицей **E**.

В реальных исследованиях часто нужно решить обратную задачу — найти оценки $\hat{\mathbf{S}}$ и $\hat{\mathbf{C}}$ по измеренным спектрам смесей **X** так, чтобы эти оценки были как можно ближе к теоретическим значениям **S** и **C**.

Если известны значения одной из этих матриц, то решение найти довольно просто, используя классический метод наименьших квадратов, который мы рассматривали в главе про множественную

линейную регрессию. Но если обе матрицы неизвестны, и по спектрам смесей нужно найти и спектры чистых компонент (так как зачастую точный химический состав смесей неизвестен и нахождение таких чистых спектров позволит его определить) и их концентрации, то однозначного решения этой задачи не существует. Так как существует бесконечно большое число возможных $\hat{\mathbf{S}}$ и $\hat{\mathbf{C}}$ которые в результате их скалярного умножения дадут идентичные значения для матрицы $\mathbf{X}$.

Для того, чтобы эту задачу решить необходимо ввести дополнительные ограничения (англ. *constraints*). В главе 2 мы подробно познакомились с одним из таких решений — методом главных компонент. МГК находит компоненты с учетом двух критериев:

1. Ориентация главной компоненты в пространстве выбирается так, чтобы объяснить максимальную величину дисперсии текущих данных. Т.е. если мы спроецируем все точки, которые соответствуют строками $\mathbf{X}$, на эту компоненту, то их дисперсия будет максимально возможной в таком случае.
2. Каждая следующая компонента перпендикулярна (ортогональна) всем предыдущим. Другими словами вклад каждой компоненты не зависит от вклада других компонент.

В терминах МГК матрица $\mathbf{S}$ — это матрица нагрузок, обычно она обозначается буквой $\mathbf{P}$. Матрица $\mathbf{C}$ в МГК играет роль матрицы счетов и обычно обозначается как $\mathbf{T}$. Соответственно привычное нам МГК решение этой проблемы выглядит как:

$$\mathbf{X} = \mathbf{T}\mathbf{P}^{\mathrm{T}} + \mathbf{E}$$

Заметим, что математической разницы между этим и исходным соотношениями нет, мы просто используем разные обозначения для матрицы компонент и матрицы вкладов, чтобы подчеркнуть, что найденной решение — это, на самом деле, разложение на главные компоненты.

С одной стороны, два критерия, которые используются для МГК, позволяют получить однозначное (уникальное) решение для любой матрицы $\mathbf{X}$. Например, с помощью алгоритма NIPALS, который мы рассмотрели в главе, посвященной МГК, или же другим способом, например, используя разложение по сингулярным значениям (англ. *singular values decomposition, SVD*), которое мы обсуждали в этой же главе. С другой стороны, МГК решение часто не имеет физического, или химического смысла, и вот почему.

Во-первых, нагрузки (т.е. значения матрицы $\mathbf{S}$, если рассматривать наше исходное, более общее представление) в МГК могут быть отрицательными, так как в МГК направление компонент неважно и никак не ограничивается (важна лишь ориентация компонент). Но если компоненты — это чистые спектры составляющих некоторой смеси, то это невозможно, так как поглощение, или отражение не может быть отрицательным.

Во-вторых, ортогональность компонент тоже плохо уживается с нашими знаниями о спектральных данных, так как спектры чистых компонент могут иметь общие, или частично пересекающиеся пики, что не позволяет им быть полностью ортогональными.

Поясним это на еще одном простом примере. В этот раз сделаем его более наглядным с помощью R.

Блок кода ниже показывает генерацию симулированных спектров с помощью функции Гаусса (известной вам из раздела про нормальное распределение). Сначала мы генерируем два "спектра", каждый из которых состоит из 100 значений ("длин волн" или спектральных каналов), пики на которых почти не пересекаются. Центры пиков расположены на позициях 30 и 60 (максимальное значение функция Гаусса имеет в точке, которая соответствует среднему значению), а их ширина задается стандартным отклонением, которое в нашем примере равно 5. Т.е. пики находятся друг от друга на расстоянии 6 стандартных отклонений и пересекаются лишь незначительно. Графическое представление этих спектров показано на левом верхнем графике рисунка 6.1.

```
# генерируем вектор с "длинами волн"
w <- 1:100

# генерируем два спектра с почти непересекающимися пиками
# с помощью функции плотности распределения вероятности для
# нормального распределения (функции Гаусса)
s11 <- dnorm(w, mean = 30, sd = 5)
s12 <- dnorm(w, mean = 60, sd = 5)
```

После этого мы генерируем другие два спектра, также состоящие из 100 значений, но с пиками, которые расположены ближе друг к другу — на расстоянии двух стандартных отклонений. Код ниже показывает как это сделать в R, а сами спектры представлены на правом верхнем графике рисунка 6.1.

```
# генерируем два спектра с частично пересекающимися пиками
# с помощью функции плотности распределения вероятности для
# нормального распределения (функции Гаусса)
s21 <- dnorm(w, mean = 40, sd = 5)
s22 <- dnorm(w, mean = 50, sd = 5)
```

Из главы 1 мы знаем, что, во-первых, любой спектр можно представить в виде вектора в пространстве длин волн, а во-вторых, если взять два вектора, нормировать их так, чтобы длина каждого была равна единице, то скалярное произведение этих векторов даст косинус угла между ними. Если угол прямой, т.е. векторы ортогональны, то косинус будет равен нулю. Мы можем также вычислить значение самого угла через арккосинус скалярного произведения.

Блок кода ниже показывает вычисление углов между двумя спектрами, для каждого из двух случаев, которые мы сгенерировали.

```
# нормируем значение спектров из первой пары так
# чтобы соответствующие векторы имели единичную длину
s11 <- s11 / sqrt(sum(s11^2))
s12 <- s12 / sqrt(sum(s12^2))

# нормируем значение спектров из второй пары так
# чтобы соответствующие векторы имели единичную длину
s21 <- s21 / sqrt(sum(s21^2))
s22 <- s22 / sqrt(sum(s22^2))

# вычисляем угол между векторами в каждой паре
# и пересчитываем их значения из радианов в градусы
a1 <- acos(crossprod(s11, s12)) / pi * 180
a2 <- acos(crossprod(s21, s22)) / pi * 180

# показываем оба значения
show(c(a1, a2))
```

```
[1] 89.99293 68.41510
```

Как можно видеть в результате, угол между спектрами, пики которых почти не пересекаются, на самом деле практически прямой (89.993 градусов). Но в случае со спектрами, пики которых частично пересекаются, этот угол гораздо меньше — 68.415 градусов. Т.е. вклад этих двух спектров в линейной комбинации уже не будет полностью независимым и решить обратную задачу нахождения этих спектров в виде главных компонент не получится.

Для того, чтобы сделать этот пример еще более наглядным, давайте сократим число спектральных каналов в каждой паре спектров до двух — возьмем только значения, которые соответствую положению пиков. Далее мы также нормируем полученные спектры на единичную длину и вычислим углы между ними. Код ниже показывает как это сделать для обоих пар.

```
# сокращаем спектры до двух значений в первой паре
# взяв их величину для каналов 30 и 60, где расположены пики
ss11 <- s11[c(30, 60)]
ss12 <- s12[c(30, 60)]

# сокращаем спектры до двух значений в второй паре
# взяв их величину для каналов 40 и 50, где расположены пики
ss21 <- s21[c(40, 50)]
ss22 <- s22[c(40, 50)]

# нормируем значение спектров из первой пары так
# чтобы соответствующие векторы имели единичную длину
ss11 <- ss11 / sqrt(sum(ss11^2))
ss12 <- ss12 / sqrt(sum(ss12^2))

# нормируем значение спектров из второй пары так
# чтобы соответствующие векторы имели единичную длину
ss21 <- ss21 / sqrt(sum(ss21^2))
ss22 <- ss22 / sqrt(sum(ss22^2))

# вычисляем угол между векторами в каждой паре
# и пересчитываем их значения из радианов в градусы
aa1 <- acos(crossprod(ss11, ss12)) / pi * 180
aa2 <- acos(crossprod(ss21, ss22)) / pi * 180

# показываем оба значения
show(c(aa1, aa2))
```

```
[1] 90.00000 74.58537
```

Как можно видеть, в этом случае угол для первой пары спектров точно равен 90 градусам, так как точки с максимальными значениями пиков не пересекаются (значение второго спектра в точке 30 практически равно нулю). Во втором же случае угол равен 74.6 градусам, так как второй спектр имеет ненулевое значение в точке, где первый спектр имеет максимум и наоборот. Так как в этом случае каждый спектр имеет только два значения, мы можем визуализировать их в виде векторов в двумерном пространстве, как показано на двух нижних графиках на рисунке 6.1. Теперь мы можем четко видеть, что угол между двумя спектрами во втором случае на самом деле не прямой.

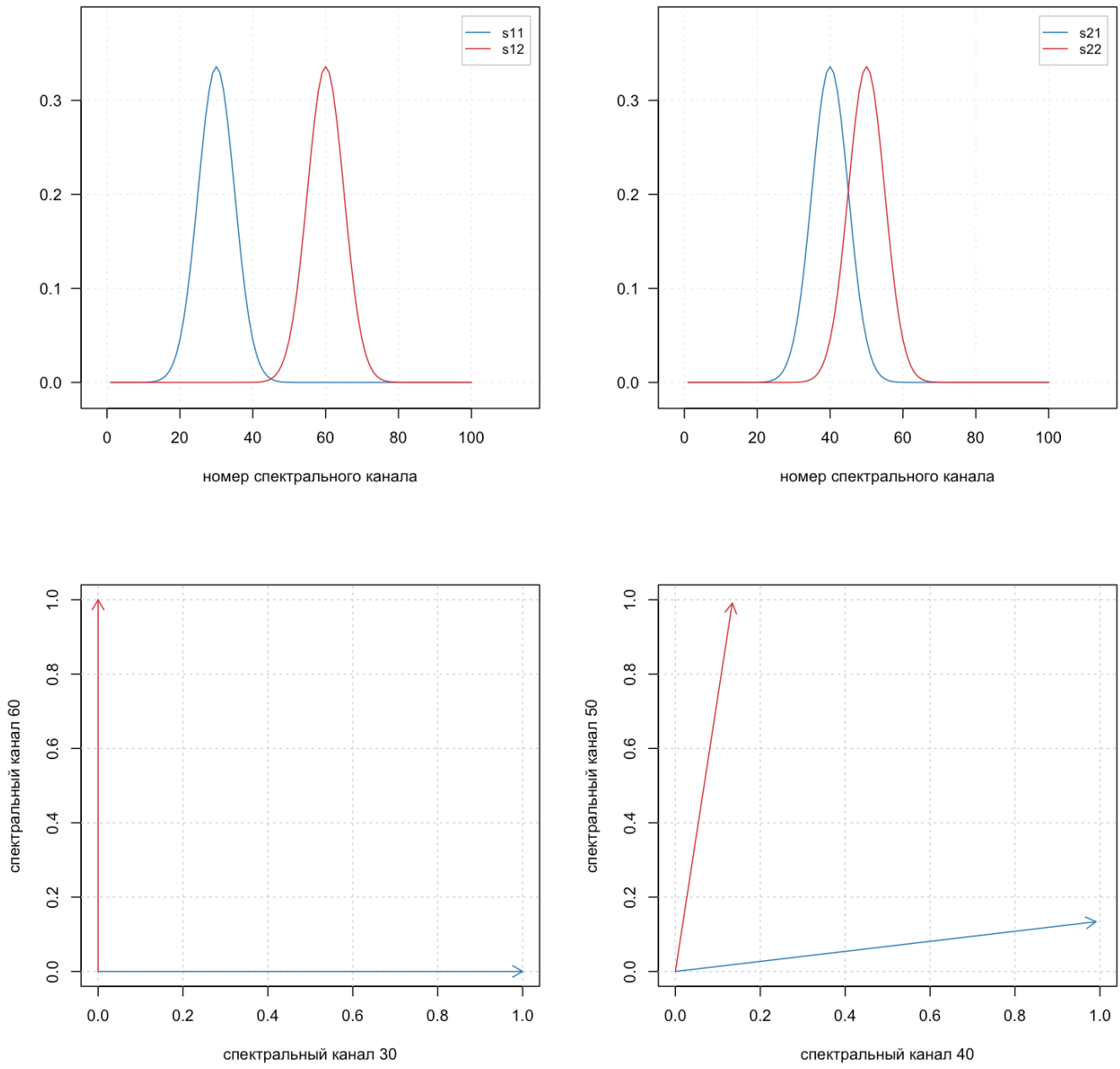


Рис. 6.1. Сгенерированные спектры чистых компонент. Сверху спектры показаны в виде линейных графиков. Снизу в виде векторов в двумерном пространстве, оси которых соответствуют положению пиков

Попробуем теперь сгенерировать спектры смесей для этих двух случаев и посмотрим сможет ли МГК правильно их разделить. Для этого мы зададим вручную значения векторов концентраций для 10 смесей, и используем соотношение для закона Ламберта-Бера, как показано в блоке кода ниже.

```
# зададим концентрации для каждой компоненты (спектра)
# и объединим их в матрицу C
c1 <- c(0.10, 0.25, 0.05, 0.07, 0.11, 0.50, 0.61, 0.92, 0.81, 0.72)
c2 <- c(0.20, 0.05, 0.75, 0.97, 0.61, 0.05, 0.01, 0.01, 0.08, 0.07)
C <- cbind(c1, c2)

# объединим спектры из каждой пары в матрицу S
S1 <- cbind(s11, s12)
S2 <- cbind(s21, s22)

# вычислим спектры смесей
X1 <- tcrossprod(C, S1)
X2 <- tcrossprod(C, S2)
```

Вы можете визуализировать эти спектры с помощью встроенной в R функции `matplot`, как показано в блоке с кодом ниже, или же воспользоваться методами `mdaplot` или `mdaplotg` из пакета `mdatools`, которые делают построение таких графики проще.

Вот код, которые позволит построить нужные графики в чистом R без специальных пакетов.

```
# сгенерируем цвета в виде набора оттенков серого
intensity <- seq(0.1, 0.9, length.out = nrow(X1))
col <- rgb(red = intensity, blue = intensity, green = intensity)

# разделим окно на две части и покажем все спектры из X1 в виде кривых
par(mfrow = c(1, 2))
matplot(w, t(X1), type = "l", lty = 1, col = col)

# добавим чистые спектры
lines(w, s11, col = "red")
lines(w, s12, col = "blue")

# покажем все спектры из X2 в виде кривых
matplot(w, t(X2), type = "l", lty = 1, col = col)

# добавим чистые спектры
lines(w, s21, col = "red")
lines(w, s22, col = "blue")
```

Графики на рисунке 6.2 показывают спектры смесей в полном варианте, в виде кривых (верхние два графика, вы сможете получить похожие, если воспользуетесь блоком кода выше) и в укороченном варианте, в виде точек (нижние два графика, попробуйте реализовать их сами). Спектры смесей показаны оттенками серого, тогда как чистые спектры показаны тем же цветом, что и на предыдущем рисунке. Как можно видеть, концентрации специально подобраны так, чтобы подчеркнуть наличие двух независимых компонент в этих смесях.

Однако, если применить МГК, то результат будет не совсем таким, как нам бы хотелось. Рисунок 6.3 показывает главные компоненты (нагрузки) в виде кривых (верхние графики), и в виде векторов (нижние графики) в двумерном пространстве переменных, соответствующих пикам, как на предыдущем рисунке. Здесь компоненты показаны цветными сплошными линиями и кривыми, тогда как оригинальные спектры показаны пунктирными полупрозрачными линиями для сравнения. Также следует отметить, что данные не были центрированы, как мы обычно делаем в МГК, так как центрирование в этом случае не имеет физического смысла.

Вот блок кода, который позволит вам воспроизвести этот пример. Имейте в виду, что в данном случае используется метод `pca()` из пакета `mdatools` для МГК разложения. Вместо этого вы можете воспользоваться функцией `nipals()` или `pcasvd()` которые мы вместе написали в главе 2.

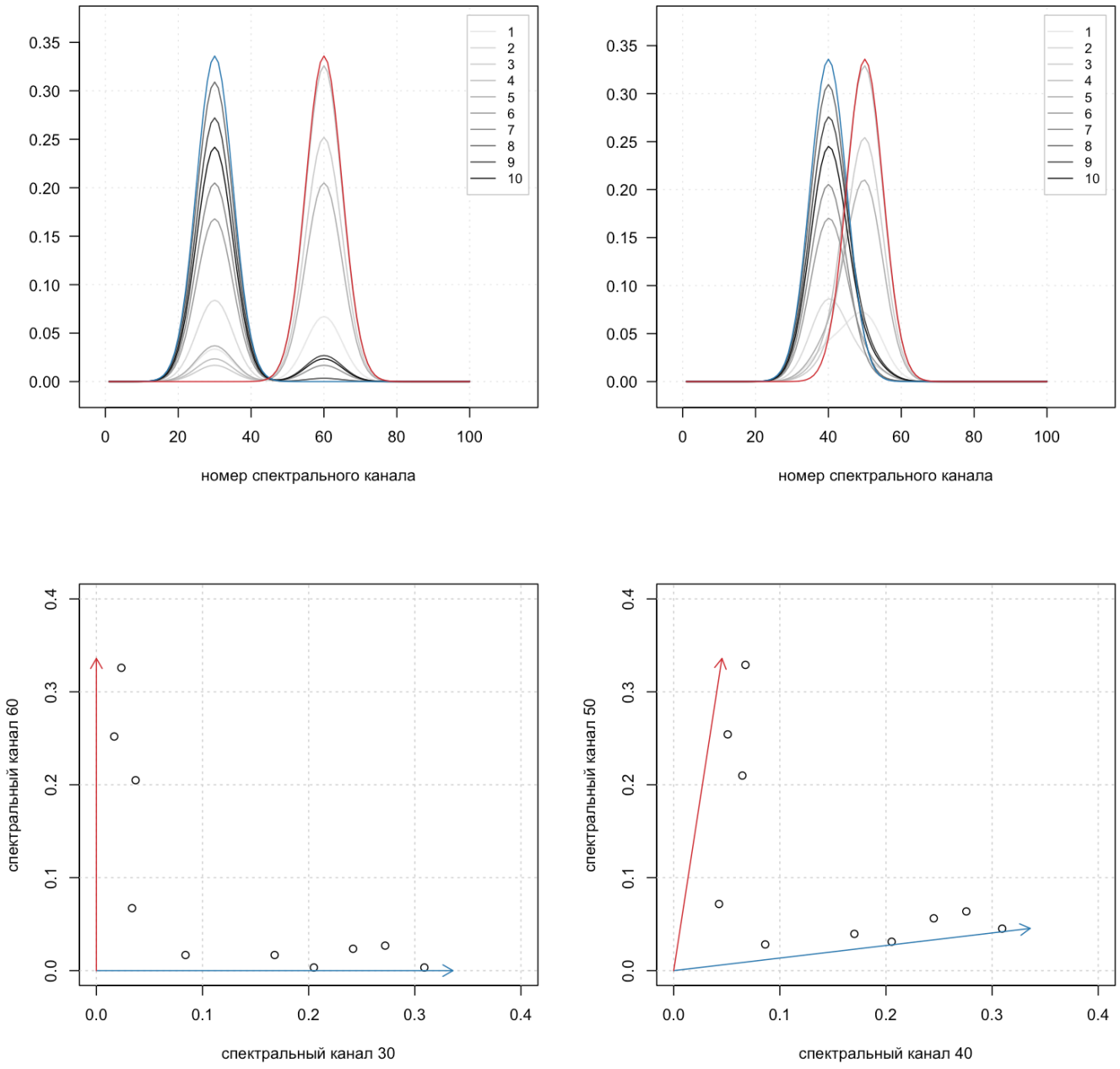


Рис. 6.2. Сгенерированные спектры чистых компонент (цветные объекты) и их смесей (объекты показанные оттенками серого). Сверху спектры показаны в виде линейных графиков. Снизу в виде векторов в двумерном пространстве, которое соответствует положению пиков

```
# загружаем пакет mdatools
library(mdatools)

# разделим окно на две части
par(mfrow = c(1, 2))

# делаем МГК разложений первого набора спектров с 2 компонентами без центрирования
m1 <- pca(X1, 2, center = FALSE)

# показываем нагрузки в виде кривых
matplot(m1$loadings, type = "l", col = c("blue", "red"), lty = 1)

# добавим чистые спектры используя полупрозрачные цвета
lines(w, s11, col = "#0000ff40", lty = 3)
lines(w, s12, col = "#ff000040", lty = 3)

# делаем МГК разложений второго набора спектров с 2 компонентами без центрирования
m2 <- pca(X2, 2, center = FALSE)

# показываем нагрузки в виде кривых
matplot(m2$loadings, type = "l", col = c("blue", "red"), lty = 1)

# добавим чистые спектры используя полупрозрачные цвета
lines(w, s21, col = "#0000ff80", lty = 3)
lines(w, s22, col = "#ff000080", lty = 3)
```

Как видно из графиков, результаты МГК для первого случая с разделенными пиками, довольно неплохие. Нагрузки по своей форме довольно близки к оригинальным спектрам (с точностью до знака, конечно) — это можно видеть и по верхним, и по нижним графикам. Однако, в каждом векторе нагрузок имеется влияние от обоих чистых спектров (это можно видеть по двум пикам для каждой компоненты в верхнем графике), что влияет на точность результата. Этот эффект виден на нижнем графике в виде поворота главных компонент на небольшой угол.

В случае же перекрывающихся пиков результат гораздо хуже. Дело в том, что направление максимального разброса данных (к чему и чувствителен МГК) в этом случае находится на диагонали между чистыми спектрами, где и проходит первая главная компонента. В этом случае нагрузки можно интерпретировать с точки зрения вариации данных, но не с точки зрения физики, или химии.

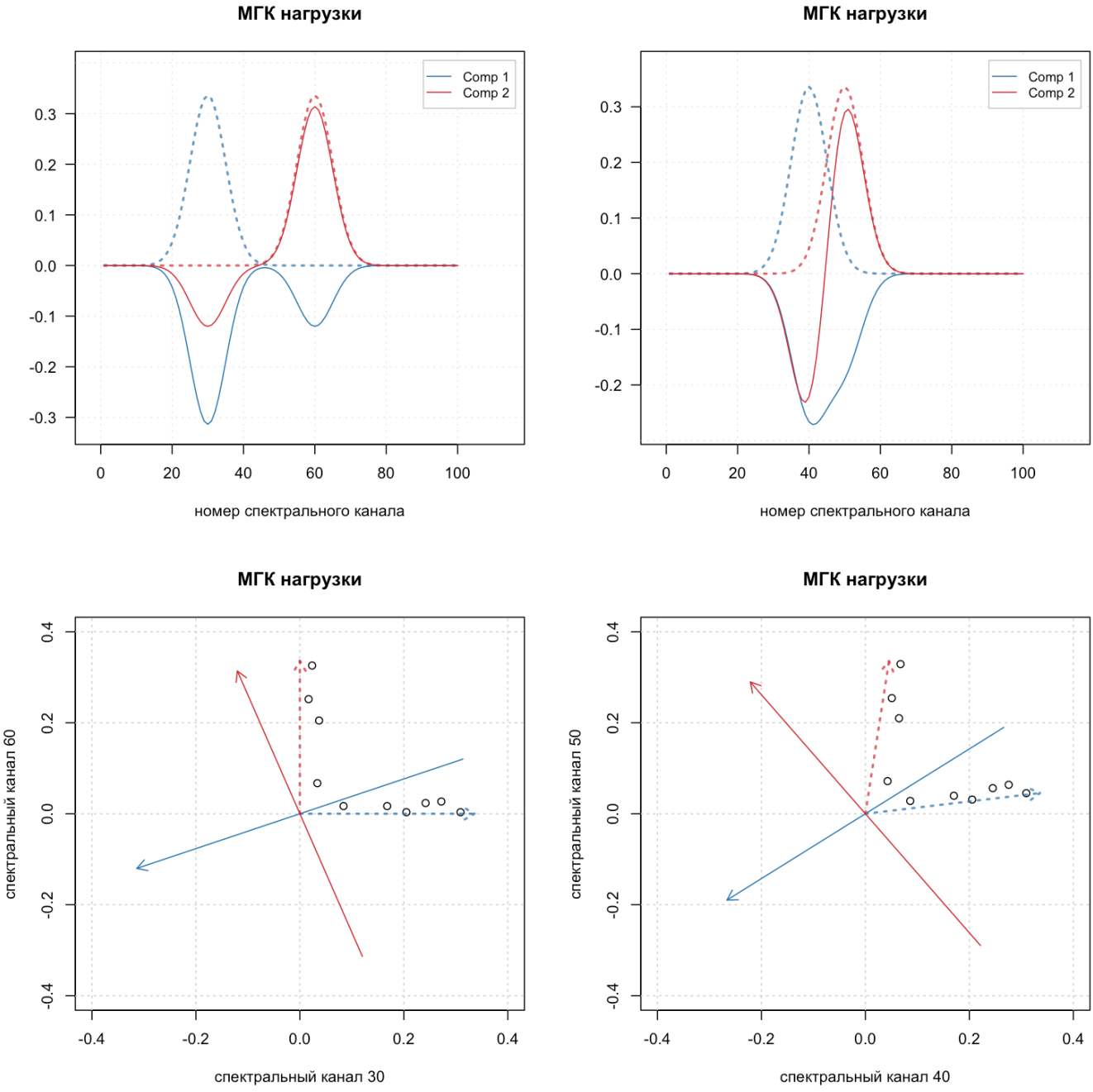


Рис. 6.3. Результаты МГК декомпозиции (графики нагрузок).

Другими словами, ограничения, которые используются для нахождения компонент в МГК (в первую очередь, принцип максимизации дисперсии и ортогональность компонент), не очень хорошо подходят, если необходимо разложить спектры смесей на линейную комбинацию спектров их составляющих.

Это конечно не означает, что МГК бесполезен для анализа спектров. Метод главных компонент можно и нужно использовать и для разведывательного анализа спектральных данных, и для их классификации с помощью SIMCA, но в результате разложения этих данных на комбинацию главных компонент векторы нагрузок чаще всего не будут соответствовать чистыми спектрам, а векторы счетов — концентрациям.

В следующем разделе мы поговорим, как обойти эти ограничения и попытаться решить поставленную выше задачу.

6.2 Анализ независимых компонент

Для начала следует сделать небольшое отступление и сказать, что поставленная нами в предыдущем разделе задача — разложить линейную комбинацию нескольких сигналов (спектров) на чистые сигналы и матрицу вкладов (концентраций) — является весьма актуальной не только в аналитической химии, но и в других областях науки и техники. Например, похожие задачи решается при распознавании речи, анализе электрокардиограмм, или энцефалограмм мозга в медицине, при анализе изображений, и так далее.

В английском языке для общего обозначения таких задач и методов их решения используется термин *разделения* (англ. *unmixing*), или *разрешения* (англ. *resolution*) сигналов, или кривых. В хемометрике более распространен второй термин *разрешение кривых* (*curve resolution*). В русском языке также иногда используется термин *декомпозиция* сигналов.

Так как задача разрешения кривых в общем виде однозначного решения не имеет, существует очень широкий круг методов, которые пытаются найти приближенное решение, используя различные подходы и хитрости. В этой главе мы познакомимся с одним из них, методом независимых компонент, однако, существуют и более эффективные методы, так что мы рекомендуем не останавливаться на материале этой главы, а использовать ее как базу для более глубокого изучения современных методов и подходов к разрешению кривых.

В основе анализа независимых компонент АНК (англ. *independent component analysis, ICA*), лежит несколько идей.

Во-первых, вместо того, чтобы представлять данные в виде точек в пространстве переменных (например, длин волн), мы представляем их в виде сигналов. Т.е. каждая строка в матрице $\mathbf{X}$ это сумма строк в матрице $\mathbf{S}^T$ с разными весами. Формально разницы здесь нет, любые многомерные данные можно представлять и так и эдак, но для понимания работы АНК это принципиально — в АНК большинство операций делаются именно со строками $\mathbf{X}$ (т.е. с отдельными спектрами смесей, которые мы измерили).

Во-вторых, мы предполагаем, что вариации интенсивностей в каждом чистом спектре (строки матрицы $\mathbf{S}^{\mathrm{T}}$) слабо зависимы. Другими словами, величина интенсивности для определенной длины волны в одном спектре не зависит (ну или зависит, но слабо в случае похожих спектров) от величины интенсивности на этой длине волны в другом спектре. Независимость иногда связывают с нулевой корреляцией между спектрами, но это не полностью взаимозаменяемые определения. Хотя, для упрощения можно считать, что мы ожидаем, что спектры чистых компонент имеют слабую корреляцию.

В-третьих, каждый чистый спектр не является случайным набором интенсивностей. Другими словами, если мы возьмем все интенсивности (например, поглощения, или отражения на разных длинах волн) и построим гистограмму их распределения, она не должна быть такой, как у полностью случайных величин. Как мы знаем из главы 1, большинство полностью случайных величин распределены нормально, форма их гистограммы распределения хорошо описывается функцией Гаусса. А это значит, что мы ожидаем, что интенсивности наших чистых спектров будут как можно хуже описываться этим законом.

И, наконец, в-четвертых, интенсивности спектров смесей (строки матрицы $\mathbf{X}$) будут как раз ближе к случайным величинам и их распределение будет лучше описываться функцией Гаусса нежели интенсивности чистых спектров. Это вытекает из центральной предельной теоремы, согласно которой если у вас есть несколько независимых источников сигнала, то суммируя эти источники с разными весами, вы приближаете эту сумму к абсолютно случайному сигналу. Например, если вы попадете на вечеринку с большим количеством людей, то первое время вы будете слышать звук, больше похожий на шум, чем на отдельные голоса.

Итак, приведем еще раз основные предположения для АНК в более короткой форме:

1. Работаем со строками матрицы данных (измеренными спектрами смесей).
2. Спектры чистых компонент статистически независимы.
3. Распределение интенсивности чистых спектров не подчиняется закону Гаусса.
4. Распределение интенсивности спектров смесей лучше описывается законом Гаусса.

Собственно, это и есть те самые ограничения, которые нам нужны, чтобы сузить набор возможных решений и попытаться найти обе матрицы $\hat{\mathbf{C}}$ и $\hat{\mathbf{S}}$ с помощью анализа независимых компонент. Однако, следует заметить, что способов реализации этих ограничений существует несколько, соответственно, есть довольно много разных алгоритмов АНК.

В этой книге мы подробно рассмотрим самый простой из них, *FastICA*, который использует третий принцип — “негауссовости” чистых спектров. В конце главы мы также дадим короткий обзор других алгоритмов, в том числе разработанных специально для анализа спектральных данных.

6.3 Алгоритм FastICA

Прежде чем описать алгоритм, давайте вспомним как можно оценить, является ли распределение чисел нормальным, или нет. Для этого есть много разных способов, некоторые из которых мы рассмотрели в главе 1. Одним из самых простых (хоть и не самых точных) способов является вычисление коэффициента эксцесса (англ. *kurtosis*), который показывает насколько острым является пик гистограммы распределения. Для нормального распределения теоретическое значение этого коэффициента равно нулю.

Т.е. для того, чтобы значения интенсивностей спектра были "негауссовыми", нужно чтобы этот коэффициент был как можно больше. Этот критерий мы и будем использовать в нашей реализации алгоритма *FastICA*.

Сам алгоритм в упрощенном виде состоит из следующих этапов:

1. Делаем так, чтобы спектры смесей (строки матрицы $\mathbf{X}$) не имели корреляции друг с другом и имели единичную дисперсию.
2. Находим вектор $\mathbf{w}$, такой, что $\mathbf{s} = \mathbf{w}^{\mathrm{T}}\mathbf{X}$ имеет наибольший коэффициент эксцесса.
3. Повторяем второй пункт для следующей компоненты, но при этом убираем зависимость от предыдущих найденных компонент.
4. Когда все компоненты найдены — завершаем работу.

Рассмотрим каждый этап подробнее и напишем свою реализацию этого алгоритма. В этой реализации мы будет использовать самый простой, но не самый эффективный подход. Это позволит нам избежать ненужных усложнений в объяснении принципов работы АНК, но на практике мы рекомендуем пользоваться более эффективными реализациям, о которых также будет упомянуто ниже.

"Отбеливание" исходных данных

Процесс предварительной подготовки спектров смесей для АНК носит название "отбеливания" (англ. *whitening*) и включает в себя два этапа. Первый этап делает так, чтобы строки исходной матрицы со спектрами смесей не коррелировали друг с другом. Второй — чтобы дисперсия значений в каждой строке была равна единице.

К счастью, мы уже знаем метод, который позволяет "убить" корреляцию — это метод главных компонент. Однако, МГК убирает корреляцию между переменными (столбцами), тогда как нам нужно сделать это для строк исходной матрицы спектров. Т.е. нам нужно просто сделать МГК разложение транспонированной матрицы $\mathbf{X}$, получить матрицу счетов и транспонировать ее обратно. После этого шкалировать значения в каждой строке этой матрицы так, чтобы дисперсия этих значений была равна единице.

Для этого подойдет метод сингулярного разложения, SVD, которые мы рассмотрели в главе про МГК. Для примера мы будем применять АНК к матрице со смесями, полученными для частично перекрывающихся

пиков, которую мы сделали в предыдущем разделе (если вы запускали этот код, то эта матрица находится в переменной X2).

```
# скопируем значения из X2
X <- X2

# центрируем строки этой матрицы вычитая среднее значение из каждой строки
# для этого транспонируем матрицу, центрируем ее столбцы, и транспонируем обратно
X <- t(scale(t(X), center = TRUE, scale = FALSE))

# применяем SVD разложение для транспонированной матрицы
m <- svd(t(X))
```

Прежде чем продолжить "декорреляцию", давайте посмотрим на сингулярные значения, которые получились в итоге:

```
show(m$d)
```

```
[1] 1.672337e+00 1.086523e+00 5.743864e-16 3.895857e-16 1.863939e-16
[6] 8.514318e-17 5.194700e-17 4.799793e-17 3.340359e-17 1.626838e-17
```

Как можно заметить, только для первых двух компонент сингулярные значения не равны нулю, все остальные значения практически нулевые (10^{-16} и меньше). Это связано с тем, как мы сгенерировали данные. Если вы помните, строки нашей матрицы со спектрами смесей, являются суммой двух чистых спектров с разными весами. Поэтому все компоненты старше двух не имеют никакого значения и должны быть отброшены. Т.е. в "отбеленной" матрице должно быть не 10 строк, как в оригинальной, а всего 2.

На практике определение числа независимых компонент в отбеленной матрице решается разными способами, например, сравнением сингулярных значений с некоторой пороговой величиной. Однако, самым простым решением будет использовать число искомых компонент. Т.е. если мы применяем АНК для разложения исходных спектров смесей на A компонент, то мы можем задать число строк в "отбеленной" матрице $\mathbf{Z}$ равным A, что мы и сделаем.

```
# берем нормированные счета для первых 2-х компонент и транспонируем их
# в "отбеленную" матрицу Z
Z <- t(m$u[, 1:2])
```

С декорреляцией разобрались, теперь нам нужно убедиться, что дисперсия строк в матрице $\mathbf{Z}$ будет единичной. Для этого надо вспомнить, как посчитать дисперсию вектора значений $\mathbf{z} = [z_1, z_2, ..., z_n]$:

$$s^2(\mathbf{z}) = \frac{1}{n-1}\sum_{i=1}^{n}(z_i - m_z)^2$$

где m_z — это среднее, вычисленное для этого же набора значений.

Однако, в нашем случае мы знаем, что среднее равно нулю априори (так как мы центрировали строки, когда делали SVD). Это значит, что нам не нужно его вычислять и, следовательно, мы не потеряем одну степень свободы. Это упрощает наше соотношение до следующего:

$$s^2(\mathbf{z}) = \frac{1}{n}\sum_{i=1}^{n}(z_i)^2$$

С другой стороны, из раздела, где мы рассказывали про разложение по сингулярным значениям, мы знаем, что длина каждого сингулярного вектора равна единице, т.е.

$$||\mathbf{z}|| = \sqrt{\sum_{i=1}^{n}(z_i)^2} = 1$$

Поэтому, для того, чтобы получить единичную дисперсию, нам нужно умножить каждое значение на квадратный корень из n (число переменных в исходных данных или число столбцов в матрицах $\mathbf{X}$ и $\mathbf{Z}$):

```
Z <- Z * sqrt(ncol(Z))
```

Проверим, что оба критерия — отсутствие корреляции между строками и единичная дисперсия значений для каждой строки — выполняются:

```
# проверяем отсутствие корреляции между строками
show(cor(t(Z)))
```

```
              [,1]          [,2]
[1,]  1.000000e+00 -1.110223e-16
[2,] -1.110223e-16  1.000000e+00
```

```
# проверяем что каждая строка имеет единичную дисперсию
show(rowSums(Z^2) / ncol(Z))
```

```
[1] 1 1
```

Заметим, что так как, по сути, матрица $\mathbf{Z}$ — это счета МГК разложения транспонированной матрицы $\mathbf{X}$, мы можем их получить, умножив $\mathbf{X}^{\mathrm{T}}$ на МГК нагрузки, и затем разделить результат на соответствующие собственные значения и транспонировать все обратно. Т.е. мы можем сформировать так называемую "отбеливающую матрицу" $\mathbf{M}_W$, которая при умножении исходной матрицы данных на нее также будет давать $\mathbf{Z}$:

$$\mathbf{Z} = \mathbf{M}_W\mathbf{X}$$

Вычислим такую матрицу и проверим, что соотношение выше действительно выполняется:

```
# формируем "отбеливающую матрицу"
MW <- t(m$v[, 1:2] %*% diag(1 / m$d[1:2]) * sqrt(ncol(Z)))

# проверяем что Z =  MW * X
# вычисляя сумму квадратов разницы между ними
sum( (Z - MW %*% X)^2 )
```

```
[1] 9.153192e-29
```

Ну и, конечно, мы можем получить и обратный результат — по "отбеленной" матрице $\mathbf{Z}$ восстановить исходную матрицу $\mathbf{X}$ для этого нам нужно сформировать матрицу обратную "отбеливающей":

$$\mathbf{X} = \mathbf{M}_W^{-1}\mathbf{Z}$$

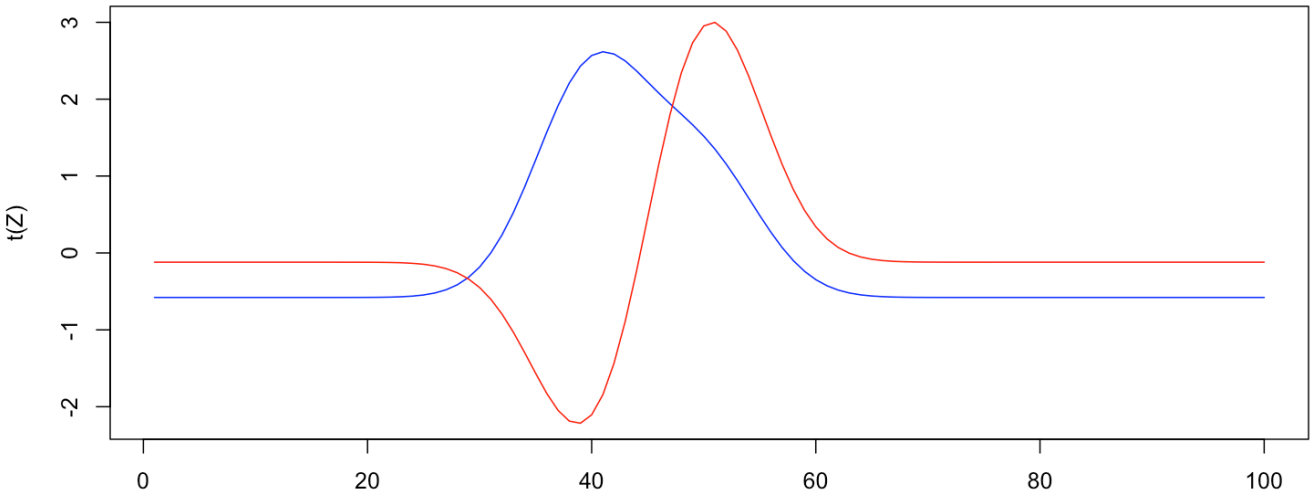
Опять же, для МГК это сделать довольно просто:

```
MWI <- t(diag(m$d[1:2]) %*% t(m$v[, 1:2]) * 1 / sqrt(ncol(Z)))

# проверяем что X = MWI * Z
# вычисляя сумму квадратов разницы между ними
sum( (X - MWI %*% Z)^2 )
```

```
[1] 1.549349e-30
```

Теперь попробуем посмотреть на две строки из $\mathbf{Z}$ в виде графиков, чтобы понять насколько мы уже близки к искомому решению.

```
matplot(t(Z), type = "l", col = c("blue", "red"), lty = 1)
```



Как мы видим, полученное на этом этапе решение очень похоже на графики нагрузок, которые у нас получились в результате классического МГК разложения этих спектров. Собственно, это практически они и есть, и это решение пока далеко до искомого.

Теперь можно приступить к основному этапу.

Нахождение первой независимой компоненты

Итак, матрица $\mathbf{Z}$ — это "отбеленная" версия исходной матрицы $\mathbf{X}$, в которой отсутствует корреляция между строками и дисперсия каждой строки равна единице. Как мы узнали только что, по сути, строки матрицы $\mathbf{Z}$ — это МГК нагрузки, отшкалированные на единичную дисперсию. Т.е. можно сказать, что АНК основывается на результатах метода главных компонент и "улучшает" их чтобы удовлетворить требуемым критериям ("негауссовость" в первую очередь).

Для того, чтобы найти первую независимую компоненту $\mathbf{s}_1$, будем использовать метод градиентного спуска — итерационный процесс, который обеспечит нам значения $\mathbf{s}_1$ с наибольшим значением коэффициента эксцесса.

В этом алгоритме мы будем искать вектор $\mathbf{w}_1$, такой, что $\mathbf{s}_1 = \mathbf{w}^{\mathrm{T}}\mathbf{Z}$.

Алгоритм следующий:

1. Сгенерировать вектор $\mathbf{w}_1$ со случайными координатами.
2. Нормировать этот вектор на единичную длину
3. Вычислить первое приближение $\mathbf{s}_1 = \mathbf{w}_1^T \mathbf{Z}$
4. Посчитать коэффициент эксцесса для $\mathbf{s}_1$ и его производную.
5. Использовать производную чтобы сделать следующее приближение для $\mathbf{w}_1$
6. Вернуться к пункту 2 и повторять пока значения $\mathbf{w}_1$ не перестанут меняться

Вот так выглядит этот алгоритм (точнее его одна итерация) на R. Не запускайте пока эти команды.

```
M <- ncol(Z)

# 1. генерируем начальное приближение для w1
w1 <- matrix(runif(nrow(Z)), ncol = 1)

# 2. нормируем вектор на единичную длину
w1 <- w1 / sqrt(sum(w1^2))

# 3. вычисляем первое приближение для s1
s1 <- crossprod(w1, Z)

# 4.-5. вычисляем новое значение для w1 на основе производной
#       для эксцесса посчитанного для s1
w1 <- (Z %*% t(s1^3)) / M - 3 * w1
```

Давайте перепишем этот код в более компактной форме и добавим несколько итераций, чтобы обеспечить сходимость. Для простоты мы не будет проверять насколько хороша эта сходимость, вместо этого просто проделаем шаги со второго по пятый 100 раз, этого должно быть достаточно для нашего примера.

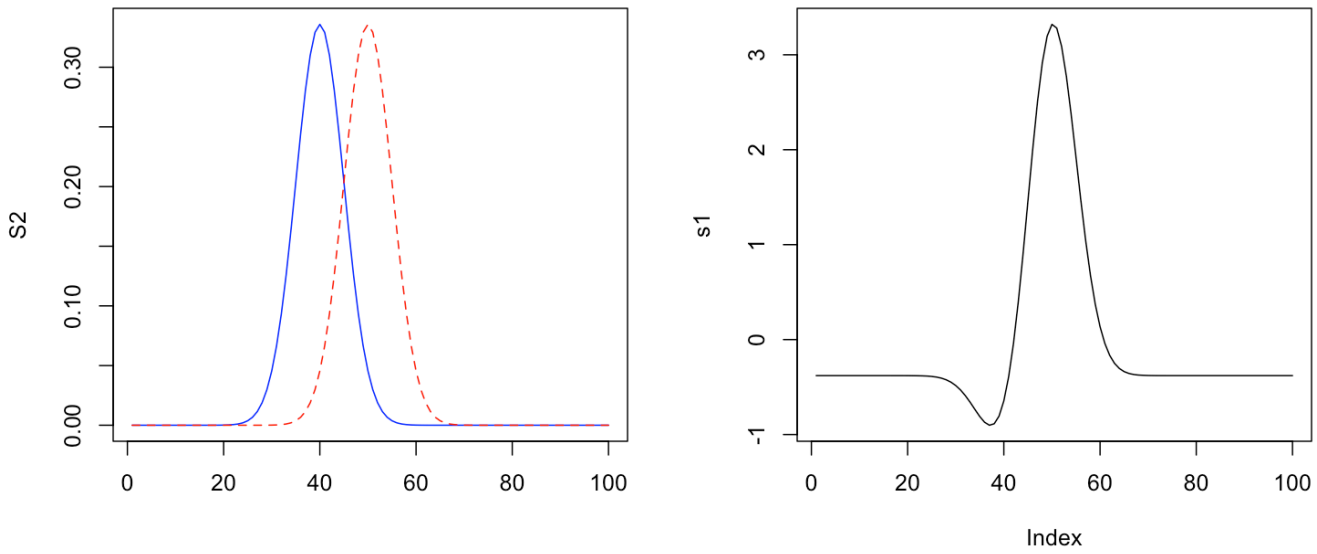
Вот новая версия этого кода:

```
set.seed(42)

M <- ncol(Z)
w1 <- matrix(runif(nrow(Z)), ncol = 1)
w1 <- w1 / sqrt(sum(w1^2))

for (i in 1:100) {
   s1 <- crossprod(w1, Z)
   w1 <- (Z %*% t(s1^3)) / M - 3 * w1
```

```
   w1 <- w1 / sqrt(sum(w1^2))
}


s1 <- crossprod(Z, w1)
```

Запустите его (убедитесь при этом что у вас уже имеется матрица **Z** в среде окружения, как результат отбеливания, которое мы проделали выше). После этого давайте посмотрим, что у нас получилось в итоге. Рисунок ниже показывает два графика — оригинальные чистые спектры (слева), которые мы использовали для получения спектров смесей, и график независимой компоненты (справа), полученный запуском блока с кодом приведенного выше.

```
par(mfrow = c(1, 2))
matplot(S2, type = "l", col = c("blue", "red"))
plot(s1, type = "l", col = "black")
```



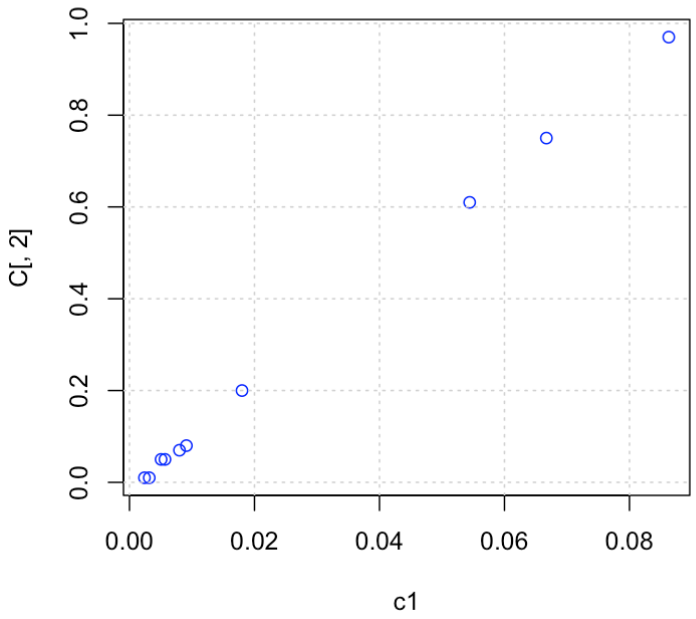
Очевидно, что полученная первая независимая компонента очень похожа на вторую исходную (ту, что показана красным цветом), ее пик совпадает с оригинальным пиком, но имеется артефакт в области пика первой компоненты. Тут следует сразу отметить, что порядок компонент в нашей исходной матрице $\mathbf{S}$ и порядок компонент, который мы получаем в результате АНК, не всегда совпадает.

Мы также можем получить и соответствующий вектор вкладов $\mathbf{c}_1$. Вспомним, что $\mathbf{w}_1$ это вклад первой компоненты в "отбеленный" сигнал. Соответственно, чтобы получить $\mathbf{c}_1$ нужно взять значения $\mathbf{w}_1$ и умножить их на матрицу, обратную "обеляющей", которую мы вычислили чуть выше:

```
c1 <- MWI %*% w1
```

Сравним их с исходными данными (как видно из кода, мы сравниваем их с концентрациями для второй исходной компоненты, так как порядок компонент в данном случае не совпадает):

```
plot(c1, C[, 2], col = "blue")
grid()
```



Как можно заметить, оценочные значения вкладов найденной компоненты в каждую из 10 смесей получились довольно близки к теоретическим, которые мы использовали для симуляции данных, однако, они примерно в 10 раз меньше. Это связано с тем, что матрица, обратная отбеливающей, делит все значения на квадратный корень из числа переменных (как раз 10). От этого эффекта можно избавится с помощью нормировки. Тем не менее, коэффициент корреляции между этими двумя векторами значений составляет 0.9998, что указывает на высокую точность декомпозиции.

Заметим также, что концентрации можно получить и с помощью проекции исходных данных на вектор $\mathbf{s}_1$

```
c1 <- X %*% s1 %*% solve(crossprod(s1))
```

что даст идентичные значения (запустите код и убедитесь в этом самостоятельно). В целом, результат получился очень неплохой, не смотря на использование самой простой версии алгоритма. Попробуем теперь найти следующую компоненту.

Нахождение второй компоненты

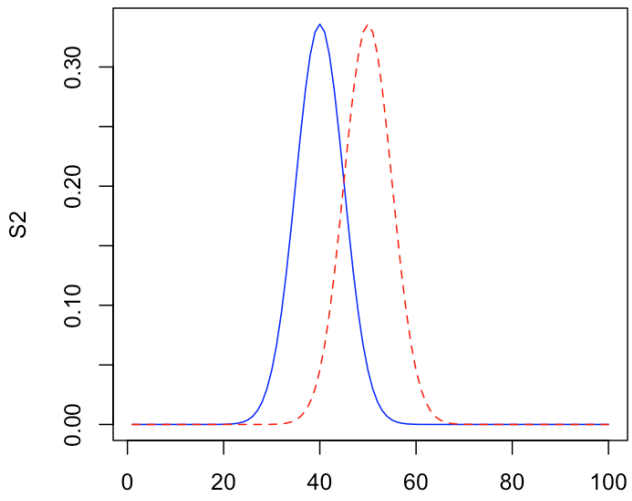
Для нахождения второй компоненты нужно использовать тот же самый подход с добавлением одного шага — нужно сделать вторую компоненту независимой от первой. Вот код, который реализует эту опцию.

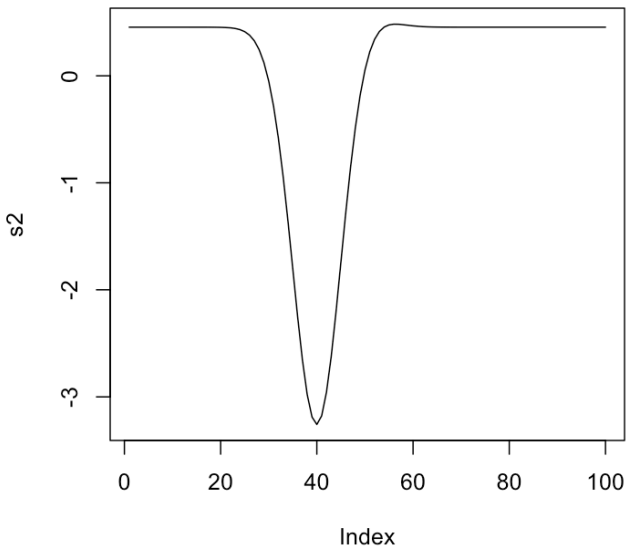
```r
set.seed(42)

w2 <- matrix(runif(nrow(Z)), ncol = 1)
w2 <- w2 / sqrt(sum(w2^2))

for (i in 1:10) {

   s2 <- crossprod(w2, Z)
   w2 <- (Z %*% t(s2^3)) / M - 3 * w2
   w2 <- w2 / sqrt(sum(w2^2))

   # дополнительная стока кода нужна чтобы сделать w2 независимой от w1
   w2 <- w2 - tcrossprod(w1) %*% w2
   w2 <- w2 / sqrt(sum(w2^2))
}



s2 <- crossprod(Z, w2)
```

Как вы можете заметить, мы используем `set.seed(42)` в начале этого и предыдущего блоков. Это нужно для того, чтобы начальное приближение для **w**, которое генерируется с помощью случайных чисел, было одно и тоже, иначе результат не всегда будет воспроизводимым. Например, компоненты могут случайно менять знак и тогда пики будут инвертированы. В реальных задачах это неважно, но для учебных примеров воспроизводимость результатов важна, поэтому мы добавили эту команду.

Можно видеть, что код для вычисления второй компоненты практически идентичен коду для нахождения первой компоненты, мы лишь добавили строку, разделяющую компоненты и дополнительную нормировку после этого. График ниже показывает результат.

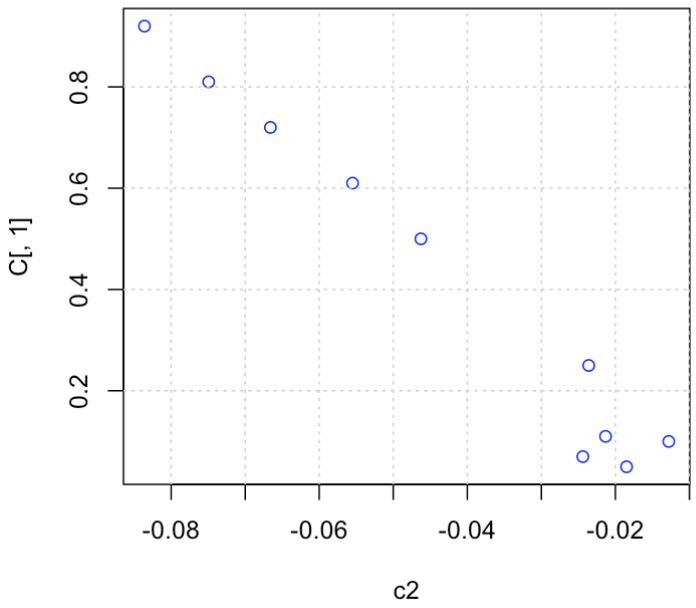
```r
par(mfrow = c(1, 2))
matplot(S2, type = "l", col = c("blue", "red"))
plot(s2, type = "l", col = "black")
```





Как можно заметить, в этом случае пик получился инвертированым, но его позиция и форма совпадают с первой теоретической компонентой. Вычислим оценки вкладов для этой компоненты и покажем их зависимость от теоретических значений:

```
c2 <- MWI %*% w2
plot(c2, C[, 1], col = "blue")
grid()
```



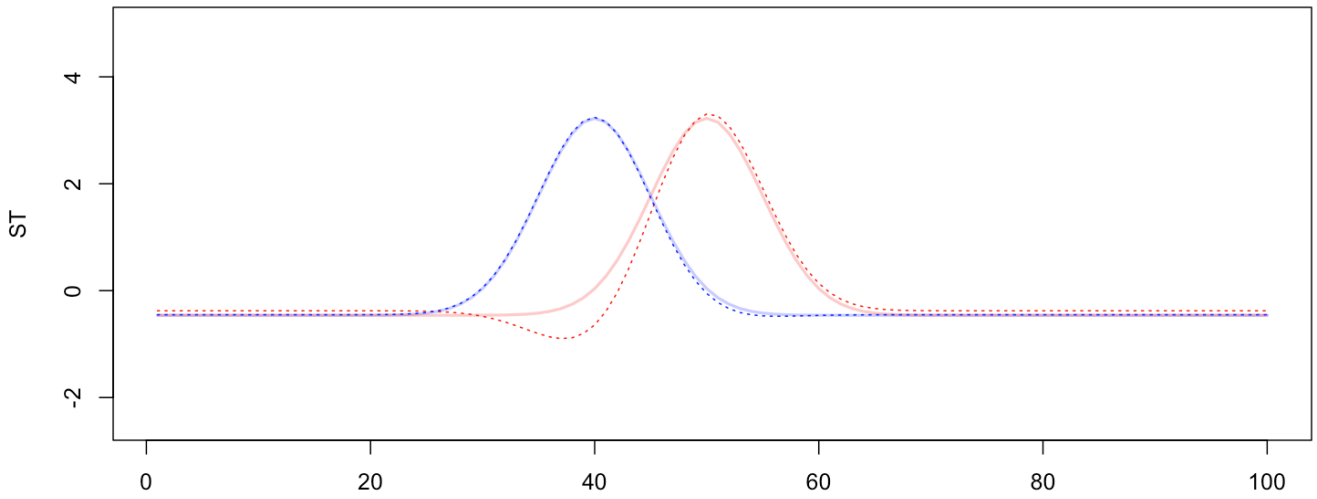
Как и ожидалось, значения получились отрицательными из-за инверсии пика, однако, абсолютные значения довольно близки к оригинальным, корреляция между ними для второй компоненты составляет -0.985, что чуть хуже, чем результат, полученной для первой компоненты.

Для сравнения теоретических и практических спектров необходимо сделать так, чтобы их базовые линии не были сдвинуты относительно друг друга, и пики имели примерно одинаковую магнитуду. Лучше всего для этого подходит, например, SNV трансформация, которая автошкалирует спектры (т.е. для каждого отдельного спектра вычитает среднее значение и делит его на стандартное отклонение).

Сделаем SNV трансформацию и исходных теоретических спектров, и тех, что получили с помощью АНК. При этом второй спектр домножим на −1, чтобы он был правильно ориентирован. После этого покажем их на одном графике с оригинальными спектрами для сравнения:

```
# делаем SNV трансформацию спектров, полученных с помощью АНК
se1 <- ( s1 - mean( s1)) / sd( s1)
se2 <- (-s2 - mean(-s2)) / sd(-s2)

# делаем SNV трансформацию теоретических спектров - столбцов матрицы S2
ST <- apply(S2, 2, function(x) (x - mean(x)) / sd(x))
```

```
# показываем исходные спектры толстыми полупрозрачными кривыми
matplot(ST, type = "l", col = c("#0000ff40", "#ff000040"),
   ylim = c(-2.5, 5), lty = 1, lwd = 2)

# показываем спектры, полученные с помощью АНК
# здесь мы меняем местами цвета, так как они имеют другой порядок
lines(se1, col = "red", lty = 3)
lines(se2, col = "blue", lty = 3)
```



Как вы можете заметить, качественно результат получился вполне приемлемым — оба пика удалось хорошо разделить и, в целом, спектры довольно хорошо совпадают.

Реализация полного алгоритма

В заключение этого раздела приведем код для всего алгоритма, который можно применять к любым данным с любым числом компонент, и покажем его работу на другом примере. Для начала соберем код, написанный выше и оформим его в виде функции, похожей на ту, что мы сделали для алгортма NIPALS:

```
fastica <- function(X, A, min.eigvalue = 10^-9, niter = 100) {
```

```
# Отбеливаем и шкалируем исходные данные
m <- svd(scale(t(X), center = TRUE, scale = FALSE))
Z <- t(m$u[, 1:A, drop = FALSE])
Z <- Z * sqrt(ncol(X))

# формируем "отбеливающую матрицу" и обратную к ней
MW <- t(m$v[, 1:A, drop = FALSE] %*% diag(1 / m$d[1:A]) )
MWI <- t(diag(m$d[1:A]) %*% t(m$v[, 1:A]))

# готовим матрицу для всех векторов w
W <- matrix(0, A, A);
M <- ncol(Z)
for (a in 1:A) {

   # генерируем начальные значения для текущего w
   w <- matrix(runif(A), ncol = 1)
   w <- w / sqrt(sum(w^2))

   for (i in 1:niter) {
      # делаем следующее приближение на основе градиента
      w <- (Z %*% (  (t(Z) %*% w)^3  )) / M - 3 * w
      w <- w / sqrt(sum(w^2))
      # убираем зависимость от предыдущих компонент
      w <- w - tcrossprod(W) %*% w
      w <- w / sqrt(sum(w^2))
   }

   W[, a] <- w
}

# вычисляем матрицу с чистыми спектрами
Sest <- crossprod(Z, W)

# вычисляем матрицу с вкладами
Cest <- MWI %*% W
```

```
    # возвращаем все результаты
    return (list(C = Cest, S = Sest, W = W, MW = MW, MWI = MWI))
}
```

Как вы можете заметить, помимо собственно оценок чистых спектров и их вкладов, метод также возвращает промежуточные результаты — вклады компонент в отбеленные данные, а так же отбеливающую матрицу и обратную к ней. Они нам понадобятся чуть ниже, когда мы будем обсуждать оценку качества алгоритма.

6.4 Альтернативные алгоритмы АНК

Существует достоточно большое число реализаций АНК, которые используют различные подходы для достижения критериев независимости и “негауссовости” разделенных сигналов. В том числе и подробно рассмотренный нами выше алгоритм *FastICA* имеет несколько реализаций, которые отличаются друг от друга функцией, с помощью которой вычисляются приближения для вектора **w**. В нашей реализации мы использовали функцию вычисления коэффициента эксцесса (и его производной), однако, похожего, а иногда и лучшего эффекта можно добиться с помощью других функций (например, через вычисление так называемой *негэнтропии*). Кроме того, существует версия этого алгоритма, позволяющая вычислить все компоненты за раз, вместо вычисления их друг за другом, как в нашей реализации. Этот алгоритм со всеми нюансами реализован в одноименном пакете для R, *FastICA*.

Ближайшими альтернативами алгоритму *FastICA* являются алгоритмы *InfoMax* и *JADE*. Так, алгоритм *InfoMax* является разновидностью другого известного метода разложения на компоненты — поиска наилучшей проекции (projection pursuit), при этом он использует критерий наибольшего правдоподобия для того, чтобы обеспечить независимость сигналов. Сущетствуют также алгоритмы АНК, основанные на использовании ядер (*Kernel ICA*), а также с помощью вычисления плотностей вероятности (*ProDenICA*). Последний также имеет реализацию для R в одноименном пакете.

В начале 2000-х годов этот метод стал постепенно использоваться и для анализа данных в аналитической химии, в том числе и для разделения спектров. Соответственно, появились алгоритмы, которые учитывают особенности спектральных данных, в частности, тот факт, что спектры чистых компонент не всегда полностью статистически независимы, а также неотрицательность спектральных значений. Из числа последних стоит упомянуть *MILCA* и *SNICA*.

Алгоритм *MILCA* (Mutual Information Least Dependent Component Analysis) основан на поиске наименее зависимых (в отличие от независимых) компонент смесей на основе минимизации численных значений взаимной информации, как меры зависимости сигналов. В свою очередь *SNICA* (Stochastic Non-Negative Independent Component Analysis), использует такой же критерий и, одновременно, накладывает условие неотрицательности получаемых сигналов, что как раз и характерно для спектров. Кроме того, SNICA использует метод Монте-Карло для минимизации информации, что дает дополнительные преимущества в контроле за точностью решения в ходе итераций, а также в применении множества эмпирических

трюков в поиске глобального минимума. Использование условия неотрицательности в тандеме с гипотезой наименьшей зависимости «чистых» компонент позволило исключить необходимость МГК декорреляции в *SNICA* и находить решения в случае частично зависимых компонент.

6.5 Анализ результатов декомпозиции

6.5.1 Индекс подобия

Как мы не раз отмечали, чистые спектры, выделенные с помощью АНК, абстрактны и подлежат идентификации, которая заключается в сравнении смоделированного сигнала с эталоном. На практике это может быть, например, поиск спектров в библиотеке, которые имеют те же самые пики, или другие особенности формы, что и полученные с помощью АНК.

Исторически первым методом сравнения спектров было их визуальное сопоставление, как мы делали с помощью графиков выше. Однако, если требуется сравнить несколько десятков, или даже сотен вариантов, визуальное сравнение может быть довольно утомительным, поэтому необходима какая-то численная величина, которая позволит отфильтровать наименее похожие спектры. Такая величина в общем случае называется *индексом подобия*.

Одним из самых простых индексов является коэффициент корреляции Пирсона, подробно рассмотренный нами в главе 2. Рисунок 6.4 показывает результаты работы *FastICA* для двух вариантов примера с двумя пиками, расположенными на разном расстоянии друг от друга. На каждом графике вы можете видеть спектры смесей, показанные светло-серым цветом, спектры чистых компонент и их оценки, полученные с помощью АНК. Кроме этого, на графиках показаны коэффициенты корреляции, посчитанные для каждой пары спектров.

Как можно видеть, даже для первого случая с сильно перекрывающимися пиками, корреляция между исходными и разрешенными спектрами довольно высокая ($r > 0.90$). В втором случае она еще выше. Ну и там, где разрешенные пики получились “перевернутыми”, соответствующий коэффициент корреляции негативный.

Следует отметить, однако, что сравнение с эталонным сигналом может быть затруднено, если эти сигналы были получены на разных устройствах (например, с разным спектральным разрешением — шагом между двумя соседними точками спектра), или в разных условиях (например, состояние вещества в процессе съемки).

6.5.2 Расстояние Амари

Для того, чтобы понять, насколько хорошо оценки вкладов (концентраций), полученные с помощью АНК, соответствуют реальным вкладам, можно посчитать расстояние Амари (иногда его также называют индексом Амари). Для этого сравниваются две матрицы вкладов, но не для оригинальных смесей $\mathbf{C}$ а для обеленных составляющих $\mathbf{W}$. Матрица $\mathbf{W}$, полученная с помощью АНК алогритма нам известна, она

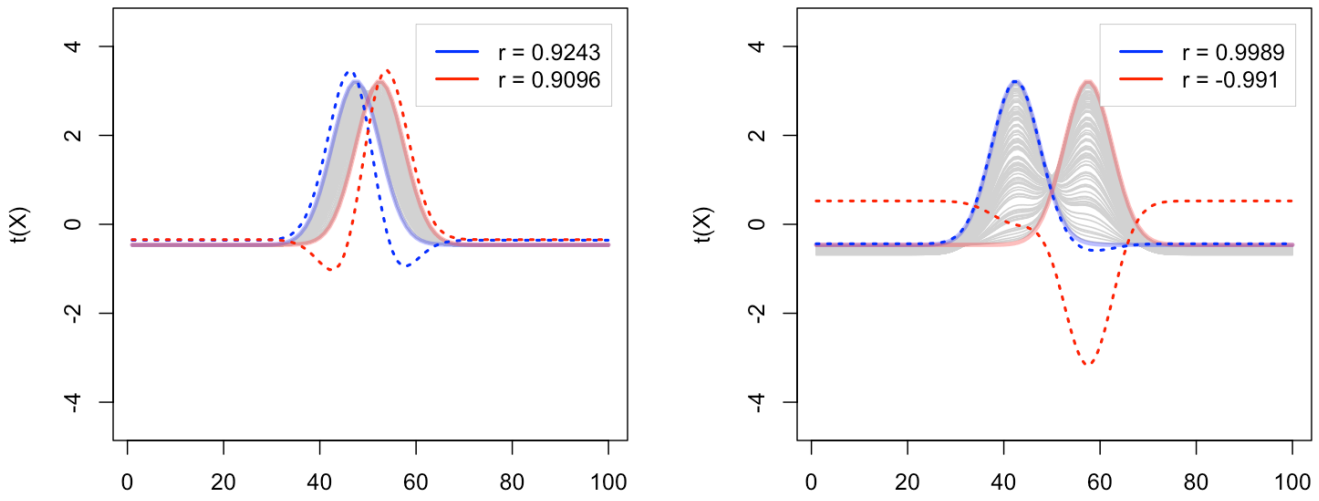


Рис. 6.4. Исходные и разделенные спектры, и коэффиент корреляции Пирсона, посчитанный для каждой пары таких спектров.

находится среди других результатов, которые возвращает наша функция, а вот ее теоретические значение нужно вычислить.

Для этого нужно просто умножить матрицу теоретических концентраций слева на обеляющую матрицу:

$$\mathbf{W}_0 = \mathbf{M}_W \mathbf{C}^{\mathrm{T}}$$

Теперь мы можем вычислить расстояние Амари между $\mathbf{W}$ и $\mathbf{W}_0$. Заметим, что обе матрицы квадратные, размером $A \times A$, где A — это число независимых компонент. Для вычисления расстояния используется следующее соотношение:

$$d(\mathbf{W}, \mathbf{W0}) = \frac{1}{2A} \sum_{i=1}^{A} (\sum_{j=1}^{A} \frac{|p_{ij}|}{max_j |p_{ij}|} - 1) + \frac{1}{2A} \sum_{j=1}^{A} (\sum_{i=1}^{A} \frac{|p_{ij}|}{max_i |p_{ij}|} - 1)$$

Здесь p_{ij} — это элементы матрицы $\mathbf{P} = \mathbf{W}_0 \mathbf{W}^{-1}$.

По сути вычисления сводятся к следующему набору шагов:

1. Найти матрицу $\mathbf{P}$, такую, что $\mathbf{PW} = \mathbf{W}_0$.
2. Избавиться от знаков, взяв абсолютные значения в матрице $\mathbf{P}$.

3. Посчитать суммы значений в каждом столбце, разделенные на максимальное значение в этом столбце и вычесть единицу из этой суммы
4. Посчитать суммы значений в каждой строке, разделенные на максимальное значение в этой строке и вычесть единицу из этой суммы
5. Сложить результаты предыдущих двух шагов и разделить на $2A$

Индекс Амари косвенно связан с корреляцией между найденными и теоретическими значениями вкладов (концентраций). При этом он не чувствителен к порядку компонент, т.е. если поменять столбцы с вкладами местами в одной из матриц, то индекс не поменяется. Тоже самое случится, если значения в одном из столбцов какой-то матрицы умножить, или поделить на постоянное значение.

Нулевое значение индекса означает, что матрицы вкладов идентичны с точностью до знака и постоянных множителей. Другими словами, значения из стоблцов матрицы с оценками вкладов линейно зависят от значений в столбцах матрицы с теоретическими вкладами. Это идеальный результат для АНК. Чем выше значение индекса, тем хуже соответствие между теоретическими вкладами и их оценкой.

Повторим один из примеров, которые мы рассматривали выше — сформируем спектры смесей двух симулированных компонент с частично перекрывающимися пиками, и применим АНК для декомпозиции смесей:

```
set.seed(42)

# начальные условия
w <- 1:100
mu <- 50
sigma <- 5
n <- 50

# устанваливаем расстояние между пиками
z <- 2

# генерируем чистые спектры в виде гауссианы
s2 <- dnorm(w, mu + sigma * z/2 , sigma)
s1 <- dnorm(w, mu - sigma * z/2, sigma)

# нормируем их на единичную длину и соединяем в матрицу S
s1 <- s1 / sqrt(sum(s1^2))
s2 <- s2 / sqrt(sum(s2^2))
S <- cbind(s1, s2)
```

```
# случайным образом генерируем значения для матрицы вкладов (100 x 2)
C <- matrix(runif(n * 2), nrow = n, ncol = 2)

# вычисляем спектры смесей с помощью закона Ламберта-Бера
X <- tcrossprod(C, S)

# используем нашу реализацию fastICA для разделения спектров
res <- fastica(X, 2)
```

Теперь посчитаем индекс Амари для результата:

```
# вычисляем W0 и P
W0 <- res$MW %*% C
P <- abs(solve(res$W, W0))
A <- ncol(P)

# вычисляем сумму столбцов
csum <- apply(P, 2, sum)
# вычисляем максимальные значения в каждом столбце
cmax <- apply(P, 2, max)
# вычисляем сумму строк
rsum <- apply(P, 1, sum)
# вычисляем максимальные значения в каждой строке
rmax <- apply(P, 1, max)

# суммируем оба результата и делим на 2A
d <- (sum(csum/cmax - 1) + sum(rsum/rmax - 1)) / (2 * A)
print(d)
```

```
[1] 0.1174623
```

Индекс равен 0.117, что немного выше 0. Давайте посчитаем матрицу взаимных корреляций между теоретическими вкладами и их оценками, полученными с помощью АНК.

```
# показываем взаимную корреляцию
show(cor(C, res$C))
```

```
          [,1]       [,2]
[1,] 0.99966314 -0.2669919
[2,] 0.08318173 -0.9774119
```

Как можно видеть, результат выглядит очень хорошо для первой разрешенной компоненты (первый столбец в матрице) и чуть хуже для второй. Попробуйте теперь увеличить расстояние между пиками, установив значение $z = 6$, и снова запустите два блока с кодом выше. В этом случае индекс Амари будет немного ниже (0.109), и корреляционная матрица будет чуть более сбалансированной, так как результат декомпозиции окажется чуть более точным.

В заключение оформим вычисление расстояния Амари в виде отдельной функции, которая принимает в качестве аргументов список с результатами, которые возвращает написанная нами функция `fastica()` и матрицу с теоретическими концентрациями, а возвращает расстояние Амари.

```
amaridistance <- function(res, C0) {

   # вычисляем W0 и P
   W0 <- res$MW %*% C0
   P <- abs(solve(res1$W, W0))
   A <- ncol(P)

   # вычисляем сумму столбцов
   csum <- apply(P, 2, sum)
   # вычисляем максимальные значения в каждом столбце
   cmax <- apply(P, 2, max)
   # вычисляем сумму строк
   rsum <- apply(P, 1, sum)
   # вычисляем максимальные значения в каждой строке
   rmax <- apply(P, 1, max)
   # суммируем оба результата и делим на 2A
   d <- (sum(csum/cmax - 1) + sum(rsum/rmax - 1)) / (2 * A)


   # суммируем оба результата и делим на 2A
   d <- (sum(csum/cmax - 1) + sum(rsum/rmax - 1)) / (2 * A)
   return (d)
}
```

6.5.3 Расчет реальных значений концентраций

Как и в случае спектров, значения концентраций отдельных компонент, полученных в результате разделения спектров, не соответствуют количественно реальным концентрациям. Для того, чтобы получить реальные значения, необходимо построить регрессионную модель, например, с помощью ПЛС регрессии, которую мы рассмотрели подробно в главе 4.

Однако, это можно сделать и на основе результатов АНК. Для этого необходимо приготовить несколько стандартных смесей с известными концентрациями компонент, и получить их спектры, используя тот же прибор и те же условия, что и для спектров смесей с неизвестными концентрациями. Далее нужно объединить все спектры вместе и использовать АНК для их разделения на чистые компоненты — т.е. получить матрицы $\hat{\mathbf{C}}$, и $\hat{\mathbf{S}}$, которые будут оценками исходных матриц $\mathbf{C}$, и $\mathbf{S}$.

После этого нужно выделить те строки матрицы $\hat{\mathbf{C}}$, которые соответствуют стандартным смесям с известными концентрациями, и построить регрессионные модели, которые для каждой компоненты позволят предсказать реальные значения из $\mathbf{C}$ по значениям, полученным в результате АНК, $\hat{\mathbf{C}}$. Каждая модель будет представлять собой простую линейную регрессию с коэффициентами наклона и сдвига.

6.6 Практический пример

В заключительном разделе мы решим практический пример, основанный на анализе реальных, а не симулированных спектров. Для этого вам понадобятся два набора данных из файлов `SimdataS.csv` и `Simdata.csv`, которые вы можете скачать из репозитория книги на GitHub, где лежат и остальные данные.

Первый файл содержит чистые UV/Vis спектры трех углеводородов, которые мы для краткости обозначим как C1, C2, и C3. Код ниже показывает, как считать эти спектры и показать их графически. Для визуализации большинства графиков мы будем использовать метод `mdaplotg` из пакета `mdatools`, так как он позволяет строить подобные графики быстрее и с меньшим количеством кода. Однако, вы можете продолжить использовать встроенный метод `matplot`, как в примерах выше.

```
library(mdatools)

S <- read.csv("SimdataS.csv")
S <- as.matrix(S)

mdaplotg(t(S), type = "l")
```

Как вы можете заметить из графика, спектры для первых двух составляющих очень похожи, мы можем ожидать, что их вклад будет довольно трудно разделить. Третий спектр выглядит вполне различимым на фоне других.

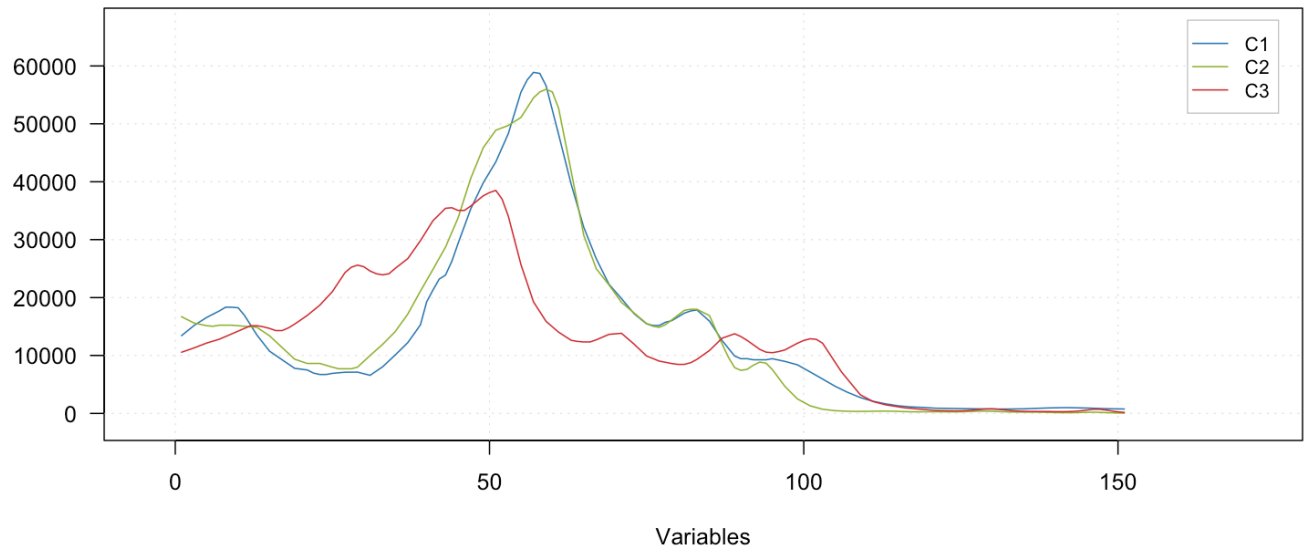


Рис. 6.5. UV/Vis спектры чистых компонент из набора *Simdata*.

Во втором файле находятся спектры смесей этих трех углеводородов с разными концентрациями. Концентрации варьировались случайным образом, но подбирались так, чтобы концентрация $C1$ имела наибольший разброс (от 0 до 1M), концентрация $C2$ имела разброс в два раза меньше (от 0 до 0.5M) а третья составляющая $C3$ имела наименьший разброс концентраций (от 0 до 0.1M). Другими словами, ее вклад будет трудно распознать, не смотря на хорошо различимый от других составляющих спектральный профиль.

Давайте считаем эти данные и покажем их также в виде графика. Этот набор устроен немного сложнее. В первом столбце находятся идентификаторы смесей, это текстовая информация. Далее идут три столбца с теоретическими концентрациями компонент. И только начиная с 5-го столбца — спектральные данные. Учтем это, когда будем считывать данные, как показано ниже.

```
# читаем данные из файла
d <- read.csv("Simdata.csv")

# столбцы со второго по четвертый содержат теоретические концентрации
C <- as.matrix(d[, 2:4])

# остальные столбцы содержат спектры смесей
X <- as.matrix(d[, 5:155])
```

```
# покажем графики спектров раскрасив их в соответствие с содержанием C1
mdaplot(X, type = "l", cgroup = C[, 1])
```

Рисунок 6.6 показывает спектры в виде трех графиков. На каждом графике цвет спектра отражает вклад одной из трех составляющих. Хорошо заметно, что вклад C1 сильно влияет на общий вид спектра, тогда как определенного визуального паттерна от вклада C3 не заметно.

Применим АНК к исходным спектрам и изучим результат. Для начала изучим расстояние Амари и коэффициенты корреляции между оценками, полученными с помощью алгоритма, и теоретическими спектрами.

```
set.seed(42)
res1 <- fastica(X, 3)
show(amaridistance(res1, C))
```

```
[1] 0.8614658
```

```
show(cor(res1$C, C))
```

```
            C1         C2          C3
[1,]  0.8935223  0.3822575 -0.04340021
[2,] -0.8069169 -0.5357928  0.06358499
[3,]  0.8254368  0.4888538  0.05975395
```

Очевидно, что результат получился неприемлемым, особенно для второй и третьей составляющих. Основная причина такого плохого результата заключается в похожести спектров. Давайте посчитаем взаимную корреляцию между ними:

```
# расчет попарной корреляции между спектрами
show(cor(S))
```

```
          C1        C2        C3
C1 1.0000000 0.9859073 0.6229912
C2 0.9859073 1.0000000 0.6887351
C3 0.6229912 0.6887351 1.0000000
```

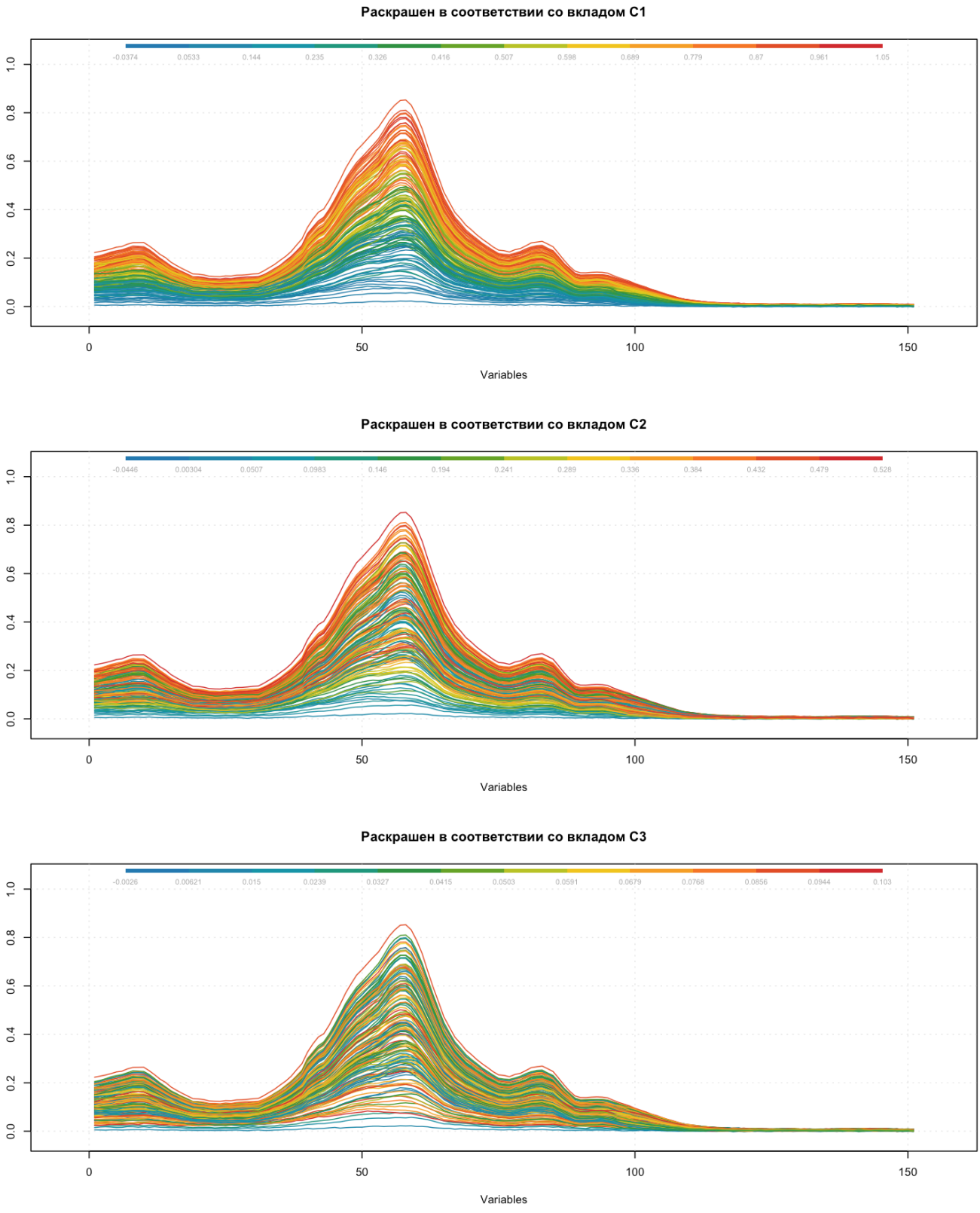


Рис. 6.6. UV/Vis спектры смесей из набора *Simdata.*

Как можно видеть из результата, спектры очень сильно скоррелированы между собой, что делает работу АНК — по крайней мере той версии алгоритма, которая основана на их независимости — затруднительной.

Однако, выход, на самом деле, есть. Дело в том, что АНК, как и МГК, можно применять как строкам, так и к столбцам матрицы **X**. Вспомним, как выглядит исходное соотношение без учета матрицы с шумом.

$$\mathbf{X} = \mathbf{C}\mathbf{S}^{\mathrm{T}}$$

Что если мы поменяем строки и столбцы в исходной матрице местами, т.е. транспонируем ее? Зная линейную алгебру, можно вывести результат следующим образом:

$$\mathbf{X}^{\mathrm{T}} = (\mathbf{C}\mathbf{S}^{\mathrm{T}})^{\mathrm{T}} = \mathbf{S}\mathbf{C}^{\mathrm{T}}$$

Другими словами, мы можем примент АНК для разделения столбцов матрицы **X** на независимые сигналы, т.е. считая независимыми концентрационные профили, а не спектры. И, так как в этом случае концентрации компонент действительно не зависят друг от друга, этот трюк сработает! Нужно лишь не забыть, что в этом случае матрица **C** будет содержать разделенные спектры чистых компонент, а матрица **S** — их концентрации.

Попробуем это сделать:

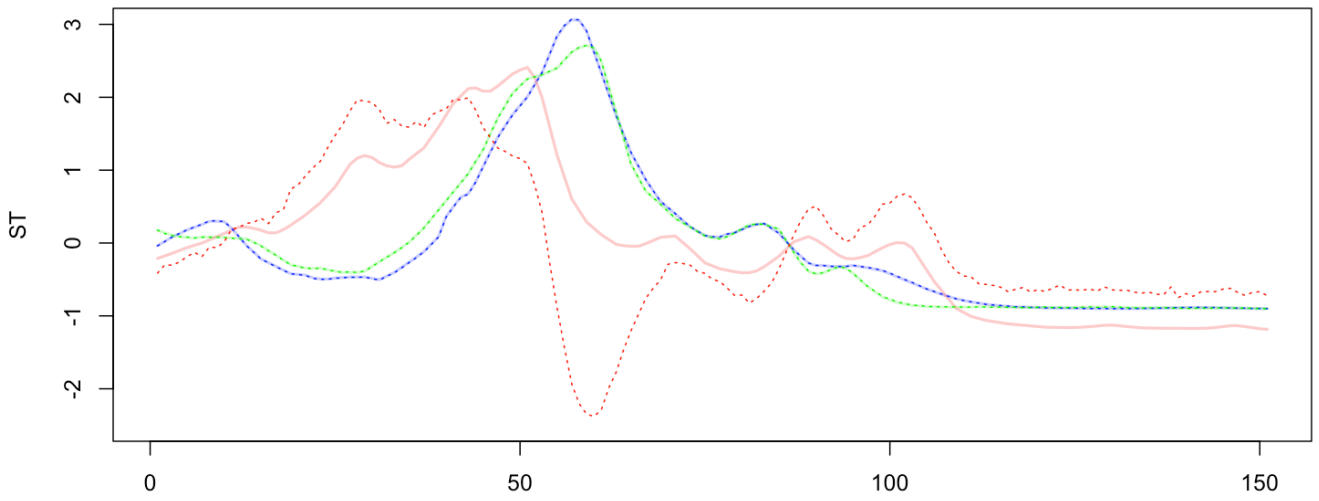
```
set.seed(42)

# применяем АНК к транспонированной матрице X
res2 <- fastica(t(X), 3)

# показываем корреляцию для концентраций
show(cor(res2$S, C))
```

```
              C1          C2          C3
[1,]  0.985640055 -0.03631737 -0.01303268
[2,]  0.044924958  0.05500210 -0.98206842
[3,] -0.003540341  0.98318630 -0.00809024
```

Результат очень обнадеживает. Можно видеть, что вторая и третья компоненты поменялись местами и что корреляция для третьей компоненты отрицательная, т.е. полученные спектр скорее всего инвертирован. .

Построим теперь графики разрешенных спектров. Из таблицы корреляций нам видно, что творой и третий спектры нужно поменять местами, а третий еще и инвертировать. Кроме этого, давайте применим SNV трансформацию и к исходным, и к разрешенным спектрам, чтобы удобнее было сравнивать их визуально.

```
# применяем SNV к теоретическим спектрам
ST <- apply(S, 2, function(x) (x - mean(x)) / sd(x))

# сохраняем спектры полученные с помощью АНК и меняем компоненты местами
SE <- res2$C[, (c(1, 3, 2))]

# инвертируем третий спектр умножая его на -1
SE[, 3] <- -SE[, 3]

# применяем SNV
SE <- apply(SE, 2, function(x) (x - mean(x)) / sd(x))

# показываем спектры попарно на графиках
matplot(ST, type = "l", col = c("#0000ff40", "#00ff0040", "#ff000040"),
   lwd = 2, lty = 1, ylim = c(-2.5, 3))
matplot(SE, type = "l", col = c("blue", "green", "red"), lwd = 1,
   lty = 3, add = TRUE)
```



Результат получился практически идеальным для первых двух компонент. Для третьей компоненты восстановленный спектр далек от исходного, однако он сохраняет ключевые пики, что и позволило добиться хорошей корреляции между оценочными и реальными значениями вкладов.

Надо сказать, что трюк с заменой строк и столбцов в исходной матрице не единственный, который позволяет улучшить результаты АНК декомпозиции. Вы также можете попробовать сделать предварительную обработку, например, для БИК и других спектров с широкими пиками часто хорошо работает фильтр Савицкого-Голея с первой, или второй производной.

Без наличия референтных спектров и концентраций оценить результат АНК конечно труднее, для этого потребуется хорошее знание химии и, возможно, физики изучаемого процесса, реакции, или явления. Тот результат, который можно надежно интепретировать с учетом этих знаний и будет искомым.

Заключение

В этом очень коротком заключении хочется немного рассказать о планах. Разделы хемометрики, которые рассмотрены в текущей версии книги, дают представление лишь о минимальном наборе методов и подходов, необходимых для погружения в предмет. Хемометрика является активно развивающейся научной дисциплиной и огромное количество интересных тем осталось за рамками этой версии книги. Это и методы многомерного разрешения кривых, и способы дизайна многокомпонентных градуировочных смесей, и параллельный факторный анализ (PARAFAC), и обработка гиперспектральных изображений, и много-много других важных и крайне интересных направлений. По нашей задумке, эта книга будет постепенно дополняться новыми главами, благо интернет-формат публикации учебника позволяет это делать с минимальной затратой сил и ресурсов. Оставайтесь с нами! Впереди много интересного!