МИНОБРНАУКИ РОССИИ

Санкт-Петербургский государственный
электротехнический университет «ЛЭТИ»

ОСНОВНЫЕ АЛГОРИТМЫ ЧИСЛЕННОГО АНАЛИЗА.

ИСПОЛЬЗОВАНИЕ ПАКЕТА R(S-PLUS) ДЛЯ АНАЛИЗА СТАТИСТИЧЕСКИХ ДАННЫХ

Санкт-Петербург
Издательство СПбГЭТУ «ЛЭТИ»
2011

МИНОБРНАУКИ РОССИИ

Санкт-Петербургский государственный
электротехнический университет «ЛЭТИ»

ОСНОВНЫЕ АЛГОРИТМЫ ЧИСЛЕННОГО АНАЛИЗА.

ИСПОЛЬЗОВАНИЕ ПАКЕТА R(S-PLUS) ДЛЯ АНАЛИЗА СТАТИСТИЧЕСКИХ ДАННЫХ

Методические указания к практическим занятиям
по дисциплине «Вычислительная математика»

Санкт-Петербург
Издательство СПбГЭТУ «ЛЭТИ»
2011

УДК 519.2

Основные алгоритмы численного анализа. Использование пакета R(S-PLUS) для анализа статистических данных: Методические указания к практическим занятиям по дисциплине «Вычислительная математика» / Сост: А. И. Коробейников, С. В. Малов, И. В. Матвеева. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2011. 40 с.

Содержат описание основных методов программирования в пакете R (S-PLUS) и его использования для обработки статистических данных.

Предназначены для студентов ФКТИ.

Утверждено
редакционно-издательским советом университета
в качестве методических указаний

Введение

Пакет **R** — прежде всего язык программирования для статистической обработки данных и работы с графикой, но в то же время это свободная программная среда с открытым исходным кодом. **R** возник как свободный аналог среды S-PLUS, которая в свою очередь является коммерческой реализацией языка расчетов S. Существует центральная система хранения и распространения пакетов – CRAN (Comprehensive R Archive Network, http://cran.r-project.org/), состоящая из стабильной базы и множества дополнений (пакетов). **R** – один из мощнейших инструментов, предназначенных для обработки статистических данных любой структуры.

Способ установки **R** зависит от операционной системы. Для Windows базовый пакет скачивается с удобного для пользователя зеркала CRAN и устанавливается стандартным образом (для других ОС см. инструкции по установке на сайте **R**). После запуска **R** появится приглашение в виде символа «приглашение к работе» `>`. **R** общается с пользователем в режиме командной строки (консоли). Отметим, что основные мощности и специальные методы запрограммированы в дополнительных пакетах, загружаемых по мере необходимости. Малая часть пакетов устанавливается непосредственно с самой программой, для установки других пакетов требуется снова войти на сайт, выбрать раздел `packages` и установить требуемый пакет. На текущий момент более 2500 пакетов можно установить с официального сайта **R**. Чтобы иметь возможность пользоваться пакетом, требуется загрузить его с помощью команды

```
> library(<название пакета>)
```

или с использованием соответствующего пункта меню. Если при выходе из **R** сохранить рабочую среду, т. е. бинарный файл `.RData`, содержащий объекты, созданные за время сессии, и текстовый файл `.Rhistory` – полную историю введенных команд, то при следующем старте они загрузятся автоматически.

Наконец, отметим, что **R** является объектно-ориентированным языком программирования. Соответствующую информацию можно получить из раздела справки `Methods`.

1. Основы программирования в R

Технически **R** является языком выражений с очень простым синтаксисом. Существуют различные способы ввода данных в **R** и сохранения информации. Рассмотрим некоторые популярные примеры.[1]

[1] Для получения более подробных сведений см. руководство «Импорт/экспорт данных в R», поставляемое с базовым пакетом.

Ввод команды. Команды состоят из выражений, разделяемых точкой с запятой («;») или переводом строки. При работе **R** учитывает регистр. Простые команды могут быть сгруппированы в составное выражение фигурными скобками («{» и «}»). Комментарии начинаются с символа решетки #. Если команда не завершена в конце строки, **R** выдаст приглашение «+» во второй и последующих строках. Командные строки, набираемые в консоли, ограничены в размере до 1024 символов.

Ввод программы (скрипта). Вводить программу (скрипт) можно непосредственно с клавиатуры (в консоль или с использованием встроенного редактора **R**) или загрузить готовую программу из файла с помощью меню «Файл» или с использованием команды

```
> source("имя файла с учетом расширения", echo=TRUE).
```

Если файл находится не в текущей директории, то требуется указать полное имя файла с учетом пути и расширения. При подготовке программы для последующей загрузки в **R** можно использовать любой текстовый редактор. Встроенный редактор **R** позволяет выделять и запускать любые фрагменты программы, что делает работу с ним более удобной, чем с каким-либо иным текстовым редактором. Отметим также, что существуют специализированные программы-оболочки для **R**, предоставляющие расширенные возможности редактирования и запуска скриптов. Под Windows отдельно выделим такие оболочки, как Tinn-R, SciViews-K и Revolution-R (последняя бесплатна только для академического использования).

Стандартное расширение файла с программой `.r`. Также можно сохранить в виде скрипта историю команд текущей сессии, например:

```
>savehistory(file="myscript.r")
```

В итоге в текущей папке появляется файл `myscript.r`, который можно редактировать и в дальнейшем загрузить в **R**.

1.1. Базовые объекты языка R

Любой объект языка **R** имеет набор атрибутов (*attributes*). Этот набор может быть разным для объектов разного вида, но каждый объект обязательно имеет 2 встроенных атрибута: Длина (length) и Тип (mode). Атрибут «длина» определяет число элементов объекта. Тип объекта – более сложная сущность. Например, тип объекта vector определяется типом элементов, из которых он состоит.

Vector. Вектор является простейшим объектом, объединяющим элементы одного примитивного типа. Создать пустой вектор можно при помощи функции `vector`:

```
v <- vector(mode = "logical", length = 0)
```

Аргумент `length` задает количество элементов вектора (размерность),[2] а `mode` – примитивный тип элементов.

Объекты примитивного типа *numeric* служат для представления чисел с плавающей точкой. Кроме «обычных» они могут принимать ряд специальных значений: `Inf` – бесконечность (положительная или отрицательная); `NaN` (*Not-A-Number*) – для отображения результатов операций, когда результат не определен, например деление нуля на ноль; `NA` (*Not Available*) – для отображения «пропусков» в данных.[3] Для объектов типа *numeric* определены обычные бинарные операции `+`, `-`, `*`, `/` (сложения, вычитания, умножения и деления соответственно), а также операции `abs()` взятия модуля, `^` – возведения в степень, `%%` – вычисления остатка от деления и `%/%` – целочисленного деления.

Объекты типа *complex* содержат комплексные числа. Мнимая часть комплексного числа записывается с символом `i` на конце. Примером комплексного вектора длины 4 может служить

```
c(5.0i, -1.3+8.73i, 2.0, 1+1i)
```

Комплексные векторы могут принимать те же специальные значения, что и числовые. Для комплексных чисел также реализованы 4 основные операции `+`, `-`, `*` и `/`, а также возведение в степень `^`, модуль `abs()` и др.

Объекты типа *logical* могут принимать одно из трех значений: `TRUE`, `FALSE` и `NA`. Кроме того, существуют (в целях обратной совместимости) глобальные переменные `T` и `F`, имеющие значение `TRUE` и `FALSE` соответственно. Для объектов типа *logical* определены обычные операции `!`, `&`, `|` – логического отрицания, логического «и» и логического «или». Также реализованы операции сравнения `x==y` и «исключающее ИЛИ» `xor(x,y)`.

Каждый элемент вектора типа *character* является строкой. Строка может быть произвольной длины. Для объектов строкового типа не существует пропущенного значения, в этой роли используется пустая строка.

Для векторов, составленных из значений определенного примитивного типа, определены все соответствующие операции покомпонентно. В случае бинарных операций с векторами различной длины выполняется «переписывание» (недостающие значения вектора меньшей длины дописываются циклически). Сформулируем *правила переписывания* (recycling rules):

1. Длина результата совпадает с длиной операнда наибольшей длины.
2. Если длина операнда меньшей длины является делителем длины второго операнда, то меньший операнд повторяется нужное количество раз до достижения длины второго операнда.

[2] В дальнейшем используем термин «длина», общий для всех объектов.

[3] Значение `NA` могут принимать объекты любого примитивного типа.

3. Если длина операнда меньшей длины не является делителем длины второго операнда, то меньший операнд повторяется нужное число раз до перекрытия длины второго. Лишние элементы отбрасываются, операция выполняется с выводом предупреждения.

В частности, если `a` – число, а `X` – вектор, то умножение `a*X` равносильно умножению каждой координаты вектора `X` на `a`.

В языке **R** реализован ряд специфических операций для векторов: команда `a%*%b` выдает скалярное (внутреннее) произведение векторов `a` и `b` одинаковой длины[4], команда `a%o%b` выдает внешнее произведение векторов `a` и `b`. Например, для вычисления длины вектора `x` в ортонормированном базисе можно ввести `sqrt(x%*%x)`.

Для доступа к элементам вектора используется оператор `[` с положительным, отрицательным или логическим аргументом. Если аргумент — положительный вектор индексов, то извлекаются элементы с индексами, соответствующими элементам вектора. Если значения аргумента отрицательные, результат содержит все элементы, кроме указанных. Типичным примером оператора `[` с логическим аргументом является конструкция вида `v[v<0]`, извлекающая из вектора `v` только отрицательные элементы. Нулевые, повторяющиеся и индексы, большие длины вектора, игнорируются. Пропущенные значения в индексах приводят к ошибке, как и смесь положительных и отрицательных индексов. Длина аргумента должна совпадать с длиной вектора, иначе аргумент переписывается нужное число раз. Извлекаются те элементы вектора, которым соответствует значение `TRUE` аргумента.

Для замены элементов вектора используют оператор присваивания `<-` (или `=`). Отметим, что длина результата равна максимальному индексу. Элементы, которым не присвоено значение, получают значение `NA`:

```
> v <- 1:5; v[10] <- 10; v
 [1]  1  2  3  4  5 NA NA NA NA 10
```

List. Список является объектом типа *list*, его длина — это число составляющих его компонентов. Каждый компонент списка, в отличие от вектора, может быть произвольным объектом языка **R**. Например, первый компонент списка может быть числовым вектором, второй — списком, а третий — строкой. Пустой список можно получить вызовом функции `vector`:

```
> l<-vector(mode="list",length=3)
```

Список с одновременной инициализацией создается функцией `list`:

```
> l<-list(c(1,5,7),"string",c(TRUE,FALSE))
```

[4]Правила переписывания для этой операции не действуют.

Отдельные компоненты списка могут опционально иметь имя. Для его задания необходимо передавать функции `list` пары «имя=значение»:

```
> l <- list(A=c(1, 5, 7), "string", B=c(TRUE, FALSE))
```

Для извлечения отдельного элемента из списка служит оператор `$`:

```
> .Machine$double.eps
```

Слева от знака `$` — список, а справа — имя извлекаемого элемента.

Оператор `$` не может быть использован, если элемент списка не имеет имени: доступ по индексу возможен только посредством оператора `[[`. Кроме того, оператор `[[` следует применять в том случае, когда интересующее имя элемента уже содержится в некоторой переменной:

```
> l <- list(A=1:10); cname <- "A"
> l$cname
 NULL
```

```
> l[[cname]]
 [1]  1  2  3  4  5  6  7  8  9 10
```

Отметим, что оператор `[` работает как с векторами, так и со списками. Отличие в том, что оператор `[[` возвращает элемент списка, а `[` — *список длины 1, содержащий требуемый элемент*:

```
> .Machine[[1]]
 [1] 2.220446e-16
```

```
> .Machine[1]
 $double.eps
 [1] 2.220446e-16
```

Кроме того, аргумент оператора `[[` всегда имеет длину 1, аргумент оператора `[` может быть вектором с семантикой, описанной выше.

Так же, как и для вектора, для замены элемента списка служит оператор присваивания `<-`.

Matrix и Array. Как уже отмечалось, каждый объект языка **R** обладает набором атрибутов, т. е. списком, компоненты которого имеют имя (именованный список). Многомерные массивы являются хорошим примером применения атрибутов на практике. Так, массив представляет собой вектор с дополнительным атрибутом `dim` и, опционально, атрибутом `dimnames`. Многомерный массив создается функцией `array()`. Атрибут `dim` — числовой вектор, длина которого равна размерности массива (матрица — двумерный массив, `dim=2`). Произведение всех элементов вектора `dim` должно совпадать с длиной объекта *array*. Атрибут `dimnames` позволяет задать имя каждой размерности массива. Например,

```
a<-array(data=NA,dim=length(data),dimnames=NULL)
```

где `data` — исходные данные для массива, `dim`, `dimnames` — значения соответствующих атрибутов.

Матрица может быть создана при помощи функции `matrix`:

```
m<-matrix(data=NA,nrow=1,ncol=1,byrow=FALSE,dimnames=NULL)
```

где `data` – исходное содержимое матрицы, `nrow`, `ncol` – количество строк и столбцов. Аргумент `byrow` позволяет задать порядок заполнения матрицы

из вектора data (по умолчанию по столбцам), dimnames позволяет задать имена строк и столбцов. Кроме того, создать матрицу или массив можно простым назначением атрибута dim вектору:

```
> v <- numeric(6); dim(v) <- c(2, 3); v
```

Массивы и наборы данных имеют свои правила доступа: каждое измерение может индексироваться независимо. Вектор индексов для каждого измерения может принимать любую из форм, допустимых для оператора [(но векторы строк будут индексировать имена измерений, а не элементов):

```
> state.x77[1:2, c("Area", "Population")]
           Area Population
Alabama  50708       3615
Alaska  566432        365
```

Если одно или несколько измерений результата имеют длину 1, то эти измерения «схлопываются»: так, state.x77[1:2,"Area"] является вектором, а не матрицей. Чтобы избежать этого, можно переопределить аргумент drop:

```
>state.x77[1:2,c("Area")]
Alabama  Alaska
  50708  566432
```

```
>state.x77[1:2,"Area",drop=FALSE]
           Area
Alabama  50708
Alaska  566432
```

Так как массивы являются обычными векторами, то их элементы можно индексировать по порядку так же, как если бы измерения отсутствовали. Матрицы индексируются по столбцам; массивы – так, что первое измерение меняется чаще всех, а последнее – реже всех. Отдельные индексирующие векторы могут быть опущены – это используется при выборке отдельных строк (столбцов) матрицы[5]:

```
> m <- matrix(1:6, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> m[2:4]
 [1] 2 3 4
```

```
> m <- matrix(1:6, nrow = 3)
> m[c(2,3), ]
     [,1] [,2]
[1,]    2    5
[2,]    3    6
 > m[,2]
 [1] 4 5 6
```

Для массивов, как и для векторов, определены основные покомпонентные операции. Для матриц реализован ряд специальных операций матричной арифметики, в том числе транспонирование матрицы t() и матричное произведение %*%. Функция solve(a,b) предназначена для решения уравнения a%*%x=b относительно x. Если второй аргумент опущен, в качестве b используется единичная матрица соответствующего размера и, как следствие этого, функция возвращает матрицу, обратную к матрице a. Функция det вычисляет определитель матрицы. Матричные разложения реализова-

[5]Такая гибкость очень часто приводит к трудноуловимым ошибкам из-за пропуска запятых при индексировании.

ны в функциях `qr`, `svd`, `chol` для QR-, SVD- и разложения Холецкого соответственно. Собственные числа и собственные векторы матрицы можно вычислить посредством вызова функции `eigen`.

Основные операции конструирования матриц представлены функциями `diag`, `cbind`, `rbind`. Последние две соединяют матрицы-аргументы по столбцам и по строкам соответственно. Если аргументы оказываются разной длины, используются правила переписывания. Поведение функции `diag` зависит от переданного ей аргумента. Если ее аргумент является матрицей, функция вернет вектор, составленный из элементов главной диагонали, для аргумента-вектора результатом будет диагональная матрица с переданным вектором в качестве главной диагонали, а если аргумент – число, то `diag` вернет единичную матрицу соответствующего размера.

Factor. Объект типа *factor* является способом представления качественных (категориальных) и порядковых типов данных в **R**. Качественные данные принимают значения в некотором конечном множестве, т. е. для задания данных достаточно задать множество *градаций* или *уровней* (levels) фактора. Обычно для кодирования градаций берут обычные числа. Создать переменную данного типа можно при помощи функции `factor`:

```
f<-factor(data,levels,labels=levels)
```

где `data` – отдельные значения (в терминах градаций); `levels` – вектор всех возможных значений для градаций; `labels` – опциональный вектор, содержащий текстовые обозначения градаций, используемые при выводе фактора на экран. Задав дополнительно аргумент `ordered` можно получить порядковый признак:

```
>f<-factor(c(1,1,2,1,3),levels=1:5,labels=c("A","B","C","D","E")); f
 [1] A A B A C
Levels: A B C D E
>f<-factor(c(1,1,2,1,3),levels=1:3,labels=c("A","B","C","D","E"),
+ ordered=TRUE);f
 [1] A A B A C
Levels: A < B < C < D < E
```

Data Frame. Объекты типа *data frame* (массивы данных) обычно используются для хранения результатов статистических экспериментов. Обычно результат каждого отдельного эксперимента – определенный набор значений переменных разного типа (например, «имя» – строкового типа, «пол» – категориального типа, «возраст» – числового типа). Тогда совокупность результатов экспериментов удобно представлять в виде списка из векторов одинаковой длины, коим по сути и является объект типа массивов данных. Создать массив данных можно при помощи функции `data.frame`:

```
df <- data.frame(..., row.names = NULL)
```

В качестве аргументов передаются столбцы создаваемого набора данных либо в виде пар типа *имя = значение*, либо просто «как есть» (в таком случае столбец получит автоматически назначаемое имя `V1`, `V2` и т. п.). Аргумент `row.names` позволяет задать вектор с названиями для отдельных строк (его длина должна совпадать с длиной векторов данных).

Отметим, что большинство объектов могут быть преобразованы в объект типа data.frame «естественным» образом, например:

```
> data.frame(matrix(1:6, nrow = 2, ncol = 3))
  X1 X2 X3
1  1  3  5
2  2  4  6
```

Доступ к элементам и замена элементов объекта типа *data.frame* производятся таким же образом, как и для объектов типа *list*.

Проверка и адаптация различных типов. Для проверки правильности выбранного типа переменной используется функция `is.xxx()`, где вместо `xxx` записывается название предполагаемого типа элемента. Данная функция выдает значение `TRUE`, если тип выбранного элемента совпадает с предполагаемым, и `FALSE` в противном случае. Аналогичный синтаксис имеет операция адаптации `as.xxx()`, где `xxx` – тип, к которому требуется адаптировать выбранный элемент. Проверке и адаптации может быть подвержен как структурный тип объекта, так и примитивный тип его элементов. Например, `as.list(array(0,dim=3))` преобразует тип нулевого вектора 3 в список, составленный из трех векторов размерности 1, а `as.numeric(c("1","2"))` преобразует строковый вектор в числовой. При адаптации в определенный структурный тип возможно использование дополнительных аргументов, например, `as.vector(c("1","2"),mode=numeric)` выдает числовой вектор. Следует быть предельно аккуратными при выполнении адаптации. Рекомендуется проверять полученный результат адаптации и его тип, хотя обычно при невозможности адаптации **R** выводит предупреждение. Отметим, что повторная адаптация далеко не всегда приводит к исходному результату. Например,

```
> a<-matrix(c(1:6),nrow=3)
> t(as.matrix(as.vector(a)))
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    2    3    4    5    6
```

```
> a<-c(0:6)
> as.numeric(as.logical(a))
[1] 0 1 1 1 1 1 1
```

1.2. Чтение и запись данных из файла

Ввод данных из файла. Данные можно представить в текстовом или в бинарном виде. Текстовые данные для статистической обработки — это текстовые таблицы, где каждая строка соответствует строчке таблицы, а колонки определяются при помощи разделителей (пробел/табуляция, за-

пятые, точки с запятой и т. д.). Целая таблица данных может быть считана напрямую функцией `read.table()`. Например, последовательность действий может быть такой:

```
> getwd() # Определим текущую директорию
> dir() # Убедимся в наличии в текущей директории нужного файла
```

Если в текущей директории нет нужного файла

```
> setwd("./R рабочая директория") # Поменяем текущую директорию на рабочую
> df <- read.table("данные.txt", sep=" ", head=TRUE)
```

Первые 3 команды можно выполнить с помощью меню «Файл». В команде `read.table()` «данные.txt» – имя файла данных, `head=TRUE` — имена столбцов, разделителем является пробел (`sep=" "`). В **R** многие команды, в том числе и `read.table()`, имеют для аргументов значения по умолчанию. Например, значение `sep` по умолчанию равно `" "`, т. е. разделителем является любое количество пробелов или знаков табуляции, поэтому аргумент `sep` можно не указывать. Также файлы можно загружать, используя относительную адресацию:

```
> df <- read.table("./R рабочая директория/данные.txt")
```

Существует также более примитивная функция ввода – `scan()`, к которой можно обращаться напрямую. При вводе с клавиатуры после запуска функции вводят значения переменной, каждое с новой строки. Например:

```
> N <- scan(what=numeric(), n=5)
```

где аргумент `what` — тип значений переменной; `n` — количество значений. Значение `n` можно не указывать, в этом случае ввод заканчивается двойным нажатием «Enter». (Об остальных аргументах функции см. «Справку») Если данные вводятся из файла, указывается его имя. Например, при формировании списка:

```
> VVOD <- scan("данные.txt", list(id=, x=0, y=0, z=0))
```

«данные.txt» — имя файла данных; второй аргумент — фиктивная списочная структура – «шаблон», задающий тип считываемых векторов — текстовый и числовой для формируемого списка.

Для сохранения и загрузки бинарных файлов служат команды `save()` и `load()`, для создания объекта `<-`, а для удаления – `rm()`.

Доступ к встроенным наборам данных. Около 100 наборов данных поставляются с **R** (в наборах данных пакетов). Просмотреть список наборов данных позволяет `data()`. Большинство наборов данных, поставляемых с **R**, доступны непосредственно по имени (см. «Справку»).

Редактирование данных. Для редактирования таблицы данных или матрицы в **R** запускается отдельное окружение, удобное для внесения небольших изменений после считывания набора данных. Команда

```
> xnew <- edit(xold)
```

позволит изменить набор данных `xold`, по завершении измененному объекту присваивается имя `xnew`. Для изменения самого набора данных `xold` проще всего использовать `fix(xold)`, что эквивалентно `xold<-edit(xold)`. Чтобы вводить данные через табличный интерфейс, набирается

```
> xnew <- edit(data.frame()).
```

Загрузка данных из других пакетов R. Для преобразования бинарных данных в текстовые таблицы существует пакет `foreign`, позволяющий читать бинарные данные из пакетов `Minitab`, `S`, `SAS`, `SPSS`, `Stata`, `Systat`, а также формат DBF. Также существуют пакеты для загрузки изображений. Для доступа к данным из конкретного пакета используется опция package, например, для просмотра наборов данных пакета `rpart` служит команда

```
> data(package="rpart")
```

Сохранение результатов. R сам может записывать таблицы, графики и другие результаты обработки данных. Например, можно просто скопировать результаты из консоли **R** и вставить в текстовый файл. Функция `sink` направляет весь последующий вывод консоли во внешний файл, для восстановления вывода в консоль вводят `sink()`. Например, в результате выполнения программы

```
> sink("1.txt", split=T)
> 2+7
[1] 9
> sink()
```

во внешний файл запишется `[1] 9`.

Таблицы данных лучше сохранять в виде текстовых таблиц, которые можно будет открыть другими приложениями или текстовыми редакторами. Для этого служит команда `write.table()`:

```
> write.table(file="trees.csv", trees, row.names=F, sep=";", quote=F)
```

В текущую рабочую директорию будет записан файл `trees.csv`, образованный из встроенной в **R** таблицы данных `trees`. Для сохранения истории команд служит команда `savehistory()`, а для сохранения всех созданных объектов — `save.image()`. Таблицы, созданные в **R**, можно сохранять, например в форматах LaTeX([1]) или HTML, при помощи пакета `xtable`. Есть и другие системы генерации отчетов, например пакет `R2HTML`, пакет `brew`, который позволяет создавать автоматические отчеты в текстовой форме (без графиков), и др.

1.3. Функции, правила передачи и назначения аргументов

Функции являются объектами типа *function*. Единственное их отличие, например, от вектора заключается в наличии в языке дополнительного

оператора вызова функции `()`. Оператор `function` создает функцию

```
f <- function(arglist) expr
```

где `arglist` — список аргументов в виде имен или пар *имя = значение*; `expr` — объект типа *expression*, составляющий тело функции. Выполнение функции происходит до выполнения ключевого слова `return`, аргумент которого становится возвращаемым значением, либо до выполнения последнего выражения в теле функции, значение которого будет возвращено из функции:

```
norm <- function(x) sqrt(x%*%x)
```

Аргументы функций можно разделить на обязательные и необязательные. Возможность передавать необязательные аргументы (опции) – одна из отличительных черт языка **R**. Существует определенный механизм *назначения аргументов*. Аргументы могут назначаться по имени и по их позиции в операторе вызова функции. Передача значений аргументам происходит в 3 этапа:

1. Аргументы, переданные в виде пары *имя = значение*, имена которых совпадают явно.
2. Аргументы, переданные в виде пары *имя = значение*, имена которых совпали по уникальному префиксу.
3. Все остальные аргументы по порядку.

Ключевое слово ... (*ellipsis*) позволяет создавать функции с произвольным числом аргументов. К аргументам до троеточия применяются стандартные правила назначения, аргументы после троеточия можно назначить исключительно по полному имени, все неназначенные аргументы «съедаются» троеточием и доступны изнутри функции как именованный список. Эти аргументы назначаются по имени, поэтому при вызове функции они могут находиться на любой позиции. Рассмотрим простой пример

```
f<-function(aa, bb, cc, ab, ..., arg1 = 5, arg2 = 10) {
  print(c(aa, bb, cc, ab, arg1, arg2)); print(list(...))}
f(a = 2, ac = 3, 4, arg1 = 7, aa = 1, 5, 6)
```

Назначение аргументов здесь происходит следующим образом:

1. Назначаются аргументы `arg1` и `aa` по полному совпадению имени.
2. Значение аргументу `aa` уже было присвоено на предыдущем шаге, поэтому `ab` может быть назначен по совпадению префикса `a`, ставшего уникальным.
3. Именованный аргумент `ac` невозможно назначить ни одному из формальных аргументов функции `f`, и поэтому происходит его добавление к списку (...). Для инициализации неназначенного именованного аргумента `arg2` используется значение по умолчанию. После этого шага именованных аргументов не остается.

4. Аргументам `bb` и `cc` присваиваются по порядку значения 4 и 5 соответственно. Оставшееся неиспользованным значение 6 добавляется к списку (...).

При отсутствии оператора троеточия вызов функции оканчивается ошибкой при попытке назначить аргументы на шагах 3 и 4.

В заключение отметим еще одну важную особенность, отличающую **R** от многих других языков программирования: аргументы функций вычисляются не в момент вызова функции, а в момент использования. В связи с этим надо соблюдать осторожность, например, при передаче в качестве аргументов результатов вызовов функций со сторонними эффектами: порядок их вызова может отличаться от ожидаемого. С другой стороны, это позволяет иногда существенно упростить задание значений по умолчанию для аргументов:

```
f<-function(X,L=N %/% 2){N<-length(X); do.something(X, L)}
```

Аргумент `L` вычисляется внутри функции `do.something`, к этому моменту переменной `N` уже назначается значение и `N %/% 2` корректно определено.

1.4. Циклы и условные операторы

Как и многие языки программирования, **R** предоставляет конструкции, позволяющие управлять исполнением программы в зависимости от внешних условий: операторы цикла и условия, но в отличие от «обычных» языков декларативного программирования (например, `C`) они используются существенно реже.

Условный оператор `if` вызывается следующим образом:

```
if (cond) cons.expr else alt.expr
```

где `cond` – логический вектор длины 1; `cons.expr`, `alt.expr` – выражения, которые будут выполнены в случае, если `cond` истинно или ложно соответственно. Значение `NA` для условия не допускается. Если длина условия больше 1, то используется только его *первый элемент* и выдается предупреждение. [6]

Циклы в **R** реализуются при помощи операторов `while` и `for`. Синтаксис первого следующий:

```
while (cond) expr
```

где `cond` – условие выполнения тела цикла (правила вычисления условия совпадают с таковыми для оператора `if`); `expr` – собственно тело цикла. Оператор `for` позволяет перечислить все элементы последовательности:

[6] Отметим, что оператор `==` работает покомпонентно, поэтому в конструкции вида `if (v1 == v2) do.something(v1, v2)` происходит сравнение только первых элементов векторов, а не их содержимого целиком. Для точного или приближенного сравнения векторов используется функция `identical` или `all.equal` соответственно.

```
for (idx in seq) expr
```

где `idx` – переменная цикла; `seq` – перечисляемая последовательность; `expr` – тело цикла. Последовательность `seq` вычисляется до первого выполнения тела цикла, ее переопределение внутри тела цикла не влияет на число итераций, а назначение какого-либо значения переменной `idx` не влияет на следующие итерации цикла.

Цикл можно прервать оператором `break`; закончить текущую итерацию и перейти к следующей – оператором `next`.

1.5. Построение графиков

Одно из основных достоинств **R** – разнообразие типов графиков. Помимо разнообразных типов графиков, предусмотренных в базовом наборе, предоставляется рекомендуемый пакет `lattice`, а также пакеты с CRAN. Графические объекты могут использоваться как в интерактивном, так и пакетном режимах. Дополнительно существует возможность настраивать графики и создавать собственные типы.

В **R** принято выделять 2 основных типа графических команд. Команды высокого уровня рисуют новый график в специальном графическом окне, а команды низкого уровня добавляют детали к уже существующему графику (например, оси, подписи и др.). Кроме того можно использовать интерактивные графические функции добавляя или извлекая информацию из существующего графика с помощью мыши.

Команды высокого уровня. Команда `plot(x,y)` — основная графическая команда, где `x` – вектор значений абсцисс, `y` – вектор значений ординат. Если указан только один аргумент — `plot(y)`, то предполагается, что `x=1:length(y)`. Команда `plot()` распознает тип объекта, который подлежит рисованию, и строит соответствующий график, например:

```
>plot(1:15,main="Первый график")
>plot((x=pi*(c(0:1000)-500)/500),sin(x),'l',main="Второй график")
>plot(chickwts,main="chickwts",sub="Третий график")
```

На первом графике изображается совокупность точек (i, a_i), $i = 1, \ldots, 15$, на втором – график функции $y = \sin(x)$, $x \in [-\pi, \pi]$, построенный по 1001 точке $x = \pi(i - 500)/500$, $i = 0, \ldots, 1000$. На третьем графике строится так называемый 'scatterplot' по таблице данных `chickwts` из двух колонок `weight` и `feed`, встроенной в **R**.

Функция `plot`, как и другие функции высокого уровня, допускает ряд дополнительных параметров. Некоторые из них приведены в табл. 1.1. О назначении других графических параметров можно прочитать в разделе «Справки» для функции `par`.

Таблица 1.1

Параметр	Действие
type=тип графика	Тип выводимого графика: "p"(по умолчанию точки), "l"(линии), "b"(линии и точки), "o"(линии и точки перекрываются), "n" (ничего не рисуется) и др.
xlab=текст ylab=текст	Подписи к оси абсцисс и ординат соответственно
main= заголовок и sub= подзаголовок	Подписи вверху и внизу графика
col=цвет	Цвет графика: "blue","red","green","cyan","magenta", "yellow","black" и т. д. Можно задать rgb-вектор функцией rgb(r,g,b); r,g,b — значения от 0 до 1

Примеры использования функций высокого уровня:

```
# barplot (строит столбовую диаграмму)
>z<- c(15,7,10,3); names(z)<- c("один "два "три "четыре")
>barplot(z,main="Диаграмма",col=c("red","yellow","blue","green"),legend=TRUE)
# pie (строит круговую диаграмму)
>pie(z,main="Диаграмма", col=c("red","yellow","blue","green"))
# hist(строит гистограмму)
>x<- rnorm(50)
>hist(x,breaks=20,col="white")
# boxplot (строит Box-Whisker plot)
>boxplot(x,col="red main="Box-Whisker plot")
>boxplot(weight,feed,data=chickwts,col="white",main="weight и feed")
>boxplot(weight~feed,data=chickwts,col="white",main="weight по feed")
>dotchart(weight)
```

Для двумерных данных могут быть использованы стратифицированные точечные диаграммы

```
>dotchart(Titanic[,,"Adult "No"])
```

В данном случае `Titanic` – встроенный в **R** четырехмерный массив, значения последних двух координат фиксированы. Полностью эти данные можно изобразить с помощью мозаичной диаграммы

```
> mosaicplot(Titanic,col=c("red "white"))
```

Также в **R** есть так называемые *Trellis graphs*, или графики-решетки:

```
> coplot(log(Volume)~log(Girth)|Height,data=trees)
```

Объект `trees` содержит 3 переменные: `Volume`, `Girth` и `Height`. При выполнении команды отображается зависимость объема древесины от объема кроны (логарифмическая шкала) деревьев различной высоты.

Графики трех переменных. Команда `image(x, y, z, ...)` рисует сетку прямоугольников, используя разные цвета для представления значений z; `contour(x,y,z,...)` рисует контурные линии, представляющие значения z; `persp(x,y,z,...)` рисует 3D-поверхность.

Низкоуровневые функции. Функции низкого уровня не стирают имеющиеся изображения, имеют дополнительные параметры, аналогичные параметрам функции `plot`. Примеры функций приведены в табл. 1.2.

Таблица 1.2

Функция	Действие
`points(x,y)` `lines(x,y)`	Дополнительные точки. Ломаная линия, соединяющая указанные точки. Параметры: `type=тип графика`(по умолчанию `"p"` для `points` и `"l"` для `lines`) и др.
`text(x, y, текст)`	Добавляет текст. Если x,y – векторы, добавляет несколько надписей. Параметры: `font=шрифт`, `col=цвет` и т. д.
`abline(k,b)`	Рисует прямую $y{=}kx{+}b$. Параметры: цвет и стиль линии и т. д.; `abline(h=y)` и `abline(v=x)` — для горизонтальных и вертикальных прямых. Указываются координаты y и x
`polygon(x,y)`	Рисует многоугольник с вершинами `x`, `y`. Параметры: цвет заполнения `col=цвет` и границы `border=цвет`
`legend(x,y,легенда)`	Добавляет легенду в указанную точку

Построить 2 графика $y = \sin(t)$ и $y = \cos(t)$ в одном окне можно следующим образом:

```
> t = seq(0,10,by=0.1); y2 = cos(t); y1 = sin(t)
> plot(t, y1, type = "n", main = "2 графика")
> lines(t, y1, col = "blue")
> lines(t, y2, col = "red")
> abline(h = 0) # Горизонтальная ось
> abline(v = 0) # Вертикальная ось
```

Отметим, что если не указывать диапазон значений по осям графика, то эти значения выбираются командой высокого уровня, при добавлении графика его часть может оказаться за пределами диапазона и не будет напечатана. В связи с этим рекомендуется выбирать диапазон значений по осям вручную с помощью опций `xlim` и `ylim`.

Математические формулы. Добавить математические символы и формулы в график можно указав выражение. Подробнее об этом можно прочитать в разделе «Справка» `plotmath`.

Графические устройства. При вводе графической команды открывается экранное графическое устройство и начинается вывод на него. Если следующая команда того же типа, то выводится новый график в том же окне. Закрыть окно можно командой

```
> dev.off()
```

Графических устройств в **R** предусмотрено несколько (количество зависит от ОС). Драйверы устройств запускаются вызовом функции драйвера устройства. Например, в результате выполнения команды

```
> postscript()
```

все последующие графики сохраняются в формате PostScript. Некоторые драйверы: `X11()` — для Unix, `windows()` — для Windows, `quartz()` — для Mac OS X, `postscript()` — печать или создание графических файлов PostScript, `pdf()` — PDF-файл, `png()` — PNG-файл растрового изображения, `jpeg()` — JPEG-файл растрового изображения и др. Для получения полного списка достаточно ввести `help(Devices)`.

Графические опции. R позволяет создать $n \times m$ массив рисунков на одной странице. Каждый рисунок имеет свои собственные поля, и массив рисунков при необходимости окружен внешними полями. Например, нарисуем 2 графика в одном окне:

```
> old.par <- par(mfrow=c(1,2))
# Изменяется одно из значений по умолчанию
> plot(x)
> par(old.par)
# Восстановление старого значения по умолчанию
> dev.off()
```

Функция `par` используется для доступа и изменения списка графических параметров текущего графического устройства. Параметр `mfrow` регулирует количество и размещение изображений на листе. Значение `mfrow` по умолчанию — c(1,1).

Интерактивная графика позволяет определить нужные точки на графике, разместить объект (например, подпись). Для многомерных данных можно вращать облако точек в плоскости разных переменных для выяснения структуры данных. Самая простая из них функция `locator()`:

```
> plot(1:10)
> text(locator(),"точка",pos=4)
```

```
> plot(x, y)
> identify(x, y)
[1] 3 5 6 11
```

После ввода команды `locator` надо щелкнуть левой кнопкой мыши на выбранной точке в графике, а затем правой кнопкой. Интерактивная графика других типов реализована командой `identify(x,y,labels)`, которая разрешает выделить точку, определенную параметрами `x` и `y` (с помощью левой кнопки мыши), рисуя компонент `labels` поблизости (или индекс номера точки), и возвращает индексы выбранных точек при нажатии другой кнопки, а также пакетами `rggobi`, `TeachingDemos` и `iplot`.

Дополнительные сведения, включая полный перечень доступных функций, могут быть получены из **R** с помощью команд:

```
> help(функция), > example(функция), > demo(функция)
```

2. Основные алгоритмы классической статистики

Рассмотрим классические методы обработки статистических данных для выборочной схемы накопления статистической информации и в модели линейной регрессии.

2.1. Электронные таблицы

В языке **R** запрограммированы электронные таблицы ряда классических распределений. В зависимости от цели обращения к электронной таблице следует ввести одну из следующих команд:

```
>qxxx(p,yyy) # вычисление p-квантили распределения
>pxxx(x,yyy) # вычисление функции распределения в точке x
>dxxx(x,yyy) # вычисление плотности распределения (вероятности для
              дискретных распределений) в точке x
>rxxx(n,yyy) # генерирование выборки размера n,
```

где вместо `xxx` подставляется соответствующий **R**-код класса распределений, а вместо `yyy` – параметры распределения. **R**-коды распределений и список параметров приведены в таблице.

Распределение	Обозначение	Код R	Параметры
Бета	$\mathrm{B}(p,q)$	beta	$p>0$, $q>0$, $\mathrm{ncp}\geq 0$
Биномиальное	$\mathrm{Bi}(m,p)$	binom	$m\in\mathbb{N}$, $p\in(0,1]$
Коши	$\mathrm{C}(a,b)$	cauchy	$a\in\mathbb{R}$, $b>0$
Хи-квадрат	χ^2_{df}	chisq	$\mathrm{df}\geq 0$, $\mathrm{ncp}\geq 0$
Показательное	$\mathrm{E}(0,b)$	exp	$b>0$
F-распределение Фишера–Снедекора	$F_{\mathrm{df1},\mathrm{df2}}$	f	$\mathrm{df1}\geq 0$, $\mathrm{df2}\geq 0$, $\mathrm{ncp}\geq 0$
Гамма	$\Gamma(b,p)$	gamma	$p>0$, $b>0$
Геометрическое	$\mathrm{Nb}(1,p)$	geom	$p\in(0,1]$
Гипергеометрическое	$\mathrm{H}(m,n,k)$	hyper	$m\in\mathbb{N}$, $n\in\mathbb{N}$, $k\in\mathbb{N}$
Логнормальное	$\mathrm{LnN}(a,\sigma^2)$	lnorm	$a\in\mathbb{R}$, $\sigma^2\geq 0$
Логистическое	$\mathrm{L}(a,b)$	logis	$a\in\mathbb{R}$, $b>0$
Отрицательное биномиальное	$\mathrm{Nb}(m,p)$	nbinom	$m\in\mathbb{N}$, $p\in(0,1]$
Нормальное	$\mathrm{N}(a,\sigma^2)$	norm	$a\in\mathbb{R}$, $\sigma\geq 0$
Пуассоновское	$\mathrm{P}(\lambda)$	pois	$\lambda\geq 0$
t-распределение Стьюдента	$\mathrm{T}(1,p)$	t	$p>0$, $\mathrm{ncp}\geq 0$
Равномерное	$\mathrm{U}(a,b)$	unif	$a\leq b$
Вейбулла	$\mathrm{W}(a,b)$	weibull	$a>0$, $b>0$
Уилкоксона	—	wilcox	$m\in\mathbb{N}$, $n\in\mathbb{N}$

В прил. 1 приводятся плотности большинства распределений из таблицы и основные их характеристики. Некоторые классы распределений допускают использование параметра нецентральности. Данный факт помечается наличием `ncp` в столбце параметров. Плотности нецентральных распределений (с параметром нецентральности `ncp > 0`) обычно имеют достаточно сложный вид, поэтому в прил. 1 параметр нецентральности не используется. Для некоторых классов распределений действуют умолчания на аргумент `yyy`. Например, в классе нормальных распределений по умолчанию выбирается стандартное нормальное $\mathrm{N}(0,1)$. Функции `qxxx` и `pxxx` помимо указанных аргументов имеют дополнительный аргумент `lower.tail=l`, где `l=TRUE` или `l=FALSE` – логическая переменная (по умолчанию `lower.tail=TRUE`). Подстановка `lower.tail=FALSE` меняет направление вещественной прямой на противоположное. Иными словами,

`> qxxx(p,yyy)` равносильно `> qxxx(1-p,yyy,lower.tail=FALSE)`,

а

`> pxxx(z,yyy)` равносильно `> 1-pxxx(-z,yyy,lower.tail=FALSE)`.

Если известны явные формулы для функции распределения и плотности распределения, таблицы можно запрограммировать самостоятельно. Составим таблицы распределения Парето

```
> ppar<-function(x,a,b){f<-0; if (x>a) f<-1-(a/x)^b; f}
> dpar<-function(x,a,b){f<-0; if (x>=a) f<-b*(a/x)^b/x; f}
> qpar<-function(x,a,b){if (x<1) f<-a/(1-y)^(1/b)}
> rpar<-function(x,a,b){qpar(runif(n),a,b)}
```

Таблицы распределения Лапласа можно запрограммировать с использованием таблиц показательного распределения

```
>plpl<-function(x,a,b){if (x<=a) f<-(1-pexp(-b*x))/2 else
 + f<-1/2+pexp(x-a,b)/2; f}
>dlpl<-function(x,a,b){f<- 0; f<- pexp(abs(x-a),b)}
>qlpl<-function(x,a,b){if(x<=.5)f<-1-qexp(1-2*x)+a else f<-qexp(2*x-1)+a;f}
>rlpl<-function(x,a,b){qlpl(runif(n),a,b)}
```

2.2. Непараметрическое оценивание

Пусть $X_1, \ldots, X_n$ – выборка из распределения с неизвестной функцией распределения F. *Эмпирическим* называется дискретное распределение, имеющее атомы $1/n$ в точках $X_1, \ldots, X_n$, а соответствующая функция распределения F_n называется *эмпирической (или выборочной) функцией распределения*. Эмпирическую функцию распределения, принимающую в точке x значение, равное отношению числа наблюдений, меньших x, к общему числу наблюдений, можно запрограммировать следующим образом:

```
> f<-function(x,t){z<-x[x<t]; length(z)/length(x)}
```

Для построения графика можно использовать следующий скрипт:

```
> xu<-unique(sort(x))
> yu<-0; for(i in 1:length(xu)) yu[i]<-f(x,xu[i]); yu[length(xu)+1]<-1
> z<-stepfun(xu,yu)
> plot(z,verticals=FALSE)
```

но лучше воспользоваться специальной функцией построения выборочной функции распределения `ecdf()` и ее графика

```
> plot(ecdf(x),<параметры>)    или    > plot.ecdf(x,<параметры>)
```

Параметры функции `plot.ecdf` главным образом относятся к оформлению графика и не являются обязательными.

Отметим, что отклонение эмпирической функции распределения от теоретической $D_n(\vec{X}) = \sup_{-\infty<x<\infty} |F_n(x) - F(x)|$ стремится к нулю с вероятностью 1 при $n \to \infty$,[7] что позволяет рассматривать эмпирическую функцию распределения как состоятельную оценку теоретической функции распределения.

Гистограмма частот и полигон. Для выборки из дискретного распределения, носитель которого не имеет точек сгущения (например, распределения со значениями на множестве целых чисел), эмпирическое распределение может использоваться для оценивания дискретной плотности распределения. Для этого следует ввести частоты $\upsilon(x)/n$, где $\upsilon(x)$ – число наблюдений, имеющих значение x. С ростом n, согласно закону больших чисел, частоты будут сходиться к теоретическим значениям $q_\theta(x) = P_\theta(X_1 = x)$. Для абсолютно непрерывного распределения локальные вероятности можно оценивать с помощью гистограммы частот. Для построения гистограммы отрезок, содержащий наибольшее и наименьшее значения выборки, разбивается на интервалы I_i одинаковой длины $h > 0$ и вычисляются выборочные аналоги вероятностей попадания в соответствующие интервалы υ_i/n, где υ_i – число наблюдений, попавших в i-й интервал. *Гистограммой* называется функция $H(x;h) = \upsilon_i/(nh)$, $x \in I_i$, $i \in \mathbb{N}$, и равная нулю при всех остальных значениях x. Уменьшая подходящим образом $h = h(n)$ (так что $nh(n) \xrightarrow[n\to\infty]{} \infty$, $h(n) \to 0$), получаем состоятельную оценку теоретической плотности распределения. Если функция плотности достаточно гладкая, то ломаными ее график можно приблизить лучше, чем ступенчатыми функциями. Отсюда следует, что для оценки гладких плотностей целесообразно использовать так называемый *полигон* вместо гистограммы. Полигон – непрерывная функция, совпадающая с гистограммой частот в середине каждого интервала и линейная между серединами двух соседних интервалов. Нетрудно видеть, что площадь подграфика полигона частот

[7] Теорема Гливенко–Кантелли.

тоже равна единице.[8] Специальная функция

```
>hist(x,<параметры>)
```

выводит гистограмму частот. Данная функция работает и без дополнительных параметров, при этом шаг гистограммы, начало отсчета и масштаб гистограммы выбираются автоматически. Параметры функции `hist` позволяют задавать величину шага с началом отсчета или границы интервалов, отображать ли по оси y частоты или нормированные частоты. Например,

```
> l<- as.integer(min(x-1))
> u<- as.integer(max(x+1))
> hist(x,breaks=c(l:u),freq=TRUE,right=TRUE,col="red")
```

задает гистограмму с шагом 1. Значение параметра `freq=TRUE` означает, что будет выбрана обычная нормировка для оценки плотности, в противном случае значение будет отражать число наблюдений, попавших в соответствующий интервал. Параметр `right=TRUE` означает, что значения, попавшие на границу интервала, будут отнесены в правый интервал. Важно, что массив `breaks` должен покрывать все значения наблюдений, в противном случае будет выдано сообщение об ошибке. Отметим, что элементы массива `breaks` не обязательно должны находиться на одинаковом расстоянии друг от друга. Если в качестве значения `breaks` использовать целое число, то **R** воспринимает его как число интервалов одинаковой длины, на которое следует разделить множество значений. Последний параметр `col="red"` означает, что столбики гистограммы будут красного цвета.

Для ядерного оценивания используется специальная функция `density`,

```
> a<-density(x,<параметры>); plot(a)
```

В качестве параметров используются степень сглаживания `adjust` и `bw` (степень сглаживания определяется произведением `adjust*bw`), форма ядра `kernel` и `window` и др.

Выборочные характеристики. Характеристики эмпирического распределения называются выборочными. Вычисление выборочных характеристик набора наблюдений `x` можно запрограммировать:

```
> mean<-sum(x)/length(x) # выборочное среднее
> var<-sum(x^2)/length(x)-mean^2 # выборочная дисперсия
> sqd<-sqrt(var) # выборочное среднеквадратическое отклонение
> asm<-sum((x-mean)^3)/length(x)/var^(3/2) # выборочная асимметрия
> exc<-sum((x-mean)^4)/length(x)/var^2-3  # выборочный эксцесс
> min(x) # наименьшее значение
> max(x) # наибольшее значение
> sort(x)[trunc(length(x)*p+1)] # выборочная квантиль порядка 'p'
```

[8] Более современные методы оценивания теоретической плотности распределения основаны на построении «ядерных оценок» вида $f_n(x) = (nh_n)^{-1} \sum_{i=1}^{n} K\Big((x - X_i)/h_n\Big)$, где K – некоторая функция ограниченной вариации («ядро»).

В **R** есть специальные функции `mean`, `var` и `sd` для вычисления выборочного среднего, исправленной выборочной дисперсии и корня из нее соответственно. Удобный способ вычисления характеристик выборки:

```
>library(fields)
>c<-cbind(<набор переменных>)
>stats(c)
```

Отметим, что специальные функции **R** вычисляют исправленные выборочную дисперсию и среднеквадратическое отклонение, которые немного отличаются от выборочных.

2.3. Параметрическое оценивание, методы построения оценок

Один из основных методов параметрического оценивания – метод максимального правдоподобия. В основу метода положен принцип максимизации функции правдоподобия $L(\vec{X};\theta) = p_\theta(\vec{X})$, где p_θ – плотность (или дискретная плотность) распределения вектора наблюдений по параметру θ. В регулярном случае обычно переходят от функции правдоподобия к ее логарифму. Пусть $X_1, \ldots, X_n$ – выборка из распределения с плотностью распределения p_θ. Тогда

$$LL(\vec{X};\theta) = \log L(\vec{X};\theta) = \sum\nolimits_{i=1}^{n} \log p_\theta(X_i).$$

Во многих случаях максимум функции правдоподобия находится явно, непосредственно или как решение системы уравнений $U(\vec{X};\theta) = 0$, где $U(\vec{X};\theta) = \frac{\partial}{\partial\theta} LL(\vec{X};\theta)$ – градиент прологарифмированной функции правдоподобия. Такие случаи рассматривать не будем, поскольку использование **R** сводится лишь к вычислению значения оценки по выведенной формуле. Рассмотрим примеры, когда оценка максимального правдоподобия не находится явно.

1. Пусть $X_1, \ldots, X_n$ – выборка из двухпараметрического гамма-распределения (см. приложение) с неизвестными параметрами b и p. Требуется максимизировать функцию

$$LL(\vec{X};b,p) = np\log b - n\log\Gamma(p) + (p-1)\sum\nolimits_{i=1}^{n}\log X_i - b\sum\nolimits_{i=1}^{n} X_i.$$

Дифференцирование данного выражения приводит к системе двух уравнений, решение которой не находится явно, что мотивирует использовать численные методы. Для нахождения максимума функции по выборке `X` можно воспользоваться функцией `maxNR` пакета `maxLik`:

```
> library(maxLik)
> LL<-function(t){sum(dgamma(X,t[1],t[2],log=TRUE))}
> ml<-maxNR(LL,start=c(1,1))
```

Функция `maxNR` вычисляет максимум методом Ньютона–Рафсона, `ml$estimate` – оценка максимального правдоподобия, `ml$maximum` – мак-

симум функции правдоподобия. Необходимым параметром для максимизации является начальная точка. Отметим, что функция `maxNR` находит локальный максимум, поэтому при наличии нескольких локальных максимумов результат может зависеть от начальной точки и не обязательно является глобальным максимумом функции правдоподобия. Асимптотическая дисперсия оценки максимального правдоподобия, в регулярном случае равная $1/I(\theta)$, играет важную роль при построении доверительных интервалов и проверке статистических гипотез. Информация Фишера для параметров гамма-распределения дана в прил. 1, оценку дисперсии получаем подстановкой оценок параметров в соответствующие формулы.

2. Пусть $X_1, \ldots, X_n$ – выборка из двухпараметрического распределения (см. прил. 1) Коши с параметрами a и b. Логарифм функции правдоподобия с точностью до слагаемого, не зависящего от параметров, равен

$$LL(\vec{X}; a, b) \cong n \log b - \sum_{i=1}^{n} \log(b^2 + (X_i - a)^2).$$

Дифференцирование по параметру приводит к нелинейной системе двух уравнений, которая явно не решается, поэтому будем искать максимум функции правдоподобия, построенной по выборке `X`, численно:

```
> library(maxLik)
> LL<-function(t){sum(dcauchy(X,t[1],t[2],log=TRUE))}
> ml<-maxNR(LL,start=c(0,1))
```

Информация Фишера для обоих параметров равна $n/(2b)^2$, таким образом в качестве оценок асимптотических дисперсий можно использовать

```
> V<- (2*ml$estimate)^2/n
```

Альтернативно, можно использовать функцию `mle2` пакета `bbmle` или функцию `nlm` базового пакета для нахождения минимума функции, противоположной LL.

2.4. Доверительное оценивание параметра

Интервал $[T_1, T_2]$, образованный парой статистик $T_1(X_1, \ldots, X_n)$ и $T_2(X_1, \ldots, X_n)$, называется *доверительным интервалом* надежности (или с уровнем доверия) $1 - \alpha$, если при всех $\theta \in \Theta$ выполняется неравенство $P_\theta(T_1 \leqslant \theta \leqslant T_2) \geqslant 1 - \alpha$. Форма доверительного интервала (односторонний, двухсторонний) определяется целями исследования. Предположим, что $G(\vec{X}; \theta)$ монотонно зависит от параметра при каждом фиксированном $\vec{X}$ и имеет одно и то же распределение при каждом фиксированном значении параметра θ. Выбираем g_1 и g_2 по определенному правилу таким образом, чтобы

$$P_\theta(g_1 \leq G(\vec{X}; \theta) \leq g_2) = F_G(g_2+) - F_G(g_1) \geq 1 - \alpha.$$

Границы доверительного интервала $T_1(\vec{X})$ и $T_2(\vec{X})$ для параметра θ получаем как решения относительно θ уравнений $G(\vec{X};\,\theta) = g_i,\ i = 1, 2$. Тогда $P_\theta(T_1(\vec{X}) \le \theta \le T_2(\vec{X})) = P_\theta(g_1 \le G(\vec{X};\,\theta) \le g_2) \ge 1 - \alpha$.

Построение доверительных интервалов по выборке из нормального закона. Пусть $X_1, \ldots, X_n$ – выборка из нормального распределения $N(a, \sigma^2)$. Построение доверительного интервала для параметра a при неизвестной дисперсии базируется на свойстве $\sqrt{n}(\overline{X} - a)/s \stackrel{\mathrm{d}}{=} S_{n-1}$,[9] где S_{n-1} – t-распределение Стьюдента с $n - 1$ степенями свободы. Для построения двухстороннего доверительного интервала следует найти t_α – $(1 - \alpha/2)$-квантиль распределения Стьюдента. Тогда $\mathbf{P}(-t_\alpha \le \sqrt{n}(\overline{X} - a)/s \le t_\alpha) = 1 - \alpha$, т. е. $[\overline{X} - st_\alpha/\sqrt{n}, \overline{X} + st_\alpha/\sqrt{n}]$ будет доверительным интервалом для σ с уровнем доверия $1 - \alpha$. Найдем границы доверительного интервала с помощью **R**, отталкиваясь от значений наблюдаемого вектора `X`, числа наблюдений `n` и уровня доверия `1-al`:

```
> a<-mean(X); s2<-mean(X^2)-a^2
> t<-qt(1-al/2,n-1)
> d<-sqrt(s2/n)*t
> I<-array(dim=2); I[1]<-a-d; I[2]<-a+d
```

Полученный вектор `I` содержит границы доверительного интервала для a с уровнем доверия `1-al`. При построении доверительного интервала для σ^2 используется свойство, что ns^2/σ^2 имеет распределение χ^2_{n-1}:

```
> T<-array(dim=2); T[1]<-n*s2/qchisq(1-al/2); T[2]<-n*s2/qchisq(al/2)
> S<-sqrt(T)
```

В результате, $[T[1], T[2]]$ является доверительным интервалом для σ^2, а $[S[1], S[2]]$ – доверительным интервалом для σ с уровнем доверия `1-al`. При выборе доверительной области $[x_{1\alpha}, x_{2,\alpha}]$ для ns^2/σ^2 использовался принцип равенства $\mathbf{P}(ns^2/\sigma^2 < x_{1\alpha}) = \mathbf{P}(ns^2/\sigma^2 > x_{2\alpha}) = \alpha/2$.

Построение (асимптотических) доверительных интервалов на базе оценок максимального правдоподобия. В регулярном случае оценка максимального правдоподобия $\hat{\theta}_n$ параметра θ обладает свойствами состоятельности и асимптотической нормальности:

$$\sqrt{I(\theta)}(\hat{\theta}_n - \theta) \Rightarrow \mathrm{N}(0, 1),$$

где $I(\theta)$ – информация Фишера для параметра θ по наблюдениям $\vec{X}$. Подстановка оценки максимального правдоподобия $\hat{\theta}_n$ вместо θ в выражение для $I(\theta)$ не нарушает свойства асимптотической нормальности. Данное свойство позволяет получить универсальный метод построения асимптотических доверительных интервалов для широкого класса параметрических

[9] Лемма Фишера.

семейств распределений. Выберем x_α из уравнения $\Phi(x_\alpha) = 1 - \alpha/2$, где $\Phi(x)$ – функция распределения стандартного нормального закона. Тогда интервал

$$[\theta_n - I(\theta_n)^{-1/2} x_\alpha, \theta_n + I(\theta_n)^{-1/2} x_\alpha]$$

является асимптотическим доверительным с уровнем доверия $1 - \alpha$. Если исходный набор наблюдений является выборкой, то $I(\theta) = nI_1(\theta)$, где $I_1(\theta)$ – информация Фишера каждого из наблюдений. Можно показать, что эти доверительные интервалы являются асимптотически наикратчайшими при любом заданном уровне доверия $1 - \alpha \in (0, 1)$.

Построим доверительные интервалы для параметров a и σ^2 по выборке $X_1, \ldots, X_n$ на базе оценок максимального правдоподобия. Информация Фишера для параметров a и σ^2: $I(a) = \sigma^{-2}$ и $I(\sigma^2) = \sigma^{-4}/2$. Получаем асимптотические доверительные интервалы

$$[\overline{X} - (sx_\alpha)/\sqrt{n}, \overline{X} + (sx_\alpha)/\sqrt{n}\,] \quad \text{и} \quad [s^2 - \sqrt{2/n}\, s^2 x_\alpha, s^2 + \sqrt{2/n}\, s^2 x_\alpha]$$

для параметров a и σ^2 соответственно. Отметим, что полученный асимптотический доверительный интервал для a отличается от точного только тем, что квантиль распределения Стьюдента t_α заменяется на квантиль стандартного нормального распределения x_α. Составим программу построения двухсторонних асимптотических доверительных интервалов для дисперсии σ^2 и среднеквадратического отклонения σ:

```
> T2<-array(dim=2); S2<-array(dim=2)
> l2<-s2*qnorm(1-al/2)/sqrt(n/2); l<-s*qnorm(1-al/2)/sqrt(2*n)
> T2[1]<-s2-l2; T2[2]<-s2+l2
> S2[1]<-s-l; S2[2]<-s+l
```

Переменная `T2` содержит границы доверительного интервала для параметра σ^2, а переменная `S2` – для среднеквадратического отклонения.

2.5. Проверка статистических гипотез

Поставим задачу проверки значимости статистической гипотезы

$$H_0 : \theta \in \Theta_0.$$

Гипотеза $H_0 : \theta = \theta_0$ называется простой, т. е. значение параметра фиксировано. В противном случае гипотеза называется сложной. Правило, согласно которому на основании имеющихся данных принимают решение об отвержении или неотвержении статистической гипотезы H_0, называют статистическим критерием. Множество значений $\vec{X}$, на котором гипотеза отвергается согласно выбранному критерию, называется критическим, а множество значений $\vec{X}$, на котором гипотеза принимается, – допустимым. Вероятностью ошибки I рода называют вероятность отвергнуть гипотезу H_0 при ее справедливости. При проверке значимости H_0 вероятность ошибки I рода ограничивают малым числом α, которое носит на-

звание уровня значимости критерия. *Статистикой критерия* называется функция $G(\vec{X}) = G(\vec{X}, H_0)$, которая при справедливости H_0 является подчиненной статистикой (т. е. ее распределение не зависит от θ). Гипотеза отвергается, если $G(\vec{X}) \notin I_\alpha$, и принимается в противном случае, а I_α выбирается из условия $P_\theta(G(\vec{X}) \notin I_\alpha) = P(G(\vec{X}) \notin I_\alpha) < \alpha$, $\theta \in \Theta_0$. Отметим, что условие подчиненности статистики критерия относительно подсемейства $\{P_\theta, \theta \in \Theta_0\}$ может быть ослаблено. В этом случае критерий строится аналогично, но множество I_α, определяющее допустимую и критическую области, должно быть выбрано из соотношения $\sup_{\theta\in\Theta_0} P_\theta(G(\vec{X}) \notin I_\alpha) < \alpha$. Для того чтобы была возможность различать H_0 с альтернативной гипотезой $H_A : \theta \in \Theta_A$, необходимо, чтобы распределение статистики критерия при альтернативе отличалось от ее распределения при H_0.

Обычно, за исключением некоторых специальных случаев, вычисление распределения статистики критерия при каждом фиксированном n оказывается непростой задачей. В связи с этим при достаточно большом объеме выборки имеет смысл использовать асимптотический подход, т. е. вместо точного распределения статистики критерия брать асимптотическое распределение. При этом вместо подчиненности статистики критерия достаточно предположить асимптотическую инвариантность (асимптотическое распределение не зависит от параметра при справедливости H_0). Асимптотические распределения наиболее часто используемых статистик критериев затабулированы. Характеристики некоторых статистических критериев и асимптотических критериев даны в прил. 2. Рассмотрим примеры статистических критериев.

Критерий Стьюдента. Рассмотрим критерий проверки сложной гипотезы согласия $H_0 : a = a_0$ по выборке $X_1, \ldots, X_n$ из нормального распределения $N(a, \sigma^2)$ при неизвестной дисперсии и двухсторонней альтернативе. Статистика критерия $T = \sqrt{n-1}\,(\overline{X} - a_0)/s$ при справедливости основной гипотезы имеет распределение Стьюдента с $(n-1)$-й степенью свободы. Вводим вектор наблюдений `X`, гипотетическое значение параметра `a` и уровень значимости `al`, вычисляем значение T-статистики и квантили порядка `1-al/2`:

```
> m<-mean(X); s<-sqrt(mean(X^2)-mean(X)^2)
> T<-sqrt(n-1)*(m-a)/s        # или   > T<-sqrt(n)*(m-a)/sqrt(var(X))
> x<-qt(1-al/2,length(X)-1)
> pv<-2-2*pt(abs(T),length(X)-1)/2
```

Если $|T| > x$, то гипотеза отвергается, в противном случае – принимается. В последней строчке вычисляется так называемое P-значение, равное наибольшему значению уровня значимости, при котором гипотеза прини-

мается. Альтернативно, можно воспользоваться специальной функцией

```
> q<-t.test(X,mu=a)
```

Объект `q` типа *list* содержит ценную информацию о выборке, полученную с использованием статистики Стьюдента. В частности, `q$statistic` – значение самой статистики, `q$p.value` – P-значение и т. д.

Методы построения параметрических критериев тесно связаны с методами построения доверительных интервалов. Далее рассмотрим некоторые непараметрические критерии, которые по большей части являются асимптотическими.

Критерии χ^2 для проверки простой гипотезы. Принцип построения критериев χ^2 базируется на свойствах мультиномиального распределения. Пусть $X_1, \ldots, X_n$ – выборка из распределения с функцией распределения F.[10] Поставим задачу проверки простой гипотезы согласия $H_0 : F \equiv F_0$. Разобьем все множество допустимых значений наблюдаемой величины на r подмножеств $I_1, \ldots, I_r$ и обозначим $p_j = \mathbf{P}_{F_0}(X_1 \in I_j)$ – гипотетические значения вероятностей, $\sum_{j=1}^r p_j = 1$. Рассмотрим набор частот $\upsilon_1, \ldots, \upsilon_r$, $\upsilon_j = \sum_{i=1}^n \mathbb{1}_{\{X_i \in I_j\}}$, $j = 1, \ldots, r$, где $\mathbb{1}_A = 1$, если условие A выполнено, $\mathbb{1}_A = 0$ в противном случае. Известно, что при справедливости H_0 с ростом $n \to \infty$ имеет место сходимость по распределению

$$X^2 = \sum_{i=1}^r (\upsilon_i - np_i)^2/(np_i) \Rightarrow \chi^2_{r-1}.$$

Асимптотический критерий χ^2 заключается в том, чтобы отвергнуть гипотезу H_0 при $X^2 > x_\alpha$ и принять ее в противном случае, где x_α – квантиль распределения χ^2_{r-1}. Реализуем задачу проверки простой гипотезы согласия $H_0 : a = a_0, \sigma^2 = \sigma_0^2$ по выборке $X_1, \ldots, X_n$ из нормального распределения $N(a, \sigma^2)$ с помощью критерия χ^2. Считаем, что вектор исходных данных `X`, гипотетические значения параметров `a`, `s2` и уровень значимости `al` уже введены. Вещественную прямую разбиваем на r интервалов вручную, вводя вектор `b` границ интервалов размерности `r-1`. Частоты можно получить с помощью функции `hist`, предназначенной для построения гистограммы:

```
> h<-hist(X,breaks=c(min(X),b,max(X)),plot=FALSE); nu<-h$counts
> p<-array(dim=r); s1<-sqrt(s2); p[1]<-pnorm(b[1],a,s1)
> p[2:(r-1)]<-pnorm(b[2:(r-1)],a,s1)-pnorm(b[1:(r-2)],a,s1)
> p[r]<-1-pnorm(b[r-1],a,s1)
> v1<-(nu-p)/n/p; v2<-v1^2
> X2<-sum(v2); xa<-qchisq(1-al,r-1); pv<-pchisq(xa,r-1)
```

[10] Отметим, что данный принцип можно использовать и для многомерных распределений, а семейство допустимых распределений может быть как параметрическим, так и непараметрическим.

Следует отвергнуть гипотезу H_0, если `X2>xa`. Иначе нет оснований для отвержения гипотезы H_0 на уровне α. В случае отвержения гипотезы переменные `v1` и `v2` позволяют понять, какие именно зоны ответственны за такое решение. Для проверки гипотезы H_0 можно воспользоваться специальной функцией

```
> q<-chisq.test(X,p=p)
```

Поскольку χ^2 – асимптотический критерий, вероятное число наблюдений в зонах должно быть достаточно велико. При справедливости H_0 данное свойство выполнено, если вероятности попадания в зоны достаточно велики. Практически рекомендуется, чтобы все значения np_i были не меньше пяти. Обычно эффективность критерия выше, если значения np_i примерно равны.[11] С другой стороны, отметим, что критерий χ^2 использует группированные данные, поэтому не различает альтернативы с вероятностями p_i такими же, как и в основной гипотезе. Для уверенного различения широкого класса гипотез число зон должно быть достаточно велико. Обычно используют от 5 до 20 зон.

Критерии χ^2 для проверки сложных гипотез. Поставим задачу проверки сложной параметрической гипотезы $H_0 : \theta \in \Theta_0$, где $\Theta_0 \subseteq \mathbb{R}^d$ – некоторое подмножество исходного множества параметров размерности d. Распределение статистики $X^2 = X^2(\theta)$ при справедливости основной гипотезы будет зависеть от θ, поскольку вероятности p_i являются функциями от θ. Известно, что при определенных условиях регулярности статистики $\widetilde{X}^2 = \inf_{\theta\in\Theta} X^2(\theta)$ и $\widehat{X}^2 = X^2(\hat{\theta}_M)$, где $\hat{\theta}_M$ – оценка максимального правдоподобия, построенная на базе мультиномиального распределения частот с вероятностями, зависящими от параметра $\theta \in \Theta_0$, асимптотически эквивалентны при справедливости основной гипотезы и сходятся по распределению к χ^2_{r-d-1}. Предположим, что имеется выборка $X_1, \ldots, X_n$ из дискретного распределения с конечным множеством значений $\{0, \ldots, m\}$. Поставим задачу проверки согласия с биномиальным распределением с вероятностями

$$p_j = \mathbf{P}(X_1 = j) = C_m^j p^j (1-p)^{m-j}, \quad p \in (0,1).$$

Выбираем интервалы группировки так, чтобы по возможности число наблюдений в каждой зоне было больше пяти. Вводим `m`, вектор частот `nu`, границы интервалов группировки `x` (считаем интервалы открытыми слева

[11] Иногда при выборе разбиения руководствуются принципом, чтобы все числа наблюдений в зонах были примерно равными. Данный принцип можно использовать при проверке как простой, так и сложной гипотезы, но использование результатов наблюдений при выборе интервалов группировки требует более веских аргументов, чем свойства мультиномиального распределения с фиксированными вероятностями.

и замкнутыми справа), включая точки `-Inf` и `Inf`, и уровень значимости `al`:

```
> v<-length(nu); p<-vector(length=v, mode=numeric)
> P<-function(a){p[1:v]<-pbinom(x[2:(v+1)],m,a)-pbinom(x[1:v],m,a);p}
> X2<-function(a){g<-n*P(a); f<-(nu-g)^2/g; sum(f)}
> XM<-nlm(X2,start=mean(X))
> LM<-function(a){g<-P(a); nu*log(g/(1-g))+m*log(g)}
> library(maxLik); XH<-X2(maxNR(LM,start=mean(X)))
```

Значение $\widehat{X}^2$ содержит переменная `XH`, а значение $\widetilde{X}^2$ – переменная `XM`.

Построим критерий для проверки независимости χ^2 по выборке $(X_1, Y_1), \ldots, (X_n, Y_n)$ из двумерного распределения с неизвестной функцией распределения F_{XY}. Сформулируем гипотезу независимости:

$$H_{\text{ind}} : F_{XY}(x, y) = F_X(x)F_Y(y) \quad \text{при любых} \quad x, y.$$

Разобьем множество допустимых значений X_1 на зоны $I_1, \ldots, I_r$, множество допустимых значений Y_1 – на $J_1, \ldots, J_m$ и составим таблицу частот $\upsilon_{sl} = \sum_{i=1}^{n} \mathbb{1}_{\{X_i \in I_s, Y_i \in J_l\}}$. Совместное распределение величин υ_{sl} является мультиномиальным с вероятностями $p_{sl} = \mathbf{P}(X_1 \in I_s, Y_1 \in I_l)$, $s = 1, \ldots, r$, $l = 1, \ldots, m$. Сформулируем гипотезу

$$H_0 : p_{ij} = p_{i+}p_{+j},$$

где $p_{i+} = \sum_j p_{ij}$ и $p_{+j} = \sum_i p_{ij}$. Очевидно, что при справедливости H_{ind} данная гипотеза также является верной. Таким образом, отвержение гипотезы H_0 влечет отвержение гипотезы H_{ind}. Гипотеза H_0 является параметрической с параметром $(p_{i+}, p_{+j}, i = 1, \ldots, r, j = 1, \ldots, m)$, размерность которого равна $r + m - 2$. Оценки максимального правдоподобия при справедливости основной гипотезы равны υ_{i+} и υ_{+j} соответственно. Реализуем данный критерий, отталкиваясь от введенных значений `nu`, размера матрицы частот `r`×`m`, числа наблюдений `n` и уровня значимости `al`:

```
> nu1<-0; nu1<-colSums(nu); nu2<-rowSums(nu); pm<-nu1%o%nu2/n
> v1<-(nu-pm)^2 /sqrt(pm); v2<-v1^2; XH<-sum(v2)
```

Для проверки независимости также можно использовать специальную функцию `chisq.test()`.

Критерий Колмогорова. Пусть $X_1, \ldots, X_n$ – выборка из распределения с функцией распределения F. Поставим задачу проверки простой гипотезы согласия

$$H_0 : F \equiv F_0,$$

где F_0 – некоторая непрерывная функция распределения. Для проверки данной гипотезы можно воспользоваться критерием Колмогорова. Статистика Колмогорова $D_n = \sup_x |F_n(x) - F_0(x)|$ имеет одно и то же распределение (зависящее только от n) при произвольном выборе непрерывной

теоретической функции распределения F_0. Статистика $\sqrt{n}D_n$ при справедливости основной гипотезы H_0 имеет предельное распределение Колмогорова. Поставим задачу проверки простой гипотезы согласия распределения выборки $X_1, \ldots, X_n$ с нормальным распределением $\mathrm{N}(a_0, \sigma_0^2)$. Вводим исходные данные `X`, гипотетические параметры `a` и `s0`, уровень значимости `al`. Для вычисления значения D_n можно использовать следующий скрипт:[12]

```
> v1<-sort(X); v2<-c(0:(n-1))/n; v3<-c(1:n)/n
> v4<-abs(pnorm(v1,a0,s0)-v2); v5<-abs(pnorm(v1,a0,s0)-v3)
> D<-max(v4,v5)
```

Отметим, что среди стандартных таблиц отсутствует таблица распределения Колмогорова, поэтому для проверки статистической гипотезы H_0 лучше использовать специальную функцию `ks.test()`.

3. Линейные регрессионные модели

Регрессией величины Y по величине X называется условное математическое ожидание $\mathbf{E}\,(Y|X)$. Рассмотрим задачи точечного и доверительного оценивания, а также проверки статистических гипотез в условиях линейной регрессионной модели, задаваемой соотношением

$$\mathbf{E}\,(Y|z) = X^T\beta,$$

где Y – вектор $n \times 1$ наблюдений; z – соответствующие значения ковариат, определяющие $m \times n$ матрицу регрессоров $X = X(z)$; β – $(m \times 1)$-вектор параметров модели. Поскольку значение вектора математических ожиданий не определяет однозначно распределение, в данной модели присутствует неявно мешающий параметр. Перепишем линейную регрессионную модель в виде

$$Y = X^T\beta + \varepsilon,$$

где ε – $(n \times 1)$-вектор ошибок, $\mathbf{E}\,\varepsilon = 0$. В задачах доверительного оценивания и проверки статистических гипотез дополнительно предполагаем, что вектор ошибок имеет нормальное распределение $\mathrm{N}(0, \sigma^2 I)$, где I – единичная матрица. Общие методы обработки статистических данных в условиях линейной регрессионной модели носят название регрессионного анализа. Специальные методы анализа данных при конечном множестве значений ковариат являются предметом дисперсионного анализа.

[12] При отсутствии повторений векторы `v2` и `v3` содержат значения правого и левого пределов выборочной функции распределения в точках, соответствующих значениям порядковых статистик, вектор `v1` – значения самих порядковых статистик.

3.1. Регрессионный анализ

Для построения оценки параметров модели используют *метод наименьших квадратов*, заключающийся в минимизации выражения

$$\mathcal{S}(\beta) = \|Y - X^T\beta\|^2 = (Y - X^T\beta)^T(Y - X^T\beta)$$

по параметру β. Оценка по методу наименьших квадратов (МНК-оценка) находится как решение системы нормальных уравнений $XX^T\beta = XY$. Если определитель матрицы $S = XX^T$ не равен нулю, то МНК-оценка вычисляется по формуле $\hat{\beta} = S^{-1}XY$. В противном случае существует множество решений системы нормальных уравнений, общее решение выражается формулой $\hat{\beta} = S^-Y + S^-Sb$, где S^- – произвольная обобщенная обратная матрица к S; b – произвольный $m \times 1$ вектор-столбец. Отметим, что класс линейных функций параметра модели, допускающих несмещенное оценивание (ДНО), при условии $\det(S) = 0$ ограничен линейными функциями вида $\psi = C^T\beta = AX^T\beta$ при произвольном выборе $(k \times n)$-матрицы A, k – размерность ДНО-функции параметра. Несмещенная оценка ДНО-функции параметра $\psi = C^T\beta$ не зависит от выбора решения системы нормальных уравнений $\hat{\psi} = C^T\hat{\beta}$. В предположении, что матрица ковариации вектора ошибок равна $\sigma^2 I$, оценка $\hat{\sigma}^2 = (n-r)^{-1}\mathrm{SS}_e = (n-r)^{-1}\mathcal{S}(\hat{\beta})$ является несмещенной для параметра дисперсии σ^2, а матрица ковариации $\hat{\psi}$ равна $\Gamma_{\hat{\psi}} = C^TS^-C\sigma^2 = B\sigma^2$.

При построении доверительных областей (эллипсоидов) ДНО-функции параметра ψ дополнительно предполагают, что компоненты вектора ε суть независимые одинаково распределенные величины с распределением $\mathrm{N}(0, \sigma^2)$. Отметим, что в этом случае МНК-оценка является оценкой максимального правдоподобия. Тогда $\hat{\psi}$ имеет $\mathrm{N}(\psi, \Gamma_{\hat{\psi}})$ -распределение и, если $|\Gamma_{\hat{\psi}}| \neq 0$, $(\hat{\psi} - \psi)^T\Gamma_{\hat{\psi}}^{-1}(\hat{\psi} - \psi)^T$ имеет χ^2_q -распределение, $q = \mathbf{rk}(\Gamma_{\hat{\psi}}) = \mathbf{rk}(B)$. Тогда $(\hat{\psi} - \psi)^TB^{-1}(\hat{\psi} - \psi)^T/(qs)$ имеет распределение Фишера–Снедекора $F_{q,n-r}$ и область

$$A_\alpha = \Big\{\vec{x} : q^{-1}(\vec{x} - \hat{\psi})^TB^{-1}(\vec{x} - \hat{\psi}) \leq s^2x_\alpha\}\Big\},$$

где $x_\alpha : F_{q,n-r}(x_\alpha) = 1 - \alpha$ – q-мерный доверительный эллипсоид (так как B^{-1} положительно определена) для ψ. В случае $q = 1$ можно построить доверительный интервал на базе распределения Стьюдента:

$$\Big[\hat{\psi} - \frac{t_\alpha\sqrt{b\,\mathrm{SS}_e}}{\sqrt{n-r}}, \hat{\psi} + \frac{t_\alpha\sqrt{b\,\mathrm{SS}_e}}{\sqrt{n-r}}\Big], \tag{1}$$

где $t_\alpha : S_{n-r}(t_\alpha) = 1 - \alpha/2$. Данный метод позволяет строить также несимметричные доверительные интервалы. При большом числе ДНО-функций

параметров достаточно сложно визуально представить доверительный эллипсоид, поэтому прибегают к построению совместных доверительных интервалов по S-методу Шеффе. Пусть $\psi_1, \ldots, \psi_q$ – базис пространства L_q ДНО-функций параметров. Тогда

$$\mathbf{P}(\hat{\psi} - x_\alpha \hat{\sigma}_{\hat{\psi}} \le \psi \le \hat{\psi} + x_\alpha \hat{\sigma}_{\hat{\psi}},\ \forall\, \psi \in L_q) = 1 - \alpha,$$

где $x_\alpha = (q f_\alpha)^{1/2}$, f_α – квантиль $F_{q,n-r}$-распределения порядка $1-\alpha$.

Поставим задачу проверки гипотезы

$$H_0 : C^T \beta = 0.$$

Для проверки данной гипотезы используют F-критерий. Статистика F-критерия может быть вычислена двумя способами:

$$\mathbb{F} = \hat{\psi}^T B^{-1} \hat{\psi} / (q s^2) = (n-r)\mathrm{SS}_H / (q \mathrm{SS}_e),$$

где $\mathrm{SS}_H = \mathcal{S}(\hat{\beta}_H) - \mathrm{SS}_e$, $\hat{\beta}_H$ – оценка максимального правдоподобия, построенная в предположении $C^T\beta = 0$. При справедливости H_0 статистика $\mathbb{F}$ имеет распределение Фишера–Снедекора $F_{q,n-r}$.

Вводим вектор наблюдений `Y`, матрицу регрессоров `X`, матрицу `C` полного ранга, уровень значимости `al` и приступаем к анализу:

```
> n<- length(Y); r<- dim(X)[1]    # число наблюдений и параметров
> S<- solve(X%*% t(X))    # обращаем матрицу
> bht<- S%*%X%*%Y    # МНК-оценка
> res<- Y−t(X)%*% bht    # ошибки
> SSE<- sum(res∧2)    # Сумма квадратов ошибок
> sht<- SSE/(n−r)    # Оценка дисперсии
> B<- t(C)%*%S%*%C    # Матрица B
> q<- dim(C)[1]    # Число ограничений на параметры
> pht<- t(C)%*%bht    # ДНО-функция параметра
> SSH<- t(pht)%*%solve(B)%*%pht    # SSH
> F<- SSH/q/sht    # F-статистика
> f<- qf(1-al,q,(n-r))    # Граница критической области
> CI<- matrix(nrow=q,ncol=2); xa<- sqrt(q*f); d<- xa*sqrt(diag(B)*sht);
> CI[,1]<- pht−d; CI[,2]<- pht+d    # совместные доверительные интервалы
```

В случае сингулярности матрицы XX^T функция `solve` работать не будет. Для нахождения обобщенной обратной матрицы можно использовать функцию `ginv` пакета `MASS`, а для вычисления ранга матрицы служит функция `rankMatrix` пакета `Matrix`.

Далее можно продолжить исследование выбирая фиксированные альтернативы, а также определить границы различимости с вероятностью ошибки II рода, не превышающей фиксированного значения $\beta \in (0,1)$.

Наиболее важные результаты анализа данных в рамках линейной модели могут быть получены с использованием специальной функции `lm`. Данная функция сконструирована для исследования регрессионной модели

$$y_i = \beta_1 + x_{2i}\beta_2 + \ldots + x_{ri}\beta_r + \varepsilon_i, \quad i = 1, \ldots, n,$$

что подразумевает ввод регрессоров по строкам; `X2,...,Xr` – векторы значений регрессоров; `Xj`$= (x_{j1}, \ldots, x_{jn})$, $j = 2, \ldots, r$. Первая строка матрицы регрессоров по умолчанию выбирается состоящей только из единиц. Первым аргументом функции `lm` идет модель. Рассмотренная ранее модель при $r = 3$ определяется формулой

```
> reg<-formula(Y~X2+X3)
```

Параметр сдвига β_1 включен в модель по умолчанию. Формула модели без параметра сдвига выглядит следующим образом:

```
> reg2<-formula(Y~X2+X3-1)
```

Способ ввода формулы позволяет рассматривать некоторые нелинейные зависимости, например `log(Y)~X2` или `Y~X2+cos(X1^2)`. Отметим, что формула `Y~X2+X2^2` равносильна `Y~X2`.

Для задания квадратичной регрессионной зависимости

$$\mathbf{E}\,(\mathtt{Y}|\mathtt{X2}) = \beta_0 + \beta_1\mathtt{X2} + \beta_2\mathtt{X2}^2$$

нужно ввести `Y~I(X2)+I(X2^2)`. Для запуска функции `lm` следует ввести

```
> ans<-lm(<формула>,weights=<веса>,...)
```

Отметим, что по умолчанию выводятся лишь значения МНК-оценок параметров. Результаты анализа можно получить, набрав в командной строке `ans`, `summary(ans)`, `anova(ans)` и `aov(ans)`. Для получения более полной информации можно воспользоваться справочной системой.

Альтернативно, можно использовать функцию `lsfit`:

```
>X2m<-array(c(X2,X3),dim=c(n,r-1)); ans<-lsfit(X2m,Y)
```

Для вывода результатов анализа следует набрать `ans`, `ls.print(ans)` и (или) `ls.diag(ans)`.

3.2. Дисперсионный анализ

В задачах дисперсионного анализа не предполагается какая-либо количественная интерпретация значений ковариат. Условия или совокупности условий, влияющие на результат эксперимента, называются факторами, различные значения факторов носят название уровней факторов. В модели однофакторного дисперсионного анализа (простая группировка) предполагается наличие одного фактора A, влияющего на результат эксперимента. Модель однофакторного дисперсионного анализа

$$y_{ij} = \beta_i + \varepsilon_{ij}, \quad i = 1, \ldots, I, \; j = 1, \ldots, n_i,$$

где ε_{ij} – независимые и одинаково распределенные случайные величины, имеющие нормальное распределение $\mathrm{N}(0, \sigma^2)$, естественным образом укладывается в модель регрессионного анализа. Столбцы матрицы регрессо-

ров представляют собой векторы $(0,\ldots,1,\ldots,0)$, состоящие из нулей и единицы на позиции, соответствующей уровню фактора. Очевидно, что $\hat{\beta}_i = y_{i+}/n_i = \overline{y}_{i+}$ («+» означает суммирование по соответствующему индексу, а черта сверху – усреднение). Для задач дисперсионного анализа характерна другая эквивалентная параметризация:

$$y_{ij} = \mu + \alpha_i^A + \varepsilon_{ij}, \quad i = 1,\ldots,I,\ j = 1,\ldots,n_i,$$

μ – некоторое базовое среднее значение, α_i^A – главные эффекты фактора A. Чтобы уравновесить размерность параметра, на главные эффекты фактора A вводятся ограничения $\alpha_*^A = \sum_{i=1}^I v_i\alpha_i = 0$, где v_i – фиксированные веса, $\sum_i v_i = 1$. В случае $v_i = 1/I$ получаем центральную параметризацию, в которой μ интерпретируется как генеральное среднее, а в случае $v_s = 1$ и $v_i = 0$ при $i \neq s$ (s-й уровень фактора A выбирается базовым), μ – среднее значение наблюдаемой величины при базовом уровне фактора A. Очевидно, что $\mu = \beta_* = \sum_i v_i\beta_i$, $\alpha_i^A = \beta_i - \beta_*$, по этим же формулам пересчитываются и МНК-оценки. Для проверки гипотезы

$$H_A : \alpha_i^A = 0,\ i = 1,\ldots,I$$

отсутствия влияния уровня фактора A на результат используют F-критерий, статистика которого $\mathbb{F} = (n-I)SS_A/((I-1)SS_e)$, где $SS_e = \sum_{ij}(y_{ij} - \overline{y}_{i+})^2$, $SS_A = \sum_i k_i(\overline{y}_{i+} - \overline{y}_{*+})^2$, $n = \sum_i n_i$. Статистика $\mathbb{F}$ при справедливости H_A имеет $F_{(I-1),(n-I)}$-распределение.

Пусть введены значения `Y` и соответствующие уровни фактора `A`. Обычно данные вводятся парами (y_i, a_i), где y_i – значение наблюдаемой величины, а a_i – значение уровня фактора. В этом случае удобно использовать алгоритмы регрессионного анализа для решения задач дисперсионного анализа. Пусть введены `Y` – вектор наблюдений и `A` – соответствующий вектор уровней факторов. Определяем матрицу регрессоров и изучаемые функции параметров в центральной параметризации:

```
> D<-levels(as.factor(A))        # Уровни фактора A
> m<-length(D); n<-length(Y)     # Уровни D[i] заменяем на i
> XM<-matrix(0,nrow=m,ncol=n); for (i in 1:n) for (j in 1:m)
 + if (A[i]==D[j]) XM[i,j]<-1
> v<-vector(1,dim=I)/I; V<-matrix(v,nrow=I,ncol=I,byrow=TRUE)
> AM<-diag(1,nrow=I,ncol=I)-V
```

Дальнейшие действия были описаны в 3.1. Параметры μ и $\alpha = (\alpha_1,\ldots,\alpha_I)$ представляют собой линейные функции параметра β и вычисляются с использованием вектора `v` и матрицы `AM` соответственно.

В модели двухфакторного дисперсионного анализа предполагается, что имеются 2 фактора A и B, которые могут влиять на результат:

$$y_{ijk} = \beta_{ij} + \varepsilon_{ijk}, \quad i = 1,\ldots,I,\ j = 1,\ldots,J,\ k = 1,\ldots,K_{ij},$$

где ε_{ijk} – независимые и одинаково распределенные случайные величины. В данной постановке задача эквивалентна однофакторному дисперсионному анализу с группировкой по двум факторам. Цели двухфакторного дисперсионного анализа – установить влияние каждого из факторов и их совместное влияние на результат эксперимента. Выберем веса v_i и u_i: $v_+ = u_+ = 1$ и рассмотрим параметризацию, в большей степени соответствующую духу дисперсионного анализа:

$$y_{ijk} = \mu + \alpha_i^A + \alpha_j^B + \gamma_{ij}^{AB} + \varepsilon_{ijk},\ i{=}1,\ldots,I,\ j{=}1,\ldots,J,\ k{=}1,\ldots,K_{ij}$$

с ограничениями $\alpha_*^A = \beta_*^B = \gamma_{i*} = \gamma_{*j} = 0$ («$*$» означает суммирование с весами v_i по индексу i и с весами u_j по индексу j). Параметры α_i^A и α_j^B называют главными эффектами факторов A и B соответственно, а γ_{ij}^{AB} – взаимодействиями. В терминах исходных параметров β_{ij}:

$$\mu = \beta_{**},\ \alpha_i^A = \beta_{i*} - \beta_{**},\ \alpha_j^B = \beta_{*j} - \beta_{**},\ \alpha_{ij}^{AB} = \beta_{ij} - \beta_{i*} - \beta_{*j} + \beta_{**}.$$

Основными гипотезами двухфакторного дисперсионного анализа являются:

$$H_{AB} : \gamma_{ij}^{AB} = 0,\ i = 1,\ldots,I,\ j = 1,\ldots,J;$$

$$H_A : \alpha_i^A = 0,\ i = 1,\ldots,I \quad \text{и} \quad H_B : \alpha_j^B = 0,\ j = 1,\ldots,I.$$

В общем случае для построения совместных доверительных интервалов и проверки статистических гипотез удобно использовать общие методы регрессионного анализа, хотя приведение модели к регрессионной и обратный переход к модели дисперсионного анализа осуществляются сложнее, чем для однофакторного случая.

Для решения большинства задач дисперсионного анализа можно воспользоваться специальными функциями `lm()` и (или) `aov()`. Первая выводит оценки параметров, вторая – таблицу дисперсионного анализа. Основным аргументом обеих функций является модель. Чтобы **R** воспринимал модель `y~a` как модель дисперсионного анализа, следует адаптировать переменную `a` к типу *factor*:

```
> A<-factor(a)
```

В данном случае фактор `A` соответствует переменной `a`. Можно также ввести допустимые уровни фактора и их порядок. По умолчанию процедуры `lm` и `aov` используют параметризацию, когда первый уровень каждого фактора выбирается в качестве базового. Для выбора других параметров можно переопределить их:

```
> MC<-array(c(...),dim=(d,d-1))
> contrasts(A)<-MC
```

где d – число уровней фактора A. Отметим, что для вычисления оценок $\hat{\beta}$ нужно воспользоваться формулой

```
>bht <-int+MC%*%aht
```

где aht – полученные оценки параметров. Случаю с одинаковыми весами соответствует матрица $MC = \begin{pmatrix} I_{d-1} \\ -\mathbf{1}_{d-1} \end{pmatrix}$, где I_{d-1} – единичная матрица размера $d-1$; $\mathbf{1}$ – строка из единиц. Существует ряд стандартных наборов параметризаций: contr.treatment, contr.helmert, contr.poly, contr.sum и contr.SAS. По умолчанию действует установка

```
> options(contrasts=c("contr.treatment","contr.poly"))
```

Для перехода к параметризации с равными весами следует набрать

```
> options(contrasts=c("contr.sum","contr.poly"))
```

При решении задач двухфакторного дисперсионного анализа переменные, задающие уровни факторов, следует адаптировать к типу *factor*. Пусть имеется набор наблюдений y и 2 фактора a и b. Рассмотрим следующие модели:

```
> am<-model(y∼a+b)   # аддитивная модель (взаимодействия равны нулю)
> sm<-model(y∼a*b)   # полная модель
> bnm<-model(y∼a+a:b)  # главные эффекты b равны нулю
```

Используя нужную модель в качестве аргумента функций lm() и aov(), получаем оценки параметров и таблицу двухфакторного дисперсионного анализа в данной модели. Отметим, что в двухфакторном (и в многофакторном) анализе по умолчанию используется параметризация, где в качестве базового выбирается первый уровень каждого фактора.

Список рекомендуемой литературы

Ивченко Г. И., Медведев Ю. И. Математическая статистика. М.: Высш. шк., 1984.

Боровков А. А. Математическая статистика. М.: Наука, 1984.

Леман Э. Теория точечного оценивания. М.: Наука, 1991.

Леман Э. Проверка статистических гипотез. М.: Наука, 1964.

Рао С. Р. Линейные статистические методы и их применение. М.: Наука, 1968.

Шеффе Г. Дисперсионный анализ. М.: Наука, 1980.

1. Основные классы распределений и их характеристики

Таблица П.1.1. Абсолютно непрерывные распределения

Класс распределения	Обозначение	Плотность распределения	Информация Фишера
Нормальное /Гауссовское/	$\mathrm{N}(a,\sigma^2)$	$\frac{1}{\sqrt{2\pi}\,\sigma}\exp\left(-\frac{(x-a)^2}{2\sigma^2}\right)$	$I(a)=\sigma^{-2}$, $I(\sigma^2)=1/2\sigma^4$
Равномерное	$U(a,b)$	$\mathbb{I}_{[a,b]}(x)/(b-a)$	Не существует
Показательное	$\mathrm{E}(a,b)$	$b\exp(-b(x-a))\mathbb{I}_{[a,\infty)}(x)$	$I(a)-\nexists$; $I(b)=b^2$
Двухстороннее показательное /Лапласа/	$\mathrm{DE}\,(a,b)$	$\frac{1}{2b}\exp\left(-\frac{\lvert x-a\rvert}{b}\right)$	$I(a)=I(b)=1/b^2$
Гамма	$\Gamma(b,p)$	$\frac{b^p x^{p-1}\exp(-bx)}{\Gamma(p)}\mathbb{I}_{[0,\infty)}(x)$	$I(b)=b^2p$; $I(p)=\psi'(p)^*$
Коши	$\mathrm{C}(a,b)$	$b/(\pi(b^2+(x-a)^2))$	$I(a)=I(b)=1/(2b^2)$
Бета	$\mathrm{B}(p,q)$	$\frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)}x^{p-1}(1-x)^{q-1}\mathbb{I}_{[0,1]}(x)$	$I(p)=\psi'(p)-\psi'(p+q)^*$; $I(q)=\psi'(b)-\psi'(p+q)$
Парето	$\mathrm{P}(a,b)$	$ba^b x^{-(b+1)}\mathbb{I}_{[a,\infty)}(x)$	$I(a)-\nexists$; $I(b)=1/b^2$
Логистическое	$\mathrm{L}(a,b)$	$\frac{\exp(-(x-a)/b)}{b(1+\exp(-(x-a)/b))^2}$	$I(a)=\frac{1}{3b^2}$; $I(b)=\frac{3+\pi^2}{9b^2}$
Вейбулла	$\mathrm{W}(a,b)$	$ab^a x^{a-1}\exp(-(bx)^a)\mathbb{I}_{\{x>0\}}$	$I(a)=(\psi(2)^2+\zeta(2)^2)^*/a^2\approx$ $\approx 1.82368/a^2$; $I(b)=a^2/b^2$
t-распределение (Стьюдента)	$\mathrm{T}(b,p)$	$\frac{\sqrt{2}\Gamma((p+1)/2)}{\Gamma(p/2)\sqrt{\pi p\,b}}\left(1+\frac{2x^2}{p\,b}\right)^{-(p+1)/2}$	$I\left(\sqrt{\frac{bp}{2}}\right)=\frac{2}{bp}-\frac{3\sqrt{\pi}\,\Gamma\left(\frac{p+1}{2}\right)}{2bp\,\Gamma(p/2)}$

Таблица П.1.2. Дискретные распределения

Класс распределения	Обозначение	Дискретная плотность распределения	Информация Фишера
Биномиальное	$\mathrm{Bi}(m,p)$	$C_m^k p^k(1-p)^{m-k}$, $k=0,\ldots,m$	$I(p)=\frac{m}{p(1-p)}$
Отрицательное биномиальное /Паскаля/	$\mathrm{Nb}(m,p)$	$C_{m+k-1}^k p^m(1-p)^k$; $k=0,1,\ldots$	$I(p)=\frac{m}{p^2(1-p)}$
Пуассона	$\mathrm{P}(\lambda)$	$\frac{\lambda^k}{k!}\exp(-a)$, $k=0,1,\ldots$	$I(a)=1/a$
Гипергеометрическое	$\mathrm{H}(m,n,k)$	$C_m^i C_n^{k-i}/C_{n+m}^k$, $i=0,\ldots,k$	Не существует

$^*\psi(p)=\Gamma'(p)/\Gamma(p)$; $\zeta(q)=\sum_{q=1}^{\infty}n^{-q}$.

2. Некоторые статистические критерии

Название	Семейство	Гипотеза	Статистика критерия $\stackrel{\mathrm{d}}{=}(\Rightarrow)$ ее точное (предельное) распределение	Примечания	**R**-команда; **R**-квантиль
Стьюдента	$\mathrm{N}(a, \sigma^2)$ $a \in \mathbb{R}$, $\sigma > 0$	Согласия с $H_0 : a = a_0$ (сложная параметрическая)	$S_n = \dfrac{\overline{X} - a}{s} \stackrel{\mathrm{d}}{=} T(1, (n-1)/2)$ t-распределение Стьюдента	σ неизвестно	`t.test()`; `qt()`
Хи-квадрат (согласия с простой гипотезой)	Непараметрическое	Согласия для вероятностей попадания в зоны (простая)	$X^2 = \sum_{i=1}^{r} \dfrac{(n_i - np_i)^2}{np_i} \Rightarrow \chi^2_{r-1}$ χ^2-распределение Пирсона	r – число зон ; n_i – число элементов в i-й зоне; p_i – теоретическая вероятность попадания в i-ю зону при H_0	`chisq.test()`; `qchisq()`
Хи-квадрат (независимости)	Непараметрическое двумерное	Независимости компонент в таблице частот (сложная параметрическая)	$X^2 = \sum_{i,j} \dfrac{(n_{ij} - \frac{n_{i+}n_{+j}}{n})^2}{n_{i+}n_{+j}/n} \Rightarrow$ $\Rightarrow \chi^2_{(r-1)(s-1)}$ χ^2-распределение Пирсона	$r/s/$ – число строк /столбцов/; n_{ij}–число элементов в (i,j)-й ячейке; $n_{i+}/n_{+j}/$– суммы n_{ij} по столбцам /строкам/; n – общее число наблюдений	
Колмогорова	Непараметрическое	Согласия с $H_0 : F \equiv F_0$ (простая)	$\max\limits_{t\in(-\infty,\infty)} \lvert F_n(t) - F_0(t)\rvert \stackrel{\mathrm{d}}{=} \mathcal{K}_n$; $\sqrt{n}\ \mathcal{K}_n \Rightarrow \mathcal{K}$; $\mathcal{K}_n(\mathcal{K})$ – точное (асимптотическое) распределение Колмогорова	F_n – эмпирическая функция распределения; F_0 – непрерывная гипотетическая функция распределения	`ks.test()`; ...
Колмогорова–Смирнова	Непараметрическое 2 выборки	Однородности выборок $H_0 : F_1 \equiv F_2$ (сложная непараметрическая)	$\max\limits_{t\in(-\infty,\infty)} \lvert F_{1n}(t) - F_{2n}(t)\rvert \stackrel{\mathrm{d}}{=} \mathcal{K}2$ $\sqrt{n}\,\mathcal{K}2_n \Rightarrow \mathcal{K}2$; $\mathcal{K}2_n(\mathcal{K}2)$ – точное (асимптотическое) распределение Колмогорова–Смирнова	F_{in} – эмпирическая функция распределения по i-й выборке; F_1, F_2 – непрерывные теоретические функции распределения	
Уилкоксона (Манна–Уитни)	Непараметрическое 2 выборки	Однородности выборок $H_0 : F_1 \equiv F_2$ (сложная непараметрическая)	$W = \sum_i R_i \stackrel{\mathrm{d}}{=} \mathcal{W}_{n,m}$; $\mathcal{W}_{n,m}$ – распределение Уилкоксона	R_i – ранги элементов 1-й выборки в объединенном наборе наблюдений; n – общее число наблюдений	`wilcox.test()` `(paired=FALSE)`; `qwilcox()`

Оглавление

Основные алгоритмы численного анализа. Использование пакета R(S-PLUS) для анализа статистических данных

Редактор Э. К. Долгатов

Подписано в печать . Формат 60 × 84 1/16. Бумага офсетная.
Печать офсетная. Гарнитура «Times New Roman». Печ. л. 2.5.
Тираж 50 экз. Заказ .

Издательство СПбГЭТУ «ЛЭТИ»
197376, С.-Петербург, ул. Проф. Попова, 5