Учебно-методическое пособие

Зарядов Иван Сергеевич

Статистический пакет **R**: теория вероятностей и математическая статистика.

Москва

Издательство Российского университета дружбы народов

2010

Оглавление

Глава 1

Основы теории вероятностей в R

1.1 Теория вероятностей в R. Базовые вероятностные распределения

1.1.1 Базовые вероятностные распределения

В базовой установке **R** (пакет **stats**) реализованы следующие вероятностные распределения:

1. дискретные:
    - биномиальное;
    - пуассоновское;
    - геометрическое;
    - гипергеометрическое;
    - отрицательно биномиальное;
    - полиномиальное;
2. непрерывные:
    - бета-распределение;
    - распределение Коши;
    - экспоненциальное;
    - χ^2-распределение;
    - распределение Фишера (f-распределение);
    - гамма-распределение;
    - логнормальное;

- логистическое;
- нормальное;
- распределение Стьюдента (t-распределение);
- равномерное;
- распределение Вейбулла.

3. ранговые распределения Вилкоксона.

Для каждого из распределений в **R** имеются четыре функции:

- плотность распределения (для непрерывных случайных величин) и вероятность принятия случайной величиной конкретного значения (дискретные с.в.) — префикс **d** перед названием распределения;
- функция распределения (ФР) с.в. — префикс **p** перед названием распределения;
- квантили распределения — префикс **q** перед названием распределения;
- случайная выборка по заданному распределению — префикс **r** перед названием распределения.

Заметим, что ФР в **R** есть вероятность $F_\xi(x) = P\{\xi \leqslant x\}$, а не $F_\xi(x) = P\{\xi < x\}$, что принято в русской школе теории вероятностей.

В следующих разделах рассмотрим распределения подробно и приведём примеры.

1.1.2 Биномиальное распределение

Случайная величина ξ, описывающее число «успехов» в ряде испытаний Бернулли, принадлежит биномиальному распределению $B(n, p)$ с параметрами p — вероятность «успеха» в испытании и n — число испытаний Бернулли.

Вероятность $P\{\xi = k\}$ имеет вид

$$P\{\xi = k\} = C_n^k p^k (1-p)^{n-k}.$$

В **R** для биномиального распределения реализованы следующие функции:

```
dbinom(x, size, prob, log = FALSE)
pbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)
qbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)
rbinom(n, size, prob)
```

Аргументы функций:

- **x** — целочисленный неотрицательный вектор — вектор значений случайной величины ξ .
- **q** — неотрицательный вектор — вектор квантилей.
- **p** — вектор вероятностей.
- **n** — длина создаваемого вектора.
- **size** — один из параметров биномиального распределения — число испытаний Бернулли.
- **prob** — второй параметр биномиального распределения — вероятность «успеха» в одном испытании Бернулли.
- **log** — логический аргумент (по умолчанию его значение **FALSE**). Нужно ли вычислять логарифм вероятности.
- **log.p** — аналогично.
- **lower.tail** — логический аргумент. Если его значение **TRUE**. то используется $P\{\xi \leqslant x\}$, в противном случае используется $\bar{F}(x) = P\{\xi > x\}$.

Что же получаем в результате:

- **dbinom(x, size, prob)** — вероятность того, что случайная величина примет заданное значение. Если **x** — нецелое или отрицательное число, то будет выдано предупреждение и результатом будет ноль.
- **pbinom(q, size, prob)** — значение функции распределения в точке **q**.
- **qbinom(p, size, prob)** — значение квантиля Q_p для заданной вероятности **p**. Квантиль Q_p определяется как наименьшее целое значение x такое, что $F(x) \geqslant p$ (заметим, что в русской школе теории вероятностей квантиль Q_p определяется как наибольшее целое значение x такое, что $F(x) < p$).
- **rbinom(n, size, prob)** — создаём вектор случайных значений (выборку), подчиняющийся заданному распределению.

Замечание. Если в качестве параметра **size** задать нецелое число, то результатом будет **NaN**.

Пример 1. *Пусть проводится 10 испытаний Бернулли, вероятность «успеха» в каждом из них* -0.5. *Найдём сначала вероятности* $P\{\xi = k\}$ *получить ровно* k *успехов* ($k = \overline{0,10}$) *в 10 испытаниях.*

```
size=10; prob=0.5
dbinom(0:10,size,prob)
 [1] 0.0009765625 0.0097656250 0.0439453125 0.1171875000 0.2050781250
 [6] 0.2460937500 0.2050781250 0.1171875000 0.0439453125 0.0097656250
[11] 0.0009765625
```

Значения функции распределения $F(x)$.

```
pbinom(-1:10,size,prob)
 [1] 0.0000000000 0.0009765625 0.0107421875 0.0546875000 0.1718750000
 [6] 0.3769531250 0.6230468750 0.8281250000 0.9453125000 0.9892578125
[11] 0.9990234375 1.0000000000
```

Квантили:

```
qbinom(seq(0,1,by=0.05),size,prob)
 [1]  0  2  3  3  4  4  4  4  5  5  5  5  5  6  6  6  6  7  7  8 10
```

Добавим ещё один параметр **lower.tail** $=$ **F**:

```
qbinom(seq(0,1,by=0.05),size,prob,lower.tail = F)
 [1] 10  8  7  7  6  6  6  6  5  5  5  5  5  4  4  4  4  3  3  2  0
```

Построим случайную выборку из 20 элементов, подчиняющуюся биномиальному закону с указанными выше параметрами.

```
> set.seed(0)
> rbinom(20,size,prob)
 [1] 7 4 4 5 7 4 7 7 6 6 3 4 4 6 5 6 5 6 9 5
```

Функция **set.seed()** *позволяет генерировать при повторных обращениях одну и туже случайную выборку (задаётся начальный параметр генерирования).*

Приведём ещё несколько примеров, приводящих к нежелательному результату.

```
>dbinom(-2,size,prob)
[1] 0
>dbinom(2.2,size,prob)
[1] 0
Предупреждение
```

```
In dbinom(2.2, size, prob) : non-integer x = 2.200000
> dbinom(2,size=-7,prob)
[1] NaN
Предупреждение
In dbinom(x, size, prob, log) : созданы NaN
> pbinom(2,size=-7,prob)
[1] NaN
Предупреждение
In pbinom(q, size, prob, lower.tail, log.p) : созданы NaN
```

1.1.3 Пуассоновское распределение

Дискретная случайная величина ξ имеет распределение Пуассона с параметром λ, если

$$P\{\xi = i\} = \frac{\lambda^i}{i!}e^{-\lambda}.$$

Функции **R** отвечающие за распределения:

```
dpois(x, lambda, log = FALSE)
ppois(q, lambda, lower.tail = TRUE, log.p = FALSE)
qpois(p, lambda, lower.tail = TRUE, log.p = FALSE)
rpois(n, lambda)
```

Здесь: **x** — неотрицательный целочисленный вектор (отрицательные значения приводят к 0), **q** и **lambda** — неотрицательные аргументы. Заметим, что использование аргумента **lower.tail** = **FALSE**, позволяет получать более точные результаты (см. пример).

Пример 2. *Найдём* $P\{\xi < 150, \xi > 250\} = 1 - P\{150 \leqslant \xi \leqslant 250\}$, $\xi \sim Pois(\lambda = 100)$. *Первый способ (через функцию распределения):*

```
> 1 - (ppois(250,lambda=100)-ppois(149,lambda=100))
[1] 0.9999981
```

Второй способ (через сумму вероятностей):

```
> 1-sum(dpois(10*(15:25), lambda=100))
[1] 0.9999993
```

1.1.4 Геометрическое распределение

Случайная величина ξ, описывающая число «неудач» до появления первого «успеха» в последовательности испытаний Бернулли, подчиняется геометрическому закону с параметром p — вероятность «успеха» в одном испытании Бернулли.

$$P\{\xi = i\} = (1-p)^i p, \quad i \geqslant 0.$$

Функции в **R**:

```
dgeom(x, prob, log = FALSE)
pgeom(q, prob, lower.tail = TRUE, log.p = FALSE)
qgeom(p, prob, lower.tail = TRUE, log.p = FALSE)
rgeom(n, prob)
```

Здесь **prob** — вероятность «успеха» p, **x** — целочисленный неотрицательный вектор (отрицательные значения приводят к 0).

1.1.5 Гипергеометрическое распределение

Рассмотрим следующую задачу. В урне (корзине) имеется m белых и n чёрных шаров. Из урны без возвращения вынимают k шаров $(0 < k < n+m)$. Случайная величина ξ, описывающая число i $(0 \leqslant i \leqslant \min(k, m))$ вынутых белых шаров, подчиняется гипергеометрическому распределению.

$$P\{\xi = i\} = \frac{C_m^i C_n^{k-i}}{C_{n+m}^k}, \quad 0 \leqslant i \leqslant \min(k, m).$$

Функции в **R**:

```
dhyper(x, m, n, k, log = FALSE)
phyper(q, m, n, k, lower.tail = TRUE, log.p = FALSE)
qhyper(p, m, n, k, lower.tail = TRUE, log.p = FALSE)
rhyper(nn, m, n, k)
```

Здесь **x** — неотрицательное число (вектор) — количество вынутых (без возвращения) белых шаров, **m** и **n** — изначальное количество белых и чёрных шаров (неотрицательные аргументы), **k** — общее количество вынимаемых шаров (неотрицательный аргумент). Неправильное задание аргументов приведёт к **NaN** и предупреждению.

Пример 3. *Рассмотрим урну, в которой находится 20 шаров: 6 белых и 14 красных. Из урны без возвращения вынимают 6 шаров. Построим распределение с.в. ξ — число вынутых белых шаров.*

```
> ph=numeric(7)
> for(i in 0:6) ph[i+1]<-dhyper(i,6,14,6)
> ph
[1] 7.747678e-02 3.099071e-01 3.873839e-01 1.878225e-01 3.521672e-02
[6] 2.167183e-03 2.579979e-05
> sum(ph)
[1] 1
```

Построим мозаичную диаграмму (рис.1.1).

```
barplot(ph,names=as.character(0:6),ylim=c(0,0.4),density=16)
```

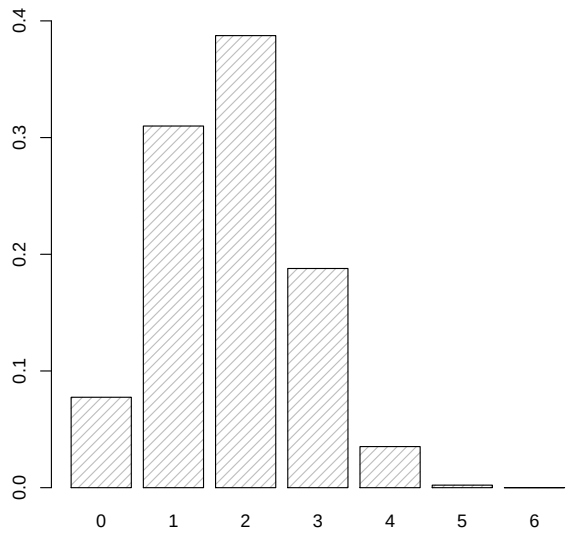

Рис. 1.1: Гипергеометрическое распределение

Как видно, наиболее вероятно извлечь 2 белых и 4 красных шара.

1.1.6 Отрицательно биномиальное распределение

Отрицательно биномиальное распределение принадлежит к двупараметрическим дискретным распределениям и используется для описания данных, у которых дисперсия намного больше математического ожидания.

Случайная величина, описывающая число «неудач» при достижении заданного числа «успехов», подчиняется отрицательно биномиальному распределению.

$$P\{\xi = i\} = C^i_{n+i-1} p^n (1-p)^i, \quad i \geqslant 0,\ n > 0.$$

Функции **R**.

```
dnbinom(x, size, prob, mu, log = FALSE)
pnbinom(q, size, prob, mu, lower.tail = TRUE, log.p = FALSE)
qnbinom(p, size, prob, mu, lower.tail = TRUE, log.p = FALSE)
rnbinom(n, size, prob, mu)
```

Аргументы:

- **x** — неотрицательное целое конечное число (вектор) — число «неудач» до появления заданного числа «успехов».
- **size** — неотрицательное конечное число — заданное число «успехов».
- **prob** — вероятность успеха в одном испытании.
- **mu** — альтернативный **prob** параметр (среднее значение). Используется в экологических моделях. Тогда вероятность определяется следующим образом **prob** = **size**/(**size**+**mu**).

Пример 4. *Пусть вероятность успеха в одном испытании Бернулли* $p = 0.2$. *Предположим, что надо получить 5 «успехов». Какое наиболее вероятное число «неудач» будет, прежде чем наступит пятый «успех».*

Сначала построим график вероятностей числа «неудач» (рис.1.2).

```
plot(0:100,dnbinom(0:100,5,0.2),type="s",xlab="x", ylab=expression(p[x]))
```

Найдём наивероятнейшее значение и соответствующую вероятность.

```
> y=dnbinom(0:100,5,0.2)
> which(y==max(y))
[1] 16
> max(y)
[1] 0.04363988
```

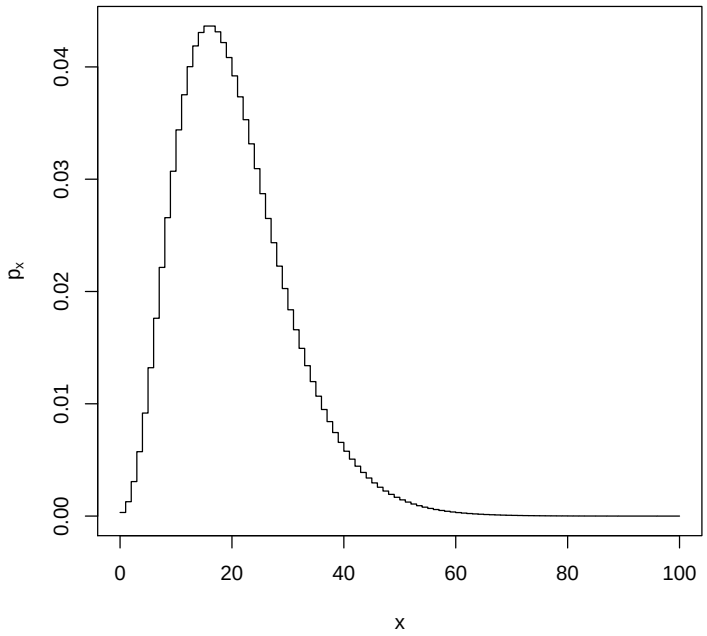

Рис. 1.2: Распределение числа «неудач» до наступления пятого «успеха» с вероятностью «успеха» $p = 0.2$

1.1.7 Полиномиальное распределение

Пусть имеется n предметов, каждый из которых может обладать только одним из k свойств с вероятностью p_i, $i = \overline{1,k}$. Вероятность того, что n_1 предмет обладает свойством 1, n_2 — свойством 2,..., n_k — свойством k, определяется формулой

$$P(n_1, n_2, \ldots, n_k) = \frac{n!}{n_1! \cdot \ldots \cdot n_k!} p_1^{n_1} \cdot \ldots \cdot p_k^{n_k}.$$

В **R** полиномиальное распределение реализовано всего двумя функциями:

```
dmultinom(x, size = NULL, prob, log = FALSE)
rmultinom(n, size, prob)
```

Аргументы:

- **x** — вектор из k неотрицательных целых чисел, таких что $x_1 + x_2 + \ldots + x_k = n$ (свойства предметов).
- **size** — общее количество предметов (n). По умолчанию **size=sum(x)**.

- **prob** — неотрицательный вектор длины k — вероятности $p_1, \ldots, p_k$. Если сумма элементов **prob** не равна 1, то аргумент **prob** будет нормирован.

- **n** — число создаваемых случайных векторов.

При вызове функции **rmultinom()** создаётся матрица размера $k \times n$, каждый столбец которой — случайный вектор, подчиняющийся полиномиальному распределению.

Пример 5. *Пусть имеются 4 предмета, каждый из которых может обладать одним из 3 свойств. Найдём всевозможные комбинации*

```
X = t(as.matrix(expand.grid(0:4, 0:4)))
X = X[, colSums(X) <= 4]
X = rbind(X, 4 - colSums(X))
dimnames(X) = list(letters[1:3], NULL)
X
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
a    0    1    2    3    4    0    1    2    3     0     1     2     0     1
b    0    0    0    0    0    1    1    1    1     2     2     2     3     3
c    4    3    2    1    0    3    2    1    0     2     1     0     1     0
  [,15]
a     0
b     4
c     0
```

и соответствующие им вероятности.

```
round(apply(X, 2, function(x) dmultinom(x, prob = c(1,2,5))), 4)
 [1] 0.1526 0.1221 0.0366 0.0049 0.0002 0.2441 0.1465 0.0293 0.0020 0.1465
[11] 0.0586 0.0059 0.0391 0.0078 0.0039
```

Построим случайную выборку размером 10, подчиняющуюся полиномиальному распределению, когда имеется 12 предметов, каждый из которых может обладать одним из трёх свойств.

```
rmultinom(10, size = 12, prob=c(0.1,0.2,0.7))
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    2    0    1    1    1    0    1    0    1     1
[2,]    2    2    2    4    2    4    0    1    5     4
[3,]    8   10    9    7    9    8   11   11    6     7
```

1.1.8 Бета-распределение

Бета-распределение относится к двупараметрическим распределениям. Плотность имеет вид:

$$p(x) = \begin{cases} 0, & x < 0 \text{ или } x > 1, \\ \dfrac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1}, & x \in [0;1]. \end{cases}$$

В **R** для бета-распределения предусмотрены следующие функции:

```
dbeta(x, shape1, shape2, ncp = 0, log = FALSE)
pbeta(q, shape1, shape2, ncp = 0, lower.tail = TRUE, log.p = FALSE)
qbeta(p, shape1, shape2, ncp = 0, lower.tail = TRUE, log.p = FALSE)
rbeta(n, shape1, shape2, ncp = 0)
```

Аргумент **shape1** соответствует параметру **a**, аргумент **shape2** — параметру **b**. Оба эти аргумента должны быть неотрицательными и конечными.

Математическое ожидание и дисперсия случайной величины:

$$M\xi = \frac{a}{a+b},$$

При $a, b > 1$ график плотности выпуклый вверх, при $0 < a < 1$ и $b > 1$ плотность убывает (график вогнут вниз), при $a > 1$ и $0 < b < 1$ плотность возрастает (график вогнут вниз), при $a, b < 1$ график плотности имеет форму буквы **U**. При $a = b = 1$ — график равномерного распределения.

Пример 6. *Построим графики бета-распределения с различными параметрами.*

```
x=seq(0,1,by=0.01);
y1=dbeta(x,0.5,0.6)
z1=pbeta(x,0.5,0.6)

op=par(mfrow=c(1,2))
plot(x,y1,type="l",xlab='x',ylab='p(x)',ylim=c(0,3))
curve(dbeta(x,1,0.5),add=T,lty=2)
curve(dbeta(x,0.5,1),add=T,lty=3)
curve(dbeta(x,1,1),add=T,lty=4)
curve(dbeta(x,3,5),add=T,lty=5)
curve(dbeta(x,5,3),add=T,lty=6)

legend(0.6,3,c(expression(beta (0.5,0.6)),expression(beta (1,0.5)),
```

```
expression(beta(0.5,1)), expression(beta(1,1)),expression(beta(3,5)),
expression(beta(5,3))),lty=1:6,cex=0.6)
title(main=expression(p(x)),xlab='x',ylab='p(x)')

plot(x,z1,type="l",xlab='x',ylab='F(x)',ylim=c(0,1.1),lty=1)
curve(pbeta(x,1,0.5),add=T,lty=2)
curve(pbeta(x,0.5,1),add=T,lty=3)
curve(pbeta(x,1,1),add=T,lty=4)
curve(pbeta(x,3,5),add=T,lty=5)
curve(pbeta(x,5,3),add=T,lty=6)
legend(0.6,0.3,c(expression(beta (0.5,0.6)),expression(beta (1,0.5)),
expression(beta(0.5,1)), expression(beta(1,1)),expression(beta(3,5)),
expression(beta(5,3))),lty=1:6,cex=0.6)
title(main=expression(F(x)))
par(op)
```

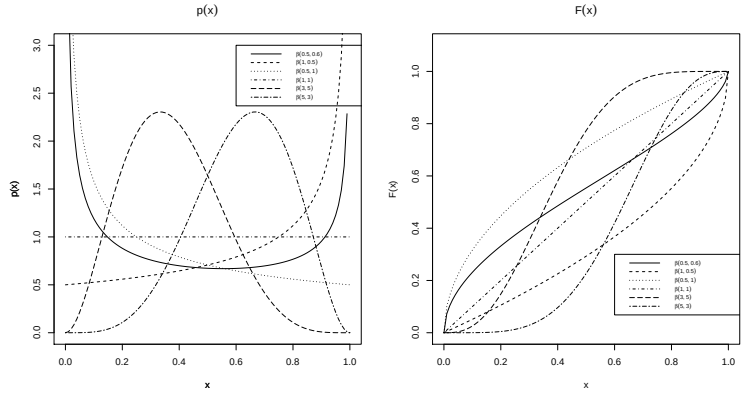

Рис. 1.3: Бета-распределение

Квантили:

```
> p=seq(0,1,by=0.1)
> qbeta(p,0.5,0.6)
 [1] 0.00000000 0.01914598 0.07540407 0.16526616 0.28298782 0.42073287
 [7] 0.56875203 0.71555172 0.84793008 0.95041263 1.00000000
> qbeta(p,1,0.5)
 [1] 0.00 0.19 0.36 0.51 0.64 0.75 0.84 0.91 0.96 0.99 1.00
> qbeta(p,0.5,1)
 [1] 0.00 0.01 0.04 0.09 0.16 0.25 0.36 0.49 0.64 0.81 1.00
> qbeta(p,1,1)
```

```
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> qbeta(p,3,5)
 [1] 0.0000000 0.1696384 0.2283265 0.2763397 0.3205858 0.3641161 0.4092151
 [8] 0.4585547 0.5167578 0.5961797 1.0000000
> qbeta(p,5,3)
 [1] 0.0000000 0.4038203 0.4832422 0.5414453 0.5907849 0.6358839 0.6794142
 [8] 0.7236603 0.7716735 0.8303616 1.0000000
```

1.1.9 Распределение Коши

Распределение Коши является двупараметрическим распределением с плотностью

$$p(x) = \frac{1}{\pi b\left(1 + \left(\frac{x-a}{b}\right)^2\right)},$$

параметр **a** — медиана (сдвиг по оси **0x**),а **b** — параметр масштаба ($b > 0$).

В **R**:

```
dcauchy(x, location = 0, scale = 1, log = FALSE)
pcauchy(q, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
qcauchy(p, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
rcauchy(n, location = 0, scale = 1)
```

Аргумент **location** соответствует параметру распределения **a**, а **scale** — **b**.

Пример 7. *Графики плотностей и ФР распределения Коши.*

```
x=seq(-5,5,by=0.01);
y1=dcauchy(x,-1,0.5)
y2=dcauchy(x,-1,2)
y3=dcauchy(x,0,1)
y4=dcauchy(x,1,0.5)
y5=dcauchy(x,1,2)
y6=dcauchy(x,1,3)
z1=pcauchy(x,-1,0.5)

op=par(mfrow=c(1,2))
plot(x,y1,type="l",xlab='x',ylab='p(x)',ylim=c(0,max(y1,y2,y3,y4,y5,y6)))
curve(dcauchy(x,-1,2),add=T,lty=2)
curve(dcauchy(x,0,1),add=T,lty=3)
curve(dcauchy(x,1,0.5),add=T,lty=4)
curve(dcauchy(x,1,2),add=T,lty=5)
curve(dcauchy(x,1,3),add=T,lty=6)
```

```
legend(1.8,0.6,c(expression(C(-1,0.5)),expression(C(-1,2)),
expression(C(0,1)), expression(C(1,0.5)),expression(C(1,2)),
expression(C(1,3))),lty=1:6,cex=0.7)
title(main=expression(p(x)),xlab='x',ylab='p(x)')

plot(x,z1,type="l",xlab='x',ylab='F(x)',ylim=c(0,1.1),lty=1)
curve(pcauchy(x,-1,2),add=T,lty=2)
curve(pcauchy(x,0,1),add=T,lty=3)
curve(pcauchy(x,1,0.5),add=T,lty=4)
curve(pcauchy(x,1,2),add=T,lty=5)
curve(pcauchy(x,1,3),add=T,lty=6)
legend(0.6,0.25,c(expression(C(-1,0.5)),expression(C(-1,2)),
expression(C(0,1)), expression(C(1,0.5)),expression(C(1,2)),
expression(C(1,3))),lty=1:6,cex=0.7)
title(main=expression(F(x)))
par(op)
```

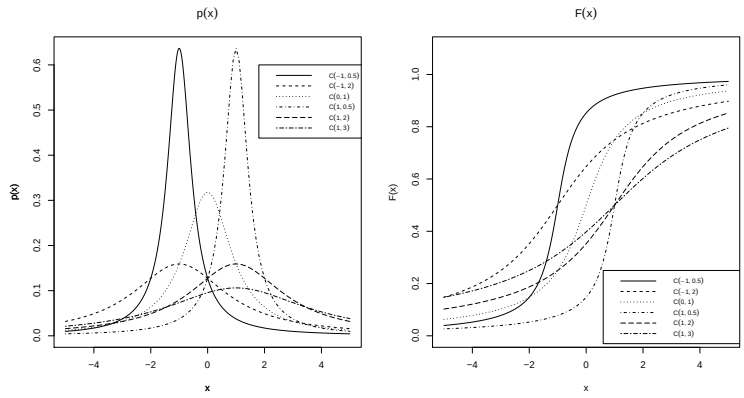

Рис. 1.4: Распределение Коши

Квантили:

```
> p=seq(0,1,by=0.1)
> qcauchy(p,-1,0.5)
 [1]       -Inf -2.5388418 -1.6881910 -1.3632713 -1.1624598 -1.0000000
 [7] -0.8375402 -0.6367287 -0.3118090  0.5388418        Inf
> qcauchy(p,-1,2)
 [1]       -Inf -7.1553671 -3.7527638 -2.4530851 -1.6498394 -1.0000000
 [7] -0.3501606  0.4530851  1.7527638  5.1553671        Inf
```

```
> qcauchy(p,0,1)
 [1]          -Inf -3.077684e+00 -1.376382e+00 -7.265425e-01 -3.249197e-01
 [6] -6.123032e-17  3.249197e-01  7.265425e-01  1.376382e+00  3.077684e+00
[11]           Inf
> qcauchy(p,1,0.5)
 [1]       -Inf -0.5388418  0.3118090  0.6367287  0.8375402  1.0000000
 [7]  1.1624598  1.3632713  1.6881910  2.5388418        Inf
> qcauchy(p,1,2)
 [1]       -Inf -5.1553671 -1.7527638 -0.4530851  0.3501606  1.0000000
 [7]  1.6498394  2.4530851  3.7527638  7.1553671        Inf
```

1.1.10 Экспоненциальное распределение

Экспоненциальное распределение реализовано в **R** при помощи функций

```
dexp(x, rate = 1, log = FALSE)
pexp(q, rate = 1, lower.tail = TRUE, log.p = FALSE)
qexp(p, rate = 1, lower.tail = TRUE, log.p = FALSE)
rexp(n, rate = 1)
```

Аргумент **rate** соответствует параметру λ экспоненциального распределения.

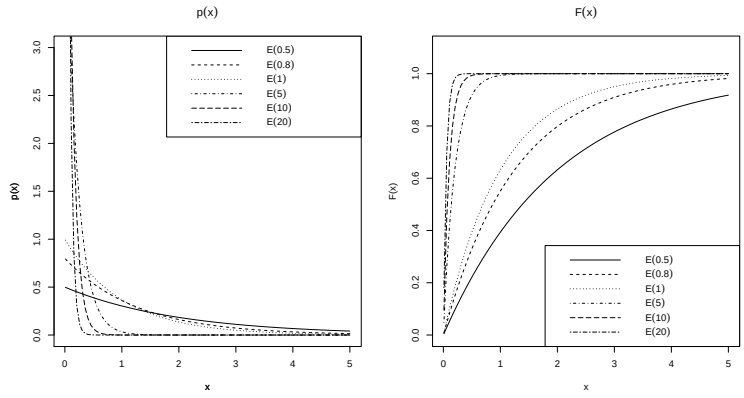

Рис. 1.5: Графики плотностей и ФР экспоненциального распределения с различными значениями λ

Замечания:

- **x**, **q** и **rate** — неотрицательные величины.
- **rate** может принимать значение **Inf**.

1.1.11 χ^2-распределение

Распределение χ^2 — частный случай гамма-распределения, зависящее от одного параметра — числа степеней свободы ν. Плотность χ^2-распределения:

$$p(x) = \begin{cases} 0, & x < 0, \\ \dfrac{1}{2^{\nu/2}\Gamma(\frac{\nu}{2})} x^{\frac{\nu}{2}-1} e^{-\frac{x}{2}}, & x \geqslant 0. \end{cases}$$

χ^2-распределение является важным потому, что многие квадратичные формы (суммы квадратов) случайных величин подчиняются данному закону, если исходные случайные величины имеют нормальное распределение.

Четыре функции в **R** отвечают за χ^2-распределение. Функция

```
dchisq(x, df, ncp=0, log = FALSE)
```

отвечает за плотность распределения случайной величины.

```
pchisq(q, df, ncp=0, lower.tail = TRUE, log.p = FALSE)
```

— функция распределения случайной величины.

```
qchisq(p, df, ncp=0, lower.tail = TRUE, log.p = FALSE)
```

— вычисляются квантили.

```
rchisq(n, df, ncp=0)
```

— построение случайной выборки.

Остановимся только на аргументах функций:

- **x** и **q** — неотрицательные числовые вектора.
- **df** — параметр χ^2-распределения — число степеней свободы. Положительное (не обязательно целое) число.
- **ncp** — параметр смещения (неотрицательное число).

Смещённое χ^2-распределение с **df**= ν степенями свободы и параметром смещения **ncp**= λ имеет плотность вида

$$p(x) = e^{-\frac{\lambda}{2}} \sum_{r=0}^{\infty} \frac{\left(\frac{\lambda}{2}\right)^r}{r!} p_{\nu+2r}(x),$$

где $p_{n+2r}(x)$ — несмещённое χ^2-распределение с $\nu + 2r$ степенями свободы; аргумент x больше или равен нулю.

Математическое распределение и дисперсия случайной величины, имеющей смещённое χ^2-распределение с параметром смещения λ и ν степенями свободы:

$$M\xi = \nu + \lambda, \quad D\xi = 2(\nu + 2\lambda).$$

Замечание. Неправильное задание аргументов **x** и **q** (отрицательные значения) приведёт к появлению **NaN**.

Пример 8. *Построим графики плотностей и ФР несмещённого и смещённого χ^2-распределения с различными степенями свободы.*

```
x=seq(0,5,by=0.01);
y1=dchisq(x,2)
z1=pchisq(x,2)
#создание графического окна
par(mfrow=c(1,2))
#построение графиков плотностей
plot(x,y1,type="l",col="red",xlab='x',ylab='p(x)')
curve(dchisq(x,1),col="blue",add=T)
curve(dchisq(x,2,ncp=1),col="violet",add=T)
curve(dchisq(x,1,ncp=1),col="orange",add=T)
legend(2,0.5,c(expression({chi^{2}}(nu=2,lambda=0)),
expression({chi^{2}}(nu=1,lambda=0)), expression({chi^{2}}(nu=2,lambda=1)),
expression({chi^{2}}(nu=2,lambda=1))),
col=c("red","blue","violet","orange"),lty=1)
title(main=expression(p[chi^2](x)),xlab='x',ylab='p(x)')
#построение графиков ФР
plot(x,z1,type="l",col="red",xlab='x',ylab='p(x)')
curve(pchisq(x,1),col="blue",add=T)
curve(pchisq(x,2,ncp=1),col="violet",add=T)
curve(pchisq(x,1,ncp=1),col="orange",add=T)
legend(2,0.4,c(expression({chi^{2}}(nu=2,lambda=0)),
expression({chi^{2}}(nu=1,lambda=0)), expression({chi^{2}}(nu=2,lambda=1)),
expression({chi^{2}}(nu=2,lambda=1))),
col=c("red","blue","violet","orange"),lty=1)
title(main=expression(F[chi^2](x)))
```

Найдём квантили χ^2-распределения.

```
qchisq(seq(0,1,by=0.1),2)
 [1] 0.0000000 0.2107210 0.4462871 0.7133499 1.0216512 1.3862944 1.8325815
 [8] 2.4079456 3.2188758 4.6051702       Inf
> qchisq(seq(0,1,by=0.1),2,ncp=1)
```

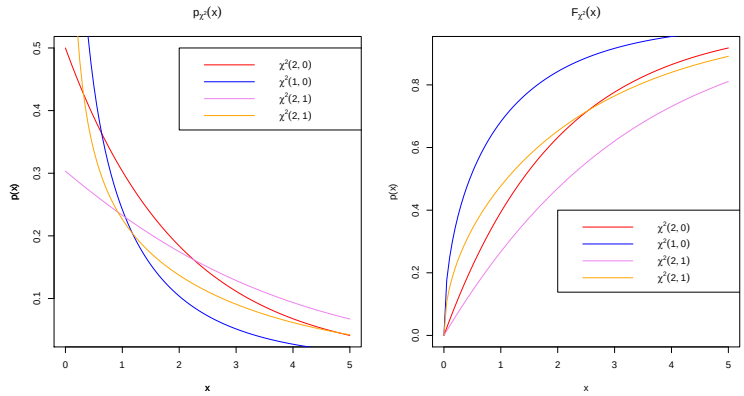

Рис. 1.6: Графики плотностей и ФР χ^2-распределения с различными параметрами

```
 [1] 0.0000000 0.3443433 0.7226199 1.1440323 1.6219662 2.1770386 2.8435615
 [8] 3.6853997 4.8447326 6.7701303       Inf
> qchisq(seq(0,1,by=0.1),1)
 [1] 0.00000000 0.01579077 0.06418475 0.14847186 0.27499590 0.45493642
 [7] 0.70832630 1.07419417 1.64237442 2.70554345        Inf
> qchisq(seq(0,1,by=0.1),1,ncp=1)
 [1] 0.00000000 0.04270125 0.17095587 0.38606908 0.69287886 1.10364331
 [7] 1.64486833 2.37280560 3.42053299 5.21879410        Inf
```

В завершение примера построим случайную выборку.

```
> set.seed(0)
> rchisq(24,1,ncp=0)
 [1] 2.81619199 3.05246608 2.84923084 0.01070220 0.08738922 0.41360540
 [7] 0.69442147 7.90929829 0.04419345 0.41794630 0.32474175 0.09721441
[13] 0.03266504 0.47419022 1.74676814 0.63847628 1.34529994 0.16799732
[19] 0.02773575 0.30974000 0.18663156 2.44004138 1.97546282 0.31231771
> set.seed(0)
> rchisq(24,1,ncp=1)
 [1] 3.362748792 5.638080875 1.661467023 1.562226888 0.409918567 0.682867104
 [7] 1.743724239 0.786649427 0.638476284 1.090194509 1.228603696 3.005186825
[13] 1.646261888 0.811537011 0.448535615 0.338408517 0.009737032 2.963270864
[19] 1.761512343 6.183818024 5.227405633 1.435451925 1.597265321 0.016649322
```

1.1.12 Распределение Фишера

Распределение названо в честь R.A. Fisher, основателя дисперсионного анализа.

Плотность распределения Фишера:

$$p(x) = \begin{cases} 0, & x < 0, \\ \dfrac{r\Gamma\left(\frac{1}{2(r+s)}\right)}{s\Gamma\left(\frac{1}{2r}\right)\Gamma\left(\frac{1}{2s}\right)} \cdot \dfrac{\left(\frac{rx}{s}\right)^{(r-1)/2}}{\left(1+\frac{rx}{s}\right)^{(r+s)/2}}, & x > 0, \end{cases}$$

где r и s — число степеней свободы.

Как и для большинства вероятностных распределений в **R** для распределения Фишера определены четыре функции:

```
df(x, df1, df2, ncp, log = FALSE)
pf(q, df1, df2, ncp, lower.tail = TRUE, log.p = FALSE)
qf(p, df1, df2, ncp, lower.tail = TRUE, log.p = FALSE)
rf(n, df1, df2, ncp)
```

Остановимся только на трёх аргументах:

- **df1** и **df2** — числа степеней свободы (r и s, соответственно). В качестве значений допускается **Inf**.
- **ncp** — параметр смещения.

Пример 9. *Снова построим графики плотностей распределения и ФР для различных значений параметров r и s.*

```
x=seq(0.01,5,by=0.01);
y1=df(x,1,10)
z1=pf(x,1,10)

op=par(mfrow=c(2,1))
plot(x,y1,type="l",xlab='x',ylab='p(x)',ylim=c(0,1))
curve(df(x,3,10),add=T,lty=2)
curve(df(x,5,10),add=T,lty=3)
curve(df(x,10,5),add=T,lty=4)
curve(df(x,Inf,5),add=T,lty=5)
curve(df(x,Inf,Inf),add=T,lty=6)

legend('topright',c(expression(F(1,10)),expression(F(3,10)),
expression(F(5,10)), expression(F(10,5)),expression(F(Inf,5)),
```

```
expression(F(Inf,Inf))),lty=1:6)
title(main=expression(p(x)),xlab='x',ylab='p(x)')

plot(x,z1,type="l",xlab='x',ylab='F(x)',ylim=c(0,1.1),lty=1)
curve(pf(x,3,10),add=T,lty=2)
curve(pf(x,5,10),add=T,lty=3)
curve(pf(x,10,5),add=T,lty=4)
curve(pf(x,Inf,5),add=T,lty=5)
curve(pf(x,Inf,Inf),add=T,lty=6)
legend('bottomright',c(expression(F(1,10)),expression(F(3,10)),
expression(F(5,10)), expression(F(10,5)),expression(F(Inf,5)),
expression(F(Inf,Inf))),lty=1:6)
title(main=expression(F(x)))
par(op)
```

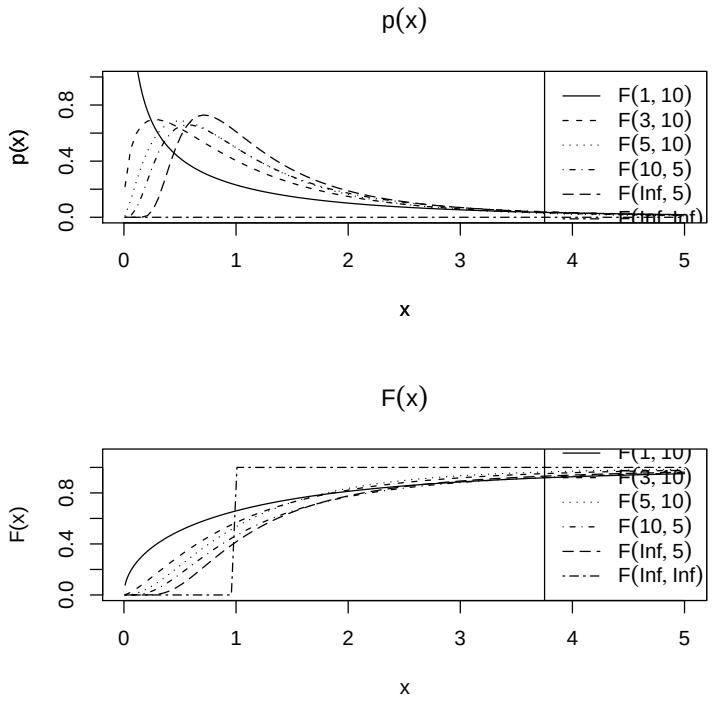

Рис. 1.7: Графики плотностей и ФР распределения Фишера при различных значениях параметров

Квантили:

```
> qf(p,1,10)
```

```
 [1] 0.00000000 0.01661268 0.06769615 0.15728481 0.29325262 0.48973692
 [7] 0.77274267 1.19477595 1.88288794 3.28501532        Inf
> qf(p,3,10)
 [1] 0.0000000 0.1911896 0.3356740 0.4852835 0.6513249 0.8450806 1.0834150
 [8] 1.3977420 1.8613988 2.7276731       Inf
> qf(p,5,10)
 [1] 0.0000000 0.3032691 0.4563364 0.6026031 0.7573473 0.9319332 1.1411893
 [8] 1.4113804 1.8027238 2.5216407       Inf
> qf(p,10,5)
 [1] 0.0000000 0.3965672 0.5547161 0.7085262 0.8762788 1.0730383 1.3203981
 [8] 1.6594670 2.1913657 3.2974017       Inf
> qf(p,Inf,5)
 [1] 0.0000000 0.5413390 0.6859392 0.8244798 0.9743043 1.1490396 1.3678021
 [8] 1.6667177 2.1344405 3.1049961       Inf
> qf(p,Inf,Inf)
 [1]   0   1   1   1   1   1   1   1   1   1 Inf
```

Построим случайную выборку для $\xi \sim F(\infty, \infty)$.

```
> set.seed(0)
> rf(24,Inf,Inf)
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

1.1.13 Гамма-распределение

Гамма-распределение имеет следующую плотность:

$$p(x) = \begin{cases} 0, & x < 0, \\ \dfrac{\lambda^\gamma x^{\gamma-1}}{\Gamma(\gamma)} e^{-\lambda x}, & x \geqslant 0, \end{cases}$$

и реализуется в **R** с помощью функций:

```
dgamma(x, shape, rate = 1, scale = 1/rate, log = FALSE)
pgamma(q, shape, rate = 1, scale = 1/rate, lower.tail = TRUE,
       log.p = FALSE)
qgamma(p, shape, rate = 1, scale = 1/rate, lower.tail = TRUE,
       log.p = FALSE)
rgamma(n, shape, rate = 1, scale = 1/rate)
```

Аргумент **shape** — это параметр γ и отвечает за форму, а аргументы **rate** и **scale** соответствуют λ и $1/\lambda$, соответственно (нужно задавать либо **rate**, либо **scale**).

Стоит отметить, что аргументы **x**, **q**, **shape** и **scale** (**rate**) должны быть неотрицательными (**shape** и **scale** не могут принимать значения **NaN**). В противном случае будут выдаваться **NaN**.

При $\gamma = 1$ получим экспоненциальное распределение с параметром λ.

Пример 10.
```
x=seq(0.01,5,by=0.01);
y1=dgamma(x,0.5,0.5)
z1=pgamma(x,0.5,0.5)

op=par(mfrow=c(1,2))
plot(x,y1,type="l",xlab='x',ylab='p(x)',ylim=c(0,3))
curve(dgamma(x,0.8,0.8),add=T,lty=2)
curve(dgamma(x,1,1),add=T,lty=3)
curve(dgamma(x,5,5),add=T,lty=4)
curve(dgamma(x,10,10),add=T,lty=5)
curve(dgamma(x,10,20),add=T,lty=6)

legend('topright',c(expression(G(0.5,0.5)),expression(G(0.8,0.8)),
expression(G(1,1)), expression(G(5,5)),expression(G(10,10)),
expression(G(10,20))),lty=1:6)
title(main=expression(p(x)),xlab='x',ylab='p(x)')

plot(x,z1,type="l",xlab='x',ylab='F(x)',ylim=c(0,1.1),lty=1)
curve(pgamma(x,0.8,0.8),add=T,lty=2)
curve(pgamma(x,1,1),add=T,lty=3)
curve(pgamma(x,5,5),add=T,lty=4)
curve(pgamma(x,10,10),add=T,lty=5)
curve(pgamma(x,10,20),add=T,lty=6)
legend('bottomright',c(expression(G(0.5,0.5)),expression(G(0.8,0.8)),
expression(G(1,1)), expression(G(5,5)),expression(G(10,10)),
expression(G(10,20))),lty=1:6)
title(main=expression(F(x)))
par(op)
```

Квантили

```
> qgamma(p,0.5,0.5)
 [1] 0.00000000 0.01579077 0.06418475 0.14847186 0.27499590 0.45493642
 [7] 0.70832630 1.07419417 1.64237442 2.70554345        Inf
> qgamma(p,0.8,0.8)
 [1] 0.00000000 0.06622728 0.16441062 0.28773972 0.43929418 0.62668903
 [7] 0.86408907 1.17901903 1.63421380 2.43157302        Inf
> qgamma(p,1,1)
```

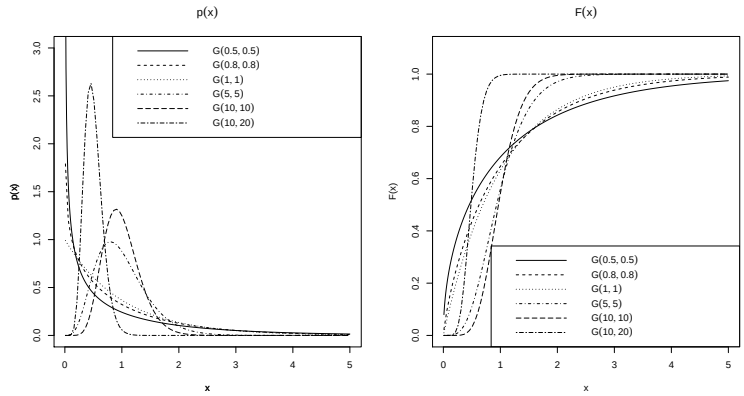

Рис. 1.8: Гамма-распределение: плотность и ФР

```
 [1] 0.0000000 0.1053605 0.2231436 0.3566749 0.5108256 0.6931472 0.9162907
 [8] 1.2039728 1.6094379 2.3025851       Inf
> qgamma(p,5,5)
 [1] 0.0000000 0.4865182 0.6179079 0.7267218 0.8295472 0.9341818 1.0473236
 [8] 1.1780723 1.3441958 1.5987179       Inf
> qgamma(p,10,10)
 [1] 0.0000000 0.6221305 0.7289220 0.8132928 0.8904415 0.9668715 1.0475684
 [8] 1.1387273 1.2518753 1.4205990       Inf
> qgamma(p,10,20)
 [1] 0.0000000 0.3110652 0.3644610 0.4066464 0.4452207 0.4834357 0.5237842
 [8] 0.5693636 0.6259376 0.7102995       Inf
```

1.1.14 Логнормальное распределение

Пусть случайная величина $\xi \sim N(m, \sigma)$, тогда с.в. $\eta = \ln \xi$ имеет логнормальное распределение с теми же параметрами . Плотность логнормального распределения:

$$p(x) = \begin{cases} \dfrac{1}{\sqrt{2\pi}\sigma x} e^{-\frac{(\ln x - m)^2}{2\sigma^2}}, & x > 0, \\ 0, & x \geqslant 0. \end{cases}$$

Математическое ожидание и дисперсия:

$$M\eta = e^{m+\sigma^2/2}, \quad D\eta = e^{2m+\sigma^2} \cdot \left(e^{\sigma^2} - 1\right)$$

В **R** реализованы функции:

```
dlnorm(x, meanlog = 0, sdlog = 1, log = FALSE)
```

```
plnorm(q, meanlog = 0, sdlog = 1, lower.tail = TRUE, log.p = FALSE)
qlnorm(p, meanlog = 0, sdlog = 1, lower.tail = TRUE, log.p = FALSE)
rlnorm(n, meanlog = 0, sdlog = 1)
```

Аргументы **meanlog** и **sdlog** задают параметры распределения (конечные значения).

Пример 11. *Графики плотности и ФР.*

```
x=seq(0,15,by=0.01);
y1=dlnorm(x,-1,0.5)
z1=plnorm(x,-1,0.5)

op=par(mfrow=c(1,2))
plot(x,y1,type="l",xlab='x',ylab='p(x)',ylim=c(0,2.5))
curve(dlnorm(x,-1,2),add=T,lty=2)
curve(dlnorm(x,0,1),add=T,lty=3)
curve(dlnorm(x,1,0.5),add=T,lty=4)
curve(dlnorm(x,1,2),add=T,lty=5)
curve(dlnorm(x,1,3),add=T,lty=6)

legend(3,2.5,c(expression(LN(-1,0.5)),expression(LN(-1,2)),
expression(LN(0,1)), expression(LN(1,0.5)),expression(LN(1,2)),
expression(LN(1,3))),lty=1:6,cex=0.7)
title(main=expression(p(x)),xlab='x',ylab='p(x)')

plot(x,z1,type="l",xlab='x',ylab='F(x)',ylim=c(0,1.1),lty=1)
curve(plnorm(x,-1,2),add=T,lty=2)
curve(plnorm(x,0,1),add=T,lty=3)
curve(plnorm(x,1,0.5),add=T,lty=4)
curve(plnorm(x,1,2),add=T,lty=5)
curve(plnorm(x,1,3),add=T,lty=6)
legend(3,0.3,c(expression(LN(-1,0.5)),expression(LN(-1,2)),
expression(LN(0,1)), expression(LN(1,0.5)),expression(LN(1,2)),
expression(LN(1,3))),lty=1:6,cex=0.7)
title(main=expression(F(x)))
par(op)
```

Квантили.

```
> p=seq(0,1,by=0.1)
> qlnorm(p,-1,0.5)
 [1] 0.0000000 0.1938296 0.2415182 0.2830306 0.3241096 0.3678794 0.4175602
```

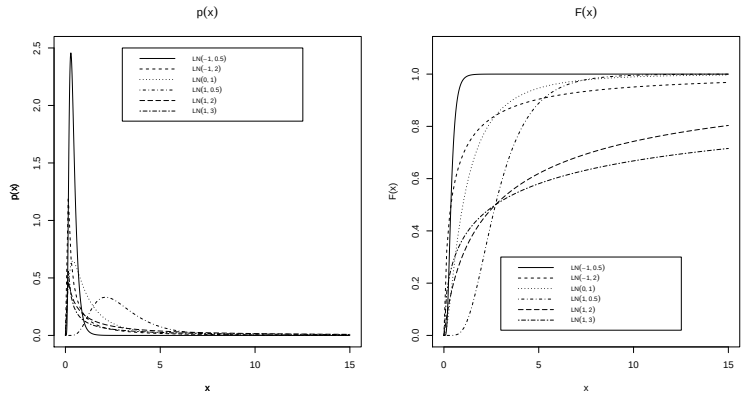

Рис. 1.9: Логнормальное распределение

```
 [8] 0.4781648 0.5603524 0.6982178       Inf
> qlnorm(p,-1,2)
 [1] 0.00000000 0.02835071 0.06834120 0.12888935 0.22164147 0.36787944
 [7] 0.61060452 1.05001140 1.98028835 4.77361143        Inf
> qlnorm(p,0,1)
 [1] 0.0000000 0.2776062 0.4310112 0.5919101 0.7761984 1.0000000 1.2883304
 [8] 1.6894457 2.3201254 3.6022245       Inf
> qlnorm(p,1,0.5)
 [1] 0.000000 1.432218 1.784591 2.091329 2.394864 2.718282 3.085376 3.533187
 [9] 4.140475 5.159170      Inf
> qlnorm(p,1,2)
 [1]  0.0000000  0.2094850  0.5049770  0.9523706  1.6377212  2.7182818
 [7]  4.5117911  7.7585932 14.6324617 35.2724826        Inf
```

1.1.15 Логистическое распределение

Логистическое распределение используется в обобщённых линейных моделях с биномиальными ошибками.

Функция распределения и плотность:

$$F(x) = \frac{1}{1+e^{-(x-m)/s}}, \quad p(x) = \frac{1}{s}\frac{e^{(x-m)/s}}{\left(1+e^{(x-m)/s}\right)^2}.$$

Математическое ожидание и дисперсия:

$$M\xi = m, \quad D\xi = \frac{\pi^2}{3}s^2.$$

В **R**:

```
dlogis(x, location = 0, scale = 1, log = FALSE)
plogis(q, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
qlogis(p, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
rlogis(n, location = 0, scale = 1)
```

Аргумент функции **location** соответствует параметру распределения **m** (параметр сдвига) и принимает конечные числовые значения, аргумент **scale** — параметр **s** (разброс) — положительные числовые значения. По умолчанию берутся значения **location** $= \mathbf{0}$ и **scale** $= \mathbf{1}$.

Пример 12. *Постоим графики плотностей и ФР логистического распределения для различных значений параметров.*

```
x=seq(-5,5,by=0.01);
y1=dlogis(x,-1,0.5)
z1=plogis(x,-1,0.5)

op=par(mfrow=c(1,2))
plot(x,y1,type="l",xlab='x',ylab='p(x)',ylim=c(0,0.6))
curve(dlogis(x,-1,2),add=T,lty=2)
curve(dlogis(x,0,1),add=T,lty=3)
curve(dlogis(x,1,0.5),add=T,lty=4)
curve(dlogis(x,1,2),add=T,lty=5)
curve(dlogis(x,1,3),add=T,lty=6)

legend(2,0.6,c(expression(L(-1,0.5)),expression(L(-1,2)),
expression(L(0,1)), expression(L(1,0.5)),expression(L(1,2)),
expression(L(1,3))),lty=1:6,cex=0.7)
title(main=expression(p(x)),xlab='x',ylab='p(x)')

plot(x,z1,type="l",xlab='x',ylab='F(x)',ylim=c(0,1.1),lty=1)
curve(plogis(x,-1,2),add=T,lty=2)
curve(plogis(x,0,1),add=T,lty=3)
curve(plogis(x,1,0.5),add=T,lty=4)
curve(plogis(x,1,2),add=T,lty=5)
curve(plogis(x,1,3),add=T,lty=6)
legend(3,0.3,c(expression(L(-1,0.5)),expression(L(-1,2)),
expression(L(0,1)), expression(L(1,0.5)),expression(L(1,2)),
expression(L(1,3))),lty=1:6,cex=0.7)
title(main=expression(F(x)))
par(op)
```

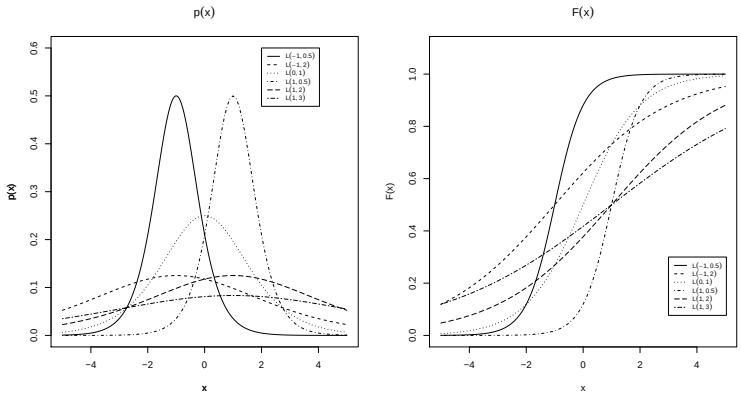

Рис. 1.10: Графики плотностей и ФР логистического распределения

Квантили.

```
> p=seq(0,1,by=0.1)
> qlogis(p,-1,0.5)
 [1]        -Inf -2.09861229 -1.69314718 -1.42364893 -1.20273255 -1.00000000
 [7] -0.79726745 -0.57635107 -0.30685282  0.09861229         Inf
> qlogis(p,-1,2)
 [1]       -Inf -5.3944492 -3.7725887 -2.6945957 -1.8109302 -1.0000000
 [7] -0.1890698  0.6945957  1.7725887  3.3944492        Inf
> qlogis(p,0,1)
 [1]       -Inf -2.1972246 -1.3862944 -0.8472979 -0.4054651  0.0000000
 [7]  0.4054651  0.8472979  1.3862944  2.1972246        Inf
> qlogis(p,1,0.5)
 [1]        -Inf -0.09861229  0.30685282  0.57635107  0.79726745  1.00000000
 [7]  1.20273255  1.42364893  1.69314718  2.09861229         Inf
> qlogis(p,1,2)
 [1]       -Inf -3.3944492 -1.7725887 -0.6945957  0.1890698  1.0000000
 [7]  1.8109302  2.6945957  3.7725887  5.3944492        Inf
> qlogis(p,1,3)
 [1]       -Inf -5.5916737 -3.1588831 -1.5418936 -0.2163953  1.0000000
 [7]  2.2163953  3.5418936  5.1588831  7.5916737        Inf
```

1.1.16 Нормальное распределение

Нормальное распределение — это центральное распределение в теории вероятностей и математической статистике.

Плотность нормального распределения:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-m)^2}{2\sigma^2}},$$

где m — математическое ожидание нормального закона, а σ — среднее квадратичное отклонение.

В **R** за нормальное распределение отвечают функции:

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

Аргументы:

- **x** и **q** — значения (квантили) случайной величины.
- **mean** и **sd** — параметры нормального закона — математическое ожидание (m) и среднее квадратичное отклонение (σ).
- **p** — вероятности.
- **n** — объём создаваемой случайной выборки.

Замечания:

- Заданный порядок аргументов функций является обязательным.
- Для функции **dnorm()** обязательным параметром является только **x**, для **pnorm()** — **q**, для **qnorm()** — **p** и для **rnorm()** — **n**. В этом случае используется стандартное нормальное распределение.

Пример 13. *Построим графики плотностей и ФР для различных значений параметров* m *и* σ*, найдём квантили и построим случайную выборку заданного размера* **n**.

```
x=seq(-5,5,by=0.1);
y1=dnorm(x)
y2=dnorm(x,1,1)
y3=dnorm(x,mean=-1,sd=2)
z1=pnorm(x)

op=par(mfrow=c(2,1))
plot(x,y1,type="l",col="red",xlab='x',ylab='p(x)',ylim=c(0,max(y1,y2,y3)))
```

```
curve(dnorm(x,1,1),col="blue",add=T)
curve(dnorm(x,mean=-1,sd=2),col="violet",add=T)
legend('topright',c(expression(N(0,1)),
expression(N(1,1)), expression(N(-1,2))),
col=c("red","blue","violet"),lty=1)
title(main=expression(p(x)),xlab='x',ylab='p(x)')

plot(x,z1,type="l",col="red",xlab='x',ylab='F(x)',ylim=c(0,1.1))
curve(pnorm(x,1,1),col="blue",add=T)
curve(pnorm(x,sd=2,mean=-1),col="violet",add=T)
legend('bottomright',c(expression(N(0,1)),
expression(N(1,1)), expression(N(-1,2))),
col=c("red","blue","violet"),lty=1)
title(main=expression(F(x)))
par(op)
```

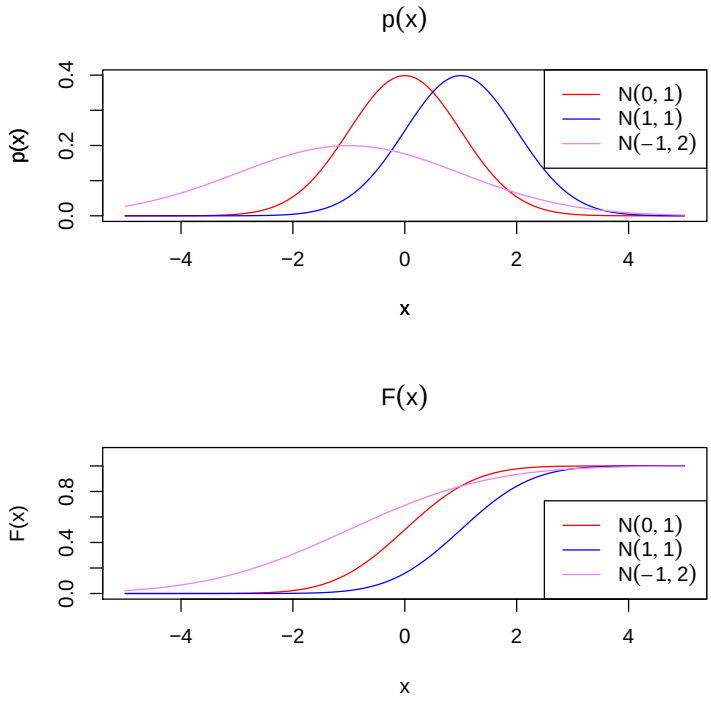

Рис. 1.11: Нормальное распределение

Квантили и случайная выборка.

```
qnorm(seq(0,1,by=0.1))
```

```
 [1]       -Inf -1.2815516 -0.8416212 -0.5244005 -0.2533471  0.0000000
 [7]  0.2533471  0.5244005  0.8416212  1.2815516        Inf
qnorm(seq(0,1,by=0.1),1,1)
 [1]       -Inf -0.2815516  0.1583788  0.4755995  0.7466529  1.0000000
 [7]  1.2533471  1.5244005  1.8416212  2.2815516        Inf
qnorm(seq(0,1,by=0.1),-1,1)
 [1]       -Inf -2.2815516 -1.8416212 -1.5244005 -1.2533471 -1.0000000
 [7] -0.7466529 -0.4755995 -0.1583788  0.2815516        Inf
set.seed(0)
rnorm(24)
 [1]  1.262954285 -0.326233361  1.329799263  1.272429321  0.414641434
 [6] -1.539950042 -0.928567035 -0.294720447 -0.005767173  2.404653389
[11]  0.763593461 -0.799009249 -1.147657009 -0.289461574 -0.299215118
[16] -0.411510833  0.252223448 -0.891921127  0.435683299 -1.237538422
[21] -0.224267885  0.377395646  0.133336361  0.804189510
set.seed(0)
rchisq(24,1,1)
 [1] 3.362748792 5.638080875 1.661467023 1.562226888 0.409918567 0.682867104
 [7] 1.743724239 0.786649427 0.638476284 1.090194509 1.228603696 3.005186825
[13] 1.646261888 0.811537011 0.448535615 0.338408517 0.009737032 2.963270864
[19] 1.761512343 6.183818024 5.227405633 1.435451925 1.597265321 0.016649322
```

1.1.17 Распределение Стьюдента (t - распределение)

Это распределение было опубликовано W.S. Gossett (под псевдонимом Student) в 1908 году. Псевдоним был взят потому, что его работодатель (пивоваренная компания Guinness)запрещал свои работником публиковать работы под своими фамилиями.

Плотность распределения Стьюдента:

$$p(x) = \frac{\Gamma\left(\frac{1}{2(r+1)}\right)}{\sqrt{\pi r}\Gamma\left(\frac{1}{2r}\right)}\left(1+\frac{x^2}{r}\right)^{-(r+1)/2}.$$

Математическое ожидание и дисперсия:

$$M\xi = 0, \text{ при } r > 1, \quad D\xi = \frac{r}{r-2}, \text{ при } r > 2.$$

Функции в **R**:

```
dt(x, df, ncp, log = FALSE)
```

```
pt(q, df, ncp, lower.tail = TRUE, log.p = FALSE)
qt(p, df, ncp, lower.tail = TRUE, log.p = FALSE)
rt(n, df, ncp)
```

Из незнакомых аргументов только **df** — число степеней свободы для распределения Стьюдента (**r**) (положительное число, может быть **Inf**).

Пример 14.
```
x=seq(-5,5,by=0.01);
y1=dt(x,1)
y2=dt(x,3)
y3=dt(x,5)
y4=dt(x,10)
y5=dt(x,5.5)
z1=pt(x,1)

op=par(mfrow=c(1,2))
plot(x,y1,type="l",xlab='x',ylab='p(x)',ylim=c(0,max(y1,y2,y3,y4,y5)))
curve(dt(x,3),add=T,lty=2)
curve(dt(x,5),add=T,lty=3)
curve(dt(x,10),add=T,lty=4)
curve(dt(x,5.5),add=T,lty=5)
curve(dt(x,Inf),add=T,lty=6)

legend(1,0.4,c(expression(t(1)),expression(t(3)),
expression(t(5)), expression(t(10)),expression(t(5.5)),
expression(t(Inf))),lty=1:6)
title(main=expression(p(x)),xlab='x',ylab='p(x)')

plot(x,z1,type="l",xlab='x',ylab='F(x)',ylim=c(0,1.1),lty=1)
curve(pt(x,3),add=T,lty=2)
curve(pt(x,5),add=T,lty=3)
curve(pt(x,10),add=T,lty=4)
curve(pt(x,5.5),add=T,lty=5)
curve(pt(x,Inf),add=T,lty=6)
legend('bottomright',c(expression(t(1)),expression(t(3)),
expression(t(5)), expression(t(10)),expression(t(5.5)),
expression(t(Inf))),lty=1:6)
title(main=expression(F(x)))
par(op)
```

Квантили:

```
> p=seq(0,1,by=0.1)
```

```
> qt(p,1)
 [1]          -Inf -3.077684e+00 -1.376382e+00 -7.265425e-01 -3.249197e-01
 [6]  6.123032e-17  3.249197e-01  7.265425e-01  1.376382e+00  3.077684e+00
[11]           Inf
> qt(p,3)
 [1]       -Inf -1.6377444 -0.9784723 -0.5843897 -0.2766707  0.0000000
 [7]  0.2766707  0.5843897  0.9784723  1.6377444        Inf
> qt(p,5)
 [1]       -Inf -1.4758840 -0.9195438 -0.5594296 -0.2671809  0.0000000
 [7]  0.2671809  0.5594296  0.9195438  1.4758840        Inf
> qt(p,10)
 [1]       -Inf -1.3721836 -0.8790578 -0.5415280 -0.2601848  0.0000000
 [7]  0.2601848  0.5415280  0.8790578  1.3721836        Inf
> qt(p,Inf)
 [1]       -Inf -1.2815516 -0.8416212 -0.5244005 -0.2533471  0.0000000
 [7]  0.2533471  0.5244005  0.8416212  1.2815516        Inf
```

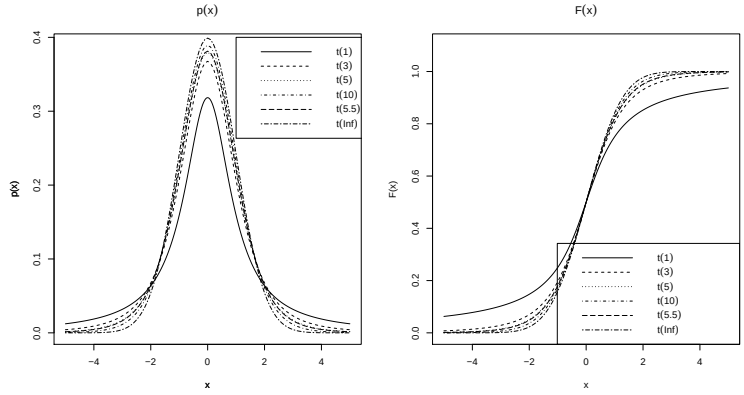

Рис. 1.12: Распределение Стьюдента — плотности и ФР

1.1.18 Равномерное распределение

Равномерное распределение

$$p(x) = \begin{cases} 0, & x < a \text{ или } x > b, \\ \dfrac{1}{b-a}, & x \in [a, b]. \end{cases}$$

в **R** представлено функциями

```
dunif(x, min=0, max=1, log = FALSE)
punif(q, min=0, max=1, lower.tail = TRUE, log.p = FALSE)
qunif(p, min=0, max=1, lower.tail = TRUE, log.p = FALSE)
runif(n, min=0, max=1)
```

Аргументы **min** и **max** соответствуют параметрам **a** и **b** распределения. Значения по умолчанию **min=0** и **max=1**.

Пример 15. *Построим случайную выборку u, подчиняющуюся равномерному на отрезке* $[0;1]$ *распределению.*

```
u = runif(20)
```

и проверим два утверждения: $p(u) = 1$ *и* $F(u) = u$.

```
> punif(u) == u
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[16] TRUE TRUE TRUE TRUE TRUE
> dunif(u) == 1
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[16] TRUE TRUE TRUE TRUE TRUE
```

1.1.19 Распределение Вейбулла

Распределение Вейбулла относится к двупараметрическим распределениям, используется в демографических исследованиях, анализе дожития (исследование смертности). Частным случаем распределения Вейбулла является экспоненциальное распределение.

Плотность распределения:

$$p(x) = \begin{cases} 0, & x < 0, \\ \alpha\lambda x^{\alpha-1} e^{-\lambda x^{\alpha}}, & x \geqslant 0. \end{cases}$$

Математическое ожидание и дисперсия:

$$M\xi = \Gamma\left(1 + \alpha^{-1}\right)\mu, \quad D\xi = \mu^2 \left(\Gamma\left(1 + \frac{2}{\alpha}\right) - \left(\Gamma\left(1 + \frac{1}{\alpha}\right)\right)^2\right).$$

R:

```
dweibull(x, shape, scale = 1, log = FALSE)
pweibull(q, shape, scale = 1, lower.tail = TRUE, log.p = FALSE)
qweibull(p, shape, scale = 1, lower.tail = TRUE, log.p = FALSE)
rweibull(n, shape, scale = 1)
```

Аргумент **shape** — параметр формы α, аргумент **scale** — параметр $1/\lambda$. Оба аргумента — положительные конечные числа.

Пример 16. *Построим графики плотности и ФР распределения Вейбулла.*

```
x=seq(0,10,by=0.01);
y1=dlogis(x,0.5,0.5)
z1=plogis(x,0.5,0.5)

op=par(mfrow=c(1,2))
plot(x,y1,type="l",xlab='x',ylab='p(x)',ylim=c(0,1),lwd=2)
curve(dweibull(x,0.5,1),add=T,lty=2)
curve(dweibull(x,0.5,2),add=T,lty=3)
curve(dweibull(x,1,1),add=T,lty=4)
curve(dweibull(x,1,2),add=T,lty=5)
curve(dweibull(x,2,1),add=T,lty=6)

legend(2,0.6,c(expression(W(0.5,0.5)),expression(W(0.5,1)),
expression(W(0.5,2)), expression(W(1,1)),expression(W(1,2)),
expression(W(2,1))),lty=1:6,cex=0.7)
title(main=expression(p(x)),xlab='x',ylab='p(x)')

plot(x,z1,type="l",xlab='x',ylab='F(x)',ylim=c(0,1.1),lty=1,lwd=2)
curve(pweibull(x,0.5,1),add=T,lty=2)
curve(pweibull(x,0.5,2),add=T,lty=3)
curve(pweibull(x,1,1),add=T,lty=4)
curve(pweibull(x,1,2),add=T,lty=5)
curve(pweibull(x,2,1),add=T,lty=6)
legend(3,0.3,c(expression(W(0.5,0.5)),expression(W(0.5,1)),
expression(W(0.5,2)), expression(W(1,1)),expression(W(1,2)),
expression(W(2,1))),lty=1:6,cex=0.7)
title(main=expression(F(x)))
par(op)
```

Квантили.

```
> p=seq(0,1,by=0.1)
> qweibull(p,0.5,0.5)
 [1] 0.000000000 0.005550419 0.024896522 0.063608508 0.130471409 0.240226507
 [7] 0.419794353 0.724775257 1.295145197 2.650949055         Inf
> qweibull(p,0.5,1)
 [1] 0.00000000 0.01110084 0.04979304 0.12721702 0.26094282 0.48045301
```

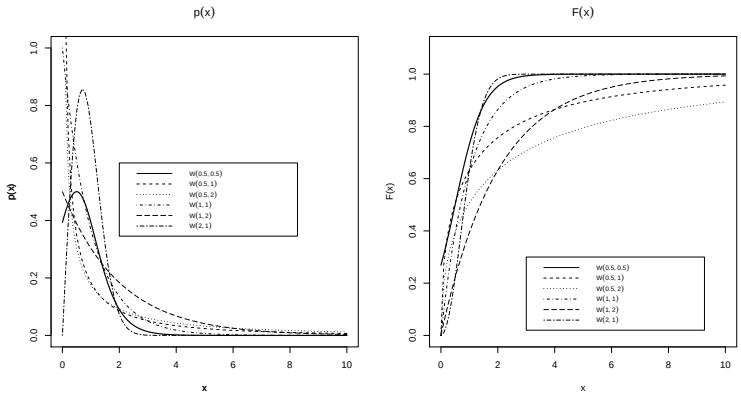

Рис. 1.13: Распределения Вейбулла

```
 [7] 0.83958871 1.44955051 2.59029039 5.30189811        Inf
> qweibull(p,0.5,2)
 [1]  0.00000000  0.02220168  0.09958609  0.25443403  0.52188564  0.96090603
 [7]  1.67917741  2.89910103  5.18058079 10.60379622         Inf
> qweibull(p,1,1)
 [1] 0.0000000 0.1053605 0.2231436 0.3566749 0.5108256 0.6931472 0.9162907
 [8] 1.2039728 1.6094379 2.3025851       Inf
> qweibull(p,1,2)
 [1] 0.0000000 0.2107210 0.4462871 0.7133499 1.0216512 1.3862944 1.8325815
 [8] 2.4079456 3.2188758 4.6051702       Inf
> qweibull(p,2,1)
 [1] 0.0000000 0.3245928 0.4723807 0.5972227 0.7147207 0.8325546 0.9572308
 [8] 1.0972569 1.2686362 1.5174271       Inf
```

1.2 Теория вероятностей в R. Дополнительные одномерные распределения

Здесь вкратце приведём перечень реализованных в **R** вероятностных распределений и укажем названия пакетов, в которых они реализованы.

Дискретные распределения:

- Распределение Дирака (Dirac) реализовано в пакете **distr**. Truncated versions of the binomial and Poisson distributions as well as zero-inflated versions of the binomial, Poisson and negative binomial distributions are implemented in **VGAM**. Zero-inflated Poisson distribution is available in **gamlss.dist**.

- Conway-Maxwell-Poisson distribution: This can be found in **compoisson**.
- Delaporte distribution : This can be found in **gamlss.dist**.
- Hypergeometric distributions : Extented hypergeometric distribution can be found in **BiasedUrn** package, which provides not only p, d, q, r functions but also mean, variance, mode functions. Generalized hypergeometric distribution is implemented in **SuppDists**.
- Logarithmic distribution : This can be found in **VGAM** and **gamlss.dist**. A fast random generator is available for the logarithmic distribution is implemented in **Runuran** as well as the 'density' function.
- Sichel distribution : This can be found in **gamlss.dist**.
- Triangle distribution : The discrete triangle distribution can be found in **TRIANG**.
- Zero inflated/modified (ZI/ZM) distributions : ZM (or Hurdle) Poisson, ZM logarithmic, ZI/ZM negative binomial, ZI Poisson inverse Gaussian, ZI/ZM binomial, ZI/ZM beta binomial distributions are implemented in **gamlss.dist**.
- Zipf law : Package **zipfR** provides tools for the Zipf and the Zipf-Mandelbrot distributions. **VGAM** also implements the Zipf distribution.
- Further distributions : The **VGAM** package provides several additional distributions, namely: Skellam, Yule-Simon, Zeta and Haight's Zeta, Borel-Tanner and Felix distribution.

Непрерывные распределения:

- Arcsine distribution : implemented in package **distr**.
- Beta distribution and its extensions : Base R provides the d, p, q, r functions for this distribution (see above). **actuar** provides moments and limited expected values. It also provide the d, p, q, r functions for the generalized beta and the inverse tranformed beta distribution. The zero and one inflated beta distribution can be found in **gamlss.dist** as well as generalized beta of the first and second kind. The beta of the second kind and the generalized beta distribution can also be found in **VGAM**. Several special cases of the generalized beta distribution are also implemented in **VGAM**: Lomax, inverse Lomax, Dagum, Fisk (aka log logistic), (inverse or not) paralogistic and Singh-Maddala distribution.

- Benini distribution : Provided in **VGAM**.
- Birnbaum-Saunders distribution : Provided in packages **bs** and **VGAM**. The generalized Birnbaum-Saunders distribution is implemented in **gbs**.
- Box Cox distributions : **gamlss.dist** provides the Box-Cox normal, the Box-Cox power exponential and the Box-Cox t distributions.
- Cardioid distribution : Provided in **VGAM**.
- Cauchy distribution : Base R provides the d, p, q, r functions for this distribution (see above). Another implementation is available in **lmomco**.
- Chi(-squared or not) distributions : Base R provides the d, p, q, r functions for this distribution (see above). Moments, limited expected values and the производящая функция моментов[1] (moment generating function) are provided by **actuar**. Only d,r functions are available for the inverse chi-squared distribution by package **geoR**. A fast random generator is available for the Chi distribution is implemented in **Runuran** as well as the density function. The non-central Chi distribution is not yet implemented.
- Davies distribution : The Davies distribution is provided in **Davies** package.
- Dirichlet distribution : functions d, r are provided in **MCMCpack** and **bayesm**.
- Exponential distribution and its extensions : Base **R** provides the d, p, q, r functions for this distribution (see above). **actuar** provides additional functions such as the moment generating function, moments and limited expected values. It also has the d, p, q, r for the inverse exponential distribution. The shifted exponential distribution is implemented in **lmomco** package with d, p, q, r functions. d, p, q, r functions for the power and the skew power exponential type 1-4 distributions are implemented in **gamlss.dist**.
- Frechet distribution : Provided in **VGAM** and **evd**. A fast random generator is available for the Frechet distribution is implemented in **Runuran** as well as the density function.

[1]Производящая функция моментов — способ задания вероятностных распределений. Используется чаще всего для вычисления моментов.

Определение. Пусть задана с.в. ξ. Её производяшей функцией моментов называется функция $M_\xi(t) = Me^{t\xi}$.

Производящая функция моментов однозначно определяет распределение.

- Friedman's Chi distribution : Provided in **SuppDists**.

- Gamma distribution and its extensions : Base **R** provides the d, p, q, r functions for this distribution (see above). **actuar** provides d, p, q, r functions for the inverse, the inverse transformed and the log gamma distributions while **ghyp** provides those functions for the variance gamma distribution. **VarianceGamma** provides d, p, q, r functions for the variance gamma distribution as well as moments (skewness, kurtosis, ...). **VGAM** provides d, p, q, r functions of the log gamma and the generalized gamma distribution. The generalized gamma distribution can also be found in **gamlss.dist**.

- Gaussian (or normal) distribution and its extensions : Base **R** provides the d, p, q, r functions for this distribution (see above). The **truncnorm** package provides d, p, q, r functions for the усечённое одномерное gaussian distribution as well as functions for the first two moments. **actuar** provides the moment generating function and moments. d, p, q, r functions for the generalized inverse gaussian distribution can be found in **gamlss.dist** and **HyperbolicDist**. A fast random generator is available for the (generalized) Inverse Gaussian distribution is implemented in **Runuran** as well as the density function. **SuppDists** provides functions for the inverse Gaussian distribution as well and furthermore includes functions for computing moments, skewness, kurtosis. **VGAM** and **fBasics** also implement the свёрнутое и смещённое (несимметрическое) normal distribution, the inverse gaussian distribution. **lmomco** implements the generalized normal distribution. The log normal distribution is implemented in Base R (see above). Ex-Gaussian distribution is implemented in **gamlss.dist**. Finally, the multivariate Gaussian distribution is provided by the packages **mvtnorm** and **mnormt**, while **mvtnormpcs** implements multivariate student/normal integrals, given a correlation matrix structure. **tmvtnorm** implements the усечённое multivariate normal distribution.

- General Error Distribution (also known as exponential power distribution) : provided in **normalp** and **fExtremes**, see exponential item.

- Generalized Extreme Value distribution : Provided in **lmomco** (d, p, q) , **VGAM**, **evd**, **evir** and **fExtremes** (d, p, q, r). Both bivariate and multivariate Extreme Value distributions as well as order/maxima/minima distributions are implemented in **evd** (d, p, r). **evdbayes** provides some additional functions for GEV distribution using MCMC.

- Gumbel distribution : Provided in packages **lmomco**, **VGAM**, **gamlss.dist** and **evd**. A fast random generator is available for the Gumbel distribution

is implemented in **Runuran** as well as the density function. The reverse Gumbel distribution is implemented in **lmomco** and **gamlss.dist**.

- Hyperbolic distribuion : **fBasics**, **ghyp** and **HyperbolicDist** packages provide d, p, q, r functions for the generalized hyperbolic distributions. A fast random generator is available for the hyperbolic distribution is implemented in **Runuran** as well as the density function.
- Johnson distribution : Provided in **SuppDists**.
- Kendall's tau distribution : Provided in **SuppDists**.
- Kruskal Wallis distribution : Provided in **SuppDists**.
- Kappa distribution : Provided in **lmomco**.
- Kumaraswamy distribution : Provided in package **VGAM**.
- (Tukey) Lambda distribution and its extentions : The generalized Lambda distribution (GLD) is well known for its wide range of shapes. There exists 3 kinds of GLD in the literature: RS, FMKL and FM5. The following packages implement such distributions (with d, p, q, r functions): **gld** for the 3 kinds of GLD (RS, FMKL and FM5), **GLDEX** for 2 kinds (RS, FMKL) and **Davies** for 1 type (RS). The original Tukey Lambda distribution can be obtained as a special case of the generalized Lambda distribution.
- Laplace and asymetric Laplace distribution : Provided in **VGAM** and **HyperbolicDist** packages. Laplace distribution (also called double exponential distribution) is implemented in **distr**. A fast random generator is available for the Laplace distribution is implemented in **Runuran** as well as the density function.
- Logistic distributions and its extensions : Base R provides the d, p, q, r functions for this distribution (see above). **actuar** provides d, p, q, r functions for the log logistic (also called Fisk), the paralogistic and the inverse paralogistic distributions. **VGAM** package also implements these distributions plus the bivariate logistic distribution. **lmomco** implements the generalized logistic distribution.
- Maxwell distribution : Provided in **VGAM**.
- Nakagami distribution : Provided in **VGAM**.

- Pareto distribution : d, p, q, r functions are implemented in **VGAM** for the Pareto distribution type IV (which includes Burr's distribution, Pareto type III, Pareto type II (also called the lomax distribution) and Pareto type I) and the (upper/lower) усечённое Pareto distribution. In an actuarial context, **actuar** provides d, p, q, r functions as well as moments and limited expected values for the Pareto I and II, the inverse Pareto, the 'generalized pareto' distributions, the Burr and the inverse Burr distributions. A fast random generator for the Burr and the Pareto II distribution is implemented in **Runuran** as well as the density. Finally **lmomco**, **POT**, **evd**, **fExtremes** and **evir** packages implement the Generalized Pareto Distribution (from Extreme Value Theory), which is depending the shape parameter's value a Pareto II distribution, a shifted exponential distribution or a generalized beta I distribution.

- Pearson type III distribution : Available in **lmomco**.

- Pearson's Rho distribution : Provided in **SuppDists**.

- Planck's distribution : a random generator is available in **Runuran**.

- Phase-type distributions : Provided in **actuar**.

- Rayleigh distribution : Provided in packages **VGAM** and **lmomco**.

- Rice distribution : Provided in **VGAM**.

- Sinh-Arcsinh distribution : Provided in **gamlss.dist**.

- Slash distribution : Provided in **VGAM**.

- Spearman's Rho distribution : Provided in **SuppDists**.

- устойчивые распределения (stable distributions) : d, p, q, r functions are available in **fBasics**, the functions use the approach of J.P. Nolan for general stable distributions.

- Student distribution and its extensions : Base **R** provides the d, p, q, r functions for Student and non central Student distribution (see above). The несимметричное (смещённое) Student distribution is provided by **skewt**, **sn** and **gamlss.dist** packages. d, p, q, r functions for the generalized t-distribution can be found in **gamlss.dist**. **fBasics** provides d, p, q, r functions for the skew and the generalized hyperbolic t-distribution. The multivariate Student distribution is provided by the packages **mvtnorm** and **mnormt**.

- Triangle distribution : Provided in packages **triangle** and **VGAM**. A fast random generator is available for the triangle distribution is implemented in **Runuran** as well as the density function.

- Tweedie distribution : the Tweedie distribution is implemented in package **tweedie**. Let us note that the Tweedie distribution is not necessarily continuous, a special case of it is the Poisson distribution.

- Wakeby distribution : Provided in **lmomco**.

- Weibull distribution and its extensions : Base **R** provides the d, p, q, r functions for this distribution (see above). The inverse Weibull is provided by **actuar** package and also the moments and the limited expevted value for both the raw and the inverse Weibull distribution. Finally, **lmomco** implements the Weibull distribution while **evd** implements the reverse Weibull distribution. d, p, q, r functions for the reverse generalized extreme value distribution are provided in **gamlss.dist**.

- Wishart and inverse Wishart distributions : functions d, r are provided in **MCMCpack** and **bayesm**.

Смеси вероятностных законов:

- Cauchy-polynomial quantile mixture : d, p, q, r functions are provided by **Lmoments**.

- Gaussian mixture : Functions d, r are provided by **mixtools** package when dealing with finite mixture models. **nor1mix** provides d, p, r functions for Gaussian mixture.

- Gamma mixture : Gamma shape mixtures are implemented (d, p, r) in the **GSM** package.

- Student mixture : The **AdMit** package provides d, r functions for Student mixtures in the context of Adaptive Mixture of Student-t distributions.

1.3 Теория вероятностей в R. Многомерные распределения

1.4 Пакет prob в R. Элементарная вероятность на конечных пространствах элементарных исходов

This package provides a framework for performing elementary probability calculations on finite sample spaces, which may represented by data frames or lists. Functionality includes setting up sample spaces, counting tools, defining probability spaces, performing set algebra, calculating probability and conditional probability, tools for simulation and checking the law of large numbers, adding random variables, and finding marginal distributions.

1.5 Генераторы случайных чисел R

Генераторы случайных чисел (Random Number Generators):

- Basic functionality : R provides several random number generators (RNGs). The random seed can be provided via set.seed and the kind of RNG can be specified using RNGkind. The default RNG is the Mersenne-Twister algorithm. Other generators include Wichmann-Hill, Marsaglia-Multicarry, Super-Duper, Knuth-TAOCP, Knuth-TAOCP-2002, as well as user-supplied RNGs. For normal random numbers, the following algorithms are available: Kinderman-Ramage, Ahrens-Dieter, Box-Muller, Inversion (default). In addition to the tools above, setRNG provides an easy way to set, retain information about the setting, and reset the RNG.

- Pseudo-randomness : RDieHarder offers several dozen new RNGs from the GNU GSL. randtoolbox provides more recent RNGs such as SF Mersenne-Twister and WELL, which are generators of Mersenne Twister type, but with improved quality parameters. rngwell19937 provides one of the WELL generators with 53 bit resolution of the output and allows seeding by a vector of integers of arbitrary length. randaes provides the deterministic part of the Fortuna cryptographic pseudorandom number generator (AES). SuppDists implements two RNGs of G. Marsaglia.

  - Support for several independent streams: rsprng implements Scalable Parallel RNGs library. rstream focuses on multiple independent streams of random numbers from different sources (in an object oriented approach).

  - For non-uniform generation, the Runuran package interfaces to the UNU.RAN library for universal non-uniform generation.

- Quasi-randomness : The randtoolbox provides the following quasi random sequences: the Sobol sequence, the Halton (hence Van Der Corput) sequence and the Torus sequence (also known as Kronecker sequence). lhs package implements the latin hypercube sampling, an hybrid quasi/pseudo random method.

- True randomness : The random package provides several functions that access the true random number service at random.org .

- RNG tests : RDieHarder offers numerous tests of RNGs based on a reimplementation and extension of Marsaglia's DieHarder battery. randtoolbox provides basic RNG tests.

- Parallel computing : For parallel computing with random numbers, see the HighPerformanceComputing task view.

1.6 Дополнительные материалы по теории вероятностей в R

Дополнительные материалы по теории вероятностей в **R**

- Benchmark : A set of 28 densities suitable for comparing nonparametric density estimators in simulation studies can be found in the benchden package. The densities vary greatly in degree of smoothness, number of modes and other properties. The package provides d,p,q and r functions.

- Empirical distribution : Base R provides functions for univariate analysis: (1) the empirical density (see density), (2) the empirical cumulative distribution function (see ecdf), (3) the empirical quantile (see quantile) and (4) random sampling (see sample). For multivariate analysis, the package mecdf provides the multivariate empirical distribution function.

- Hierarchical models : Distributions whose some parameters are no longer constant but random according to a particular distribution. VGAM provides a lot of hierarchical models: beta/binomial, beta/geometric and beta/normal distributions. bayesm implements: binary logit, linear, multivariate logit and negative binomial models. Furthermore LearnBayes and MCMCpack provides poisson/gamma, beta/binomial, normal/normal and multinomial/Dirichlet models.

- Object-orientation : General discrete and continuous distributions are implemented in package distr respectively via S4-class DiscreteDistribution and AbscontDistribution providing the classic d, p, q and r functions. distrEx extends available distributions to multivariate and conditional distributions as well as methods to compute useful statistics (expectation, variance,...) and distances between distributions (Hellinger, Kolmogorov,... distance). Finally package distrMod provides functions for the computation of minimum criterion estimators (maximum likelihood and minimum distance estimators). See other packages of the distr-family (distrSim, distrTEst, distrTeach, distrDoc).

- TI 83 scientific calculator : distributions provides the base R probability distributions (binomial, poisson, geometric, normal, chi square, Fisher, Student) based on TI-83 Plus graphic scientific calculator.

- Transformation : Lebesgue decomposition are implemented in distr, as well as Convolution, Truncation and Huberization of distributions. Furthermore, distr provides distribution of the maximum or minimum of two distributions. See Object-orientation below.

- Transversal functions : Package modeest provides mode estimation for various distributions, while lmomco and Lmoments focuse on (L-)moments estimation. VGAM provides a lot of parameter estimation for usual and "exotic"distributions. Package MASS implements the flexible fitdistr function for parameter estimations. fitdistrplus greatly enlarges and enhances the tools to fit probability distributions.

Глава 2

Основы математической статистики в R

В этой главе будут рассмотрены функции **R**, позволяющие производить базовые статистические расчёты.

2.1 Анализ категориальных данных

2.1.1 Обработка данных — выделение категорий. Построение статистического ряда

Некоторые функции, позволяющие обрабатывать категориальные данные, были рассмотрены ранее в части первой данного пособия «Введение в статистический пакет **R**: типы переменных, структуры данных, чтение и запись информации, графика». Это были следующие функции: **factor()**, **ordered()** и **table()**, позволяющие определить различные категории в имеющемся массиве данных, а также сколько элементов выборки относится к данным категориям.

Графические функции, с помощью которых можно анализировать категориальные данные, это **barplot()** (построение мозаичных диаграмм) и **pie()** (построение круговых диаграмм).

Если нужно перевести числовые данные в категориальные, то следует воспользоваться функцией **cut()**:

```
cut(x, breaks, labels = NULL,
include.lowest = FALSE, right = TRUE, dig.lab = 3,
ordered_result = FALSE, ...)
```

где **x** — числовой вектор, преобразуемый в факторы; **breaks** — либо число ($\geqslant 2$), задающее число интервалов разбиения, либо числовой вектор, задаю-

щий границы (точки) разбиения; **labels** — символьный вектор — вектор названий создаваемых категорий (по умолчанию в качестве названий категорий берутся сами интервалы разбиения «$(a, b]$»); **include.lowest** — логический аргумент — нужно ли элемент x_i вектора **x**, совпадающий с нижней (верхней) границей одного из интервалов разбиения, включать в этот интервал; **right** — логический аргумент — задаёт вид интервала разбиения: $(a, b]$ (по умолчанию) или $[a, b)$; **dig.lab** — числовой аргумент — число знаков после запятой в названии категорий (если названия категорий — интервалы разбиения); **ordered_result** — логический аргумент — нужно ли упорядочить созданные категории.

Примечание. Если **breaks** — число, то создаются интервалы разбиения одинаковой длины, причём нижняя граница первого интервала и верхняя граница последнего сдвигаются в сторону уменьшения (увеличения) на 0.1%, чтобы наименьшее и наибольшее значения вектора **x** попали в интервалы разбиения.

Примечание. Если **breaks** — числовой вектор (точки разбиения), то длина вектора **labels** названий категорий разбиения должна быть на единицу меньше длины вектора **breaks**.

Рассмотрим пример, в котором используются числовые и символьные категориальные данные.

Пример 17. *Рассмотрим динамику изменения оценок по теории вероятностей и математической статистике за последние два года: оценки по стобалльной, по пятибалльной и по европейской (буквенной) системам.*

Имеются два массива данных: **год1** *и* **год2**, *длиной 52 и 61 соответственно.*

```
> год1
 [1] 83 70 86 51 61 67 80 84 70 64 83 55 88 75 61 70 95 52 75 92 86 89 83 58 51
[26] 65 87 54 90 62 67 66 83 65 70 87 69 51 51 60 52 74 80 79 85 85 92 22 92 57
[51] 84 92
> год2
 [1] 70 31 70 46 78 69 36 33 65 70 74 51 90 26 62 70 86 86 80 33 55 31 69 31 62
[26] 63 86 55 69 62 31 90 31 31 69 64 57 75 53 89  0 31 81 53 86 52 98 58 31 69
[51] 69 51 75 39  0 56 51 69 46 41 65
```

Выделим следующие категории:

- *по пятибалльной системе:* **2** *— баллы от 0 до 50,* **3** *— от 51 до 68,* **4** *— от 69 до 85 и* **5** *— от 86 до 100;*
- *по европейской системе:* **F** *— от 0 до 30,* **FX** *— от 31 до 50,* **E** *— от 51 до 60,* **D** *— от 61 до 68,* **C** *— от 69 до 85 ,* **B** *— от 86 до 95,* **A** *— от 96 до 100.*

Для этого воспользуемся функцией **cut()** *для первого вектора данных :*

```
> разбиение1_1=cut(год1,breaks=c(-1,50,68,85,100),labels=c('2','3','4','5'))
> разбиение1_1
 [1] 4 4 5 3 3 3 4 4 4 3 4 3 5 4 3 4 5 3 4 5 5 5 4 3 3 3 5 3 5 3 3 3 4 3 4 5 4 3
[39] 3 3 3 4 4 4 4 4 5 2 5 3 4 5
Levels: 2 3 4 5
> разбиение1_2=cut(год1,breaks=c(-1,30,50,60,68,85,95,100),
+ labels=c('F','FX','E','D','C','B','A'))
> разбиение1_2
 [1] C C B E D D C C C D C E B C D C B E C B B B C E E D B E B D D D C D C B C E
[39] E E E C C C C C B F B E C B
Levels: F FX E D C B A
> разбиение2_1=cut(год2,breaks=c(-1,50,68,85,100),labels=c('2','3','4','5'))
> разбиение2_1
 [1] 4 2 4 2 4 4 2 2 3 4 4 3 5 2 3 4 5 5 4 2 3 2 4 2 3 3 5 3 4 3 2 5 2 2 4 3 3 4
[39] 3 5 2 2 4 3 5 3 5 3 2 4 4 3 4 2 2 3 3 4 2 2 3
Levels: 2 3 4 5
> разбиение2_2=cut(год2,breaks=c(-1,30,50,60,68,85,95,100),
+ labels=c('F','FX','E','D','C','B','A'))
> разбиение2_2
 [1] C  FX C  FX C  C  FX FX D  C  C  E  B  F  D  C  B  B  C  FX E  FX C  FX D
[26] D  B  E  C  D  FX B  FX FX C  D  E  C  E  B  F  FX C  E  B  E  A  E  FX C
[51] C  E  C  FX F  E  E  C  FX FX D
Levels: F FX E D C B A
```

Таким образом числовые векторы были преобразованы в факторы.

Определим число элементов, попавших к построенные категории. Сначала для пятибалльной системы оценок:

```
> таблица1_1=table(разбиение1_1);таблица1_1
разбиение1_1
 2  3  4  5
 1 20 19 12
> таблица2_1=table(разбиение2_1);таблица2_1
разбиение2_1
 2  3  4  5
18 18 17  8
```

а теперь и для европейской системы:

```
 F FX  E  D  C  B  A
 1  0 11  9 19 12  0
```

```
> таблица2_2=table(разбиение2_2);таблица2_2
разбиение2_2
 F FX  E  D  C  B  A
 3 15 11  7 17  7  1
```

Как видно из полученных результатов, увеличилось количество неудовлетворительных оценок при сокращении положительных оценок.

Таблицы относительных частот определяются следующим образом:

```
> table(разбиение1_1)/length(разбиение1_1)

разбиение1_1
         2          3          4          5
0.01923077 0.38461538 0.36538462 0.23076923
> table(разбиение2_1)/length(разбиение2_1)

разбиение2_1
        2         3         4         5
0.2950820 0.2950820 0.2786885 0.1311475
> table(разбиение1_2)/length(разбиение1_2)

разбиение1_2
         F          FX          E          D          C          B          A
0.01923077 0.00000000 0.21153846 0.17307692 0.36538462 0.23076923 0.00000000
> table(разбиение2_2)/length(разбиение2_2)

разбиение2_2
         F          FX          E          D          C          B          A
0.04918033 0.24590164 0.18032787 0.11475410 0.27868852 0.11475410 0.01639344
```

В разобранном выше примере данные (дискретная выборка) были преобразованы в категориальные по нескольким группам (системы оценок), далее были построены различные таблицы, первая строка которых представляла из себя категории, а вторая — либо число элементов выборки, либо относительная доля элементов выборки, принадлежащих каждой категории. Т.е. были построены статистические ряды и таблицы относительных частот.

В двух следующих разделах будут рассмотрены графические функции, предназначенные для анализа категориальных данных.

2.1.2 Мозаичные диаграммы — функция barplot()

Для создания мозаичных диаграмм используется функция **barplot()**

```
barplot(height, width = 1, space = NULL,
names.arg = NULL, legend.text = NULL, beside = FALSE,
horiz = FALSE, density = NULL, angle = 45,
col = NULL, border = par("fg"),
main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
xlim = NULL, ylim = NULL, xpd = TRUE, log = "",
axes = TRUE, axisnames = TRUE,
cex.axis = par("cex.axis"), cex.names = par("cex.axis"),
inside = TRUE, plot = TRUE, axis.lty = 0, offset = 0,
add = FALSE, args.legend = NULL, ...)
```

Её аргументы:

- **height** — единственный обязательный аргумент — числовой массив (вектор или матрица), определяющий высоту элементов строющейся диаграммы.

- **width** — дополнительный аргумент, определяющий ширину столбца диаграммы. Либо числовой вектор, длина которого приводится к числу столбцов диаграммы, либо число, задающее ширину всех столбцов диаграммы.

- **space** — числовой аргумент (скаляр или вектор), определяющий пространство (доля средней ширины столбца) перед каждым столбцом диаграммы;

- **names.arg** — символьный аргумент — названия столбцов диаграммы или названия поддиаграмм; если аргумент не задан, то берутся названия элементов вектора **height** или названия столбцов матрицы **height**.

- **legend.text** — символьный или логический аргумент, выводящий легенду диаграммы. Применяется только в случае, если **height** — матрица. Длина символьного вектора **legend.text** должна совпадать с количеством строк матрицы **height**. Если **legend.text** — логический аргумент и его значение **TRUE**, то используются имена строк матриц.

- **beside** — логический аргумент — используется, если **height** — матрица. Определяет, должны ли столбцы поддиаграмм располагаться друг над другом (**beside** = **FALSE**) или сбоку друг от друга (**beside** = **TRUE**) (см. рис. 2.3 и рис. 2.4).

- **horiz** — логический аргумент. Если значение **FALSE**, то столбцы диаграммы строятся вертикально и первый столбец находится слева. Если

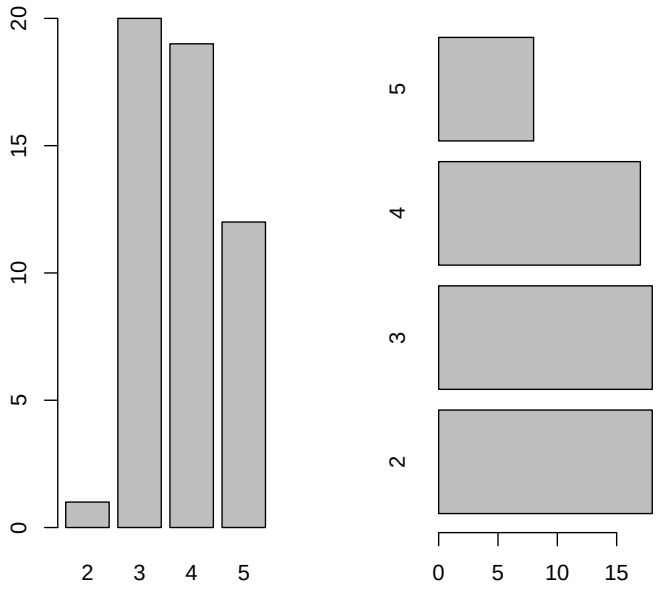

Рис. 2.1: Мозаичная диаграмма: **horiz** = **FALSE** и **horiz** = **TRUE**

horiz = **TRUE**, то столбцы располагаются горизонтально, первый столбец находится внизу (см. рис. 2.1).

- **density** и **angle** — два числовых параметра, отвечающих за штриховку столбоцв диаграммы. Аргумент **density** задаёт число штриховочных линий на один дюйм, если **density** — вектор, то определяется либо штриховка каждого столбца диаграммы, либо каждой поддиаграммы. Данный аргумент принимает только положительные значения. **angle** — числовой аргумент, определяющий угол наклона штриховки (против часовой стрелки). Если **angle** — вектор, то тем самым задаётся штриховка каждого столбца или поддиаграммы (см.‘2.5). Задание **density** = **NULL** подавляет штриховку (по умолчанию)

- **col** — символьный или числовой вектор, определяющий цвет каждого столбца или поддиаграммы. По умолчанию, если **height** – вектор, то все столбцы закрашиваются в серый цвет, если же **height** — матрица, то для

каждой поддиаграммы задается свой оттенок серого цвета[1]. Если задана штриховка, то аргумент **col** определяет цвет штриховых линий (см.‘2.5).

- **border** — числовой, символьный или логический аргумент, задающий цвет границ столбцов диаграммы. Если **border = NA** или **border = FALSE**. то границы столбцов не рисуются. Если **border = TRUE**, то цвет границы каждого столбца определяется значением аргумента **col**.
- **main** и **sub** — основной и дополнительный заголовки диаграммы.
- **xlab** и **ylab** — названия осей **X** и **У**.
- **xlim** и **ylim** — верхние и нижние границы осей координат.
- **xpd** — логический аргумент — могут ли столбы диаграммы выходить за область построения диаграммы.
- **log** — символьный аргумент — задаёт тип осей координат (обычные или логарифмические).
- **axes** — логический аргумент — нужно ли строить координатную ось (на этой оси выводятся значения аргумента **height**).
- **axisnames** — логический аргумент. если значение этого аргумента **TRUE** и определён **names.arg**, то строится вторая ось с названиями категорий.
- **cex.axis** и **cex.names** — числовые аргументы — определяют размер числовых делений и подписей на осях соответственно (значения меньше 1 — уменьшение размера, больше — увеличение).
- **inside** — логический аргумент — нужно ли рисовать граничные линии для столбцов, расположенных впритык и сбоку друг от друга. Используется, если только **space = 0**.
- **plot** — логический аргумент — нужно ли выводить в графическом окне саму диаграмму.
- **axis.lty** — числовой или символьный параметр, отвечающий за вид линии координатной оси.

[1]Более подробно о задании цвета в графических функциях изложено в части первой данного пособия «Введение в статистический пакет **R**: типы переменных, структуры данных, чтение и запись информации, графика»

- **offset** — числовой вектор — задаёт смещение столбцов относительно оси **X** ((см.‘2.5)).

- **add** — логический аргумент — нужно ли добавлять строящуюся диаграмму к ранее нарисованной.

- **args.legend** — список дополнительных аргументов, определяющих тип легенды. Используется только, если задана сама легенда — **legend.text**.

- **...** — дополнительные аргументы, отвечающие за оси и заголовки.

Примечания:

- если **height** — вектор, то диаграмма состоит из прямоугольных столбцов, высота которых определяется значениями соответствующих элементов вектора (см. рис.2.2);

- если **height** — числовая матрица и значение логического аргумента **beside = FALSE**, то в рамках одной мозаичной диаграммы строится несколько поддиаграмм (число поддиаграмм определяется числом столбцов матрицы), высота составных частей столбца поддиаграммы определяется значениями элементов столбца матрицы (см. рис.2.3);

- если **height** — числовая матрица и значение логического аргумента **beside = TRUE**, то в рамках одной мозаичной диаграммы строится несколько поддиаграмм (число поддиаграмм определяется числом столбцов матрицы), высота столбцов поддиаграммы определяется значениями элементов столбца матрицы (см. рис.2.4);

- аргумент **width**, являющийся скаляром, не используется, если не задан аргумент **xlim**;

- если **height** — числовая матрица и **beside = TRUE**, то аргумент **space = c(a, b)**, где a — расстояние между столбцами поддиаграммы, а b — расстояние, отделяющее одну поддиаграмму от другой; значения по умолчанию **space = c(0, 1)**.

Пример 18. *Продолжим пример 17 и построим мозаичные диаграммы для полученных нами таблиц.*

Сначала построим диаграммы по созданным таблицам **таблица1_1** — **таблица2_2**, *т.е. когда* **height** — *числовой вектор — рис.‘2.2.*

Затем на основе **таблицы1_1** *и* **таблицы2_1** *создадим* **матрица1** *и построим мозаичную диаграмму для случая, когда* **height** — *числовая матрица и* **beside = FALSE** — *рис.‘2.3*

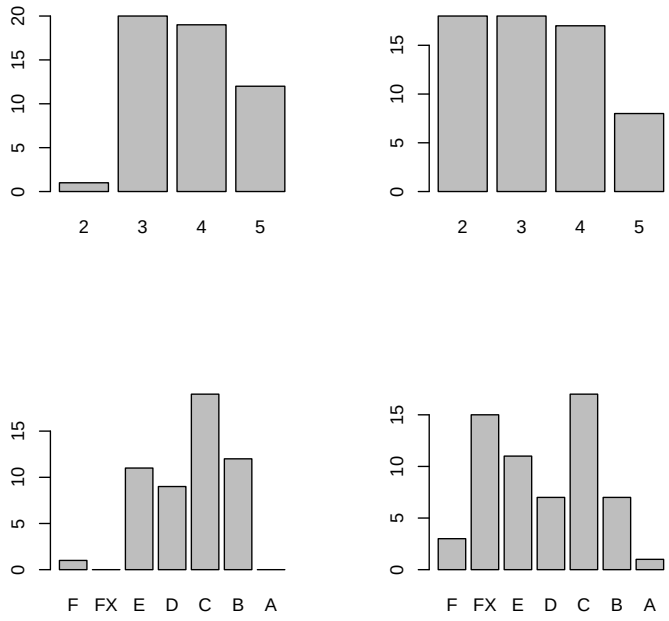

Рис. 2.2: Мозаичная диаграмма. **height** — числовой вектор

```
> матрица1=cbind(таблица1_1,таблица2_1);матрица1
  таблица1_1 таблица2_1
2          1         18
3         20         18
4         19         17
5         12          8
```

Последний вариант: **height** *— числовая матрица и значение логического аргумента* **beside** $=$ **TRUE** *— рис.'2.4.*

```
> матрица2=cbind(таблица1_2,таблица2_2);матрица2
   таблица1_2 таблица2_2
F           1          3
FX          0         15
E          11         11
D           9          7
C          19         17
B          12          7
A           0          1
```

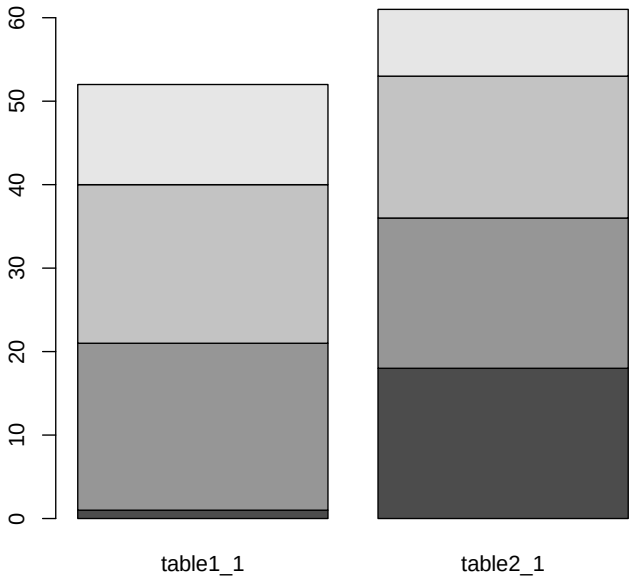

Рис. 2.3: Мозаичная диаграмма. **height** — числовая матрица, **beside** = **FALSE**

Рассмотрим различные варианты задания штриховки, цвета столбцов и цвета границ столбцов, а также смещения столбцов относительно оси **X**.

```
par(mfrow=c(2,2))
barplot(height=table1_1,density= c(3,-3,5,-5),angle=c(20,30,40,50))
barplot(height=table2_1,density= c(3,7,5,9),angle=c(20,30,40,50),
offset=1:4)
barplot(height=table1_2, density=c(4,6),angle=c(50,75),col=c('red','green'))
barplot(height=table2_2,density=c(3,-3),angle=c(-30,120), col="green",
border='purple',offset=c(1,3))
```

Заметим, что мозаичные диаграммы можно рассматривать как аналог гистограммы, о которой речь пойдёт в 2.2.2.

2.1.3 Круговые диаграммы — функция pie()

Прежде чем перейти к круговым диаграммам, ещё раз отметим, что с точки зрения математической статистики и её приложений, диаграммы данного

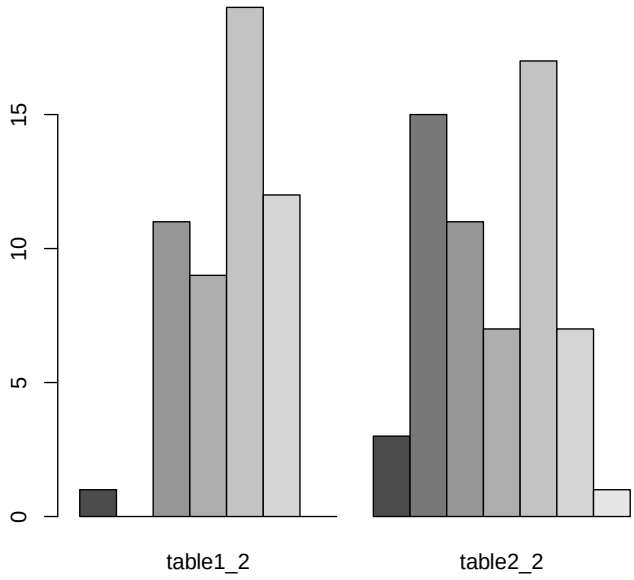

Рис. 2.4: Мозаичная диаграмма. **height** — числовая матрица, **beside** = **TRUE**

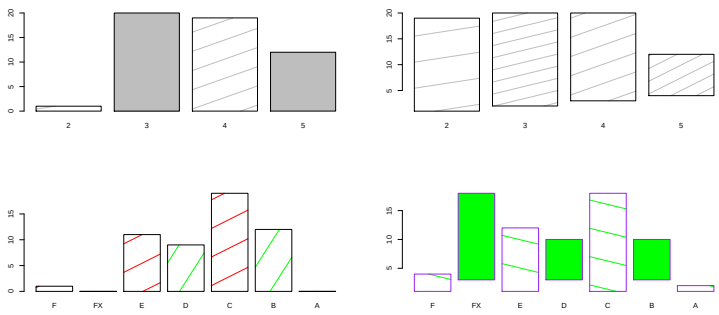

Рис. 2.5: Различные варианты штриховки, цвета, границ и смещения столбцов

вида малоинформативны. Они несут иллюстративный (описательный) характер.

Для создания круговых диаграмм используется функция

```
pie(x, labels = names(x), edges = 200, radius = 0.8,
clockwise = FALSE, init.angle = if(clockwise) 90 else 0,
```

```
density = NULL, angle = 45, col = NULL, border = NULL,
lty = NULL, main = NULL, ...)
```

Её аргументы:

- **x** — неотрицательный числовой вектор, его значения определяю площадь секторов диаграммы.
- **labels** — символьный вектор или выражения тина **expression** — подписи к секторам диаграммы. По умолчанию используются имена вектора **x**.
- **edges** — положительное целое число. Хотя диаграмма и именуется «круговой», на самом деле строится правильный многоугольник, аппроксимирующий круг.
- **radius** — радиус круговой диаграммы. Диаграмма строится внутри квадрата, стороны которого проходят через координаты -1 и 1 по осям **X** и **У**. Таким образом, максимальное значение, которое может принимать радиус диаграммы — это $\sqrt{2}$.
- **clockwise** — логический аргумент — сектора диаграммы выводят по часовой или против часовой стрелки (по умолчанию).
- **init.angle** — числовой аргумент, определяющий угол, начиная с которого выводятся сектора. Если **clockwise** = **FALSE**, то начальный угол по умолчанию равен 0 (3 часа), в противном случае — 90 градусов (12 часов).
- **density** и **angle** — число или числовой вектор — плотность и угол наклона штриховых линий.
- **col** — числовой или символьный вектор — цвет секторов или цвет штриховки секторов.
- **border** и **lty** — аргументы (вектора), отвечающие за вид каждого сектора (многоугольника) диаграммы.
- **main** — основной заголовок диаграммы.
- **...** — дополнительные аргументы, отвечающие только за заголовок и подписи к секторам.

Рассмотрим на примере эту функцию.

Пример 19. *Построим круговые диаграммы для таблиц, полученных в примере 17.*

```
par(mfrow=c(2,2))
pie(table1_1,labels=c('2','3','4','5'),edges=40,radius=1,clockwise=T,
init.angle=60,density=c(2,4,6,8),angle=c(15,45,60,75),col=1:4,
border=NULL,main='table1_1')
pie(table2_1,labels=c('2','3','4','5'),edges=60,radius=0.8,clockwise=F,
init.angle=45,col=3:6,border='purple',main='table2_1')
pie(table1_2,labels=c('F','FX','E','D','C','B','A'),edges=80,radius=1,
clockwise=T,init.angle=30,density=c(2,4,6,8,10,12,14),
angle=c(15,45,60,75,90,105,120),col=1:7,border=NULL,main='table1_2')
pie(table2_2,labels=c('F','FX','E','D','C','B','A'),edges=100,radius=1,
clockwise=F,init.angle=30,col=1:7,border=NULL,main='table2_2')
```

Результат представлен на следующем рисунке:

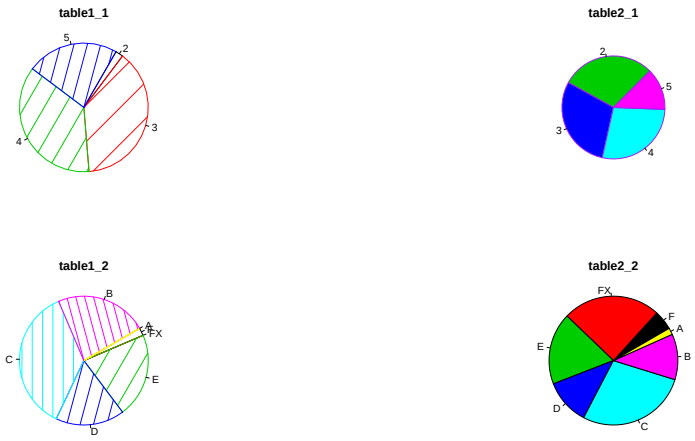

Рис. 2.6: Круговые диаграммы

2.2 Статистические характеристики

В этом разделе будут рассмотрены базовые числовые характеристики математической статистики, например выборочное среднее и дисперсия, выборочная ковариация и коэффициент корреляции, выборочные квантили, медиана и пр.

Также будут рассмотрены графические функции **hist()** и **boxplot()**.

2.2.1 Простейшие числовые характеристики

Функции mean(), weighted.mean() и sd()

```
mean(x, trim = 0, na.rm = FALSE, ...)
weighted.mean(x, w, ..., na.rm = FALSE)
sd(x, na.rm = FALSE)
```

Функция **mean()** позволяет вычислять выборочное среднее $\bar{m} = \frac{1}{n}\sum\limits_{i=1}^{n} x_i$ по заданной выборке **x**, которая является единственным обязательным аргументом данной функции. Дополнительный аргумент **trim** определяет долю элементов выборки, которая должна быть отсечена от начала и конца перед вычислением выборочного среднего, возможные значения этого аргумента — числа от 0 до 0.5. Аргумент **na.rm** позволяет исключить из рассмотрения **NA**.

Функция **weighted.mean** вычисляет по выборке **x** взвешенное среднее $\tilde{m} = \sum\limits_{i=1}^{n} x_i w_i$ согласно заданному вектору весов **w** (стоит отметить, что элементы вектора весов нормируются автоматически). Длины векторов **x** и **w** должны совпадать. Если аргумент **w** не задан, то всем элементам выборки приписываются одинаковые веса и, таким образом, получаем выборочное среднее.

Выборочное стандартное отклонение $\sqrt{s^{2*}}$, где s^{2*} — несмещённая выборочная дисперсия, $s^{2*} = \frac{1}{n-1}\sum\limits_{i=1}^{n}(x_i - \bar{m})^2$, по выборке **x** можно найти при помощи функции **sd()**.

Пример 20. *Построим выборку размером $n = 100$ и найдём её выборочные среднее и стандартное отклонение.*

```
> x=rnorm(100,3,2)
> mean(x)
[1] 2.644506
> sd(x)
[1] 2.061032
```

Теперь вычислим взвешенное среднее для выборки **y**.

```
y=rbinom(200,10,0.5)
```

Построим статистический ряд по заданной выборке:

```
> table(y)
y
 1  2  3  4  5  6  7  8  9
 4  8 27 33 44 46 23 13  2
```

Таким образом, элементы выборки $\mathbf{y}$ *— это числа* $1, 2, 3, 4, 5, 6, 7, 8, 9$. *В качестве вектора весов зададим частоты появления этих элементов.*

```
> y1=1:9
> w=as.vector(table(y))
```

Взвешенное среднее:

```
> weighted.mean(y1,w)
[1] 5.06
```

В данном примере, это взвешенное среднее совпадает с выборочным средним:

```
> mean(y)
[1] 5.06
```

Таким образом, при помощи функции **weighted.mean** *можно вычислять выборочное среднее по статистическому ряду.*

Функции var(), cov(), cor() и cov2cor()

Здесь будут рассмотрены функции, позволяющие находить выборочную дисперсию, ковариацию и коэффициент корреляции для заданных выборок.

```
var(x, y = NULL, na.rm = FALSE)
cov(x, y = NULL, use = "everything",
method = c("pearson", "kendall", "spearman"))
cor(x, y = NULL, use = "everything",
method = c("pearson", "kendall", "spearman"))
cov2cor(V)
```

Функция **var()** вычисляет выборочную дисперсию по выборке $\mathbf{x}$, если $\mathbf{x}$ — числовой вектор, и ковариационную матрицу, если $\mathbf{x}$ — матрица. В последнем случае каждый столбец матрицы $\mathbf{x}$ рассматривается как отдельная выборка.

Если задан второй аргумент $\mathbf{y}$ и если $\mathbf{x}$ и $\mathbf{y}$ — вектора, то строится ковариационная матрица размером 2×2. Если $\mathbf{x}$ и $\mathbf{y}$ — матрицы или таблицы данных (одинаковой размерности), то снова строится ковариационная матрица, элементы которой — ковариации столбца матрицы $\mathbf{x}$ и столбца матрицы $\mathbf{y}$.

Функция **cov()** позволяет построить ковариационную матрицу для заданных выборок, а функция **cor()** — матрицу коэффициентов корреляции. Нужно отметить, что для этих функция желательно, чтобы один из двух аргументов —$\mathbf{x}$ или $\mathbf{y}$ — был матрицей.

Наконец, функция **cov2cor()** создаёт корреляционную матрицу на основе заданной ковариационной.

Рассмотрим аргументы этих функций:

- **x** и **y** — числовые векторы, матрицы или таблицы данных, причём аргумент **x** — обязательный.

- **na.rm** — логический аргумент — позволяет исключать из рассмотрения отсутствующие значения — **NA**.

- **use** — дополнительный символьный аргумент, определяющий как вычислять ковариацию или коэффициент корреляции при отсутствующих значениях (**NA**). Его возможные значения (полностью или сокращённо):

  - **«everything»** (по умолчанию) — **NA** остаются в выборке и учитываются при нахождении выборочной ковариации (коэффициента корреляции). Если **NA** есть, то результатом будет также **NA**.
  - **«all.obs»** — **NA** остаются ка элементы выборки, но по вычислении выводится сообщение об ошибке
  - **«complete.obs»** — **NA** не рассматриваются при вычислениях. Но если вся выборка состоит из **NA**, то выводится сообщение об ошибке.
  - **«na.or.complete»** — аналогично предыдущему, но в случае, если вся выборка состоит из **NA**, результатом будет **NA**.
  - **«pairwise.complete.obs»** — при нахождении ковариации или корреляции, если хотя бы одна из пары переменных принимает значение **NA**, то вся пара значений отбрасывается. Используется для **cov()**. если только **method=«pearson»**.

- **method** — символьный аргумент, определяющий на основе какого метода нужно вычислять коэффициент корреляции. Названия методов (полностью или сокращённо):

  - **«pearson»** (по умолчанию) — вычисление обычной выборочной ковариации или коэффициента корреляции.
  - **«kendall»** и **«spearman»** — ранговые коэффициенты корреляции.

- **V** — симметричная числовая матрица (положительно определённая), рассматриваемая в качестве матрицы ковариаций и преобразуемая в матрицу коэффициентов корреляции.

Пример 21. *Создадим выборку из 500 элементов, ТФР которой подчиняется биномиальному закону $B(k,p)$, где $k = 10$ — число испытаний Бернулли, а $p = 0.5$ — вероятность успеха в одном испытании Бернулли. Для этой выборки найдём дисперсию.*

```
> set.seed(0)
> x=rbinom(500,10,0.5)
> var(x)
[1] 2.303824
> cov(x,y=x)
[1] 2.303824
```

Теперь разобьём выборку на пять частей (как будто у нас 5 выборок) и представим результат в виде матрицы. Построим матрицу ковариаций.

```
> x1=matrix(x,ncol=5)
> cov(x1)
            [,1]        [,2]        [,3]        [,4]        [,5]
[1,]  1.79232323  0.05818182  0.35797980 -0.27151515  0.12121212
[2,]  0.05818182  2.00040404 -0.02262626 -0.01535354  0.13131313
[3,]  0.35797980 -0.02262626  2.16606061  0.27393939 -0.06060606
[4,] -0.27151515 -0.01535354  0.27393939  2.55313131 -0.23232323
[5,]  0.12121212  0.13131313 -0.06060606 -0.23232323  2.90909091
> cor(x1)
            [,1]         [,2]        [,3]         [,4]        [,5]
[1,]  1.00000000  0.030727007  0.18168332 -0.126925630  0.05308346
[2,]  0.03072701  1.000000000 -0.01086973 -0.006793803  0.05443405
[3,]  0.18168332 -0.010869728  1.00000000  0.116488382 -0.02414360
[4,] -0.12692563 -0.006793803  0.11648838  1.000000000 -0.08524667
[5,]  0.05308346  0.054434050 -0.02414360 -0.085246669  1.00000000
```

Увеличим размерность исходной выборки в 10 раз.

```
> set.seed(0)
> x=rbinom(5000,10,0.5)
> var(x)
[1] 2.622585
> x1=matrix(x,ncol=5)
> var(x1)
            [,1]        [,2]        [,3]        [,4]        [,5]
[1,]  2.50877978 -0.07256056 -0.06449850  0.03971471  0.05762663
[2,] -0.07256056  2.66953353 -0.07131932  0.08868869  0.01798599
[3,] -0.06449850 -0.07131932  2.53917518 -0.04649650  0.06213614
[4,]  0.03971471  0.08868869 -0.04649650  2.85723223 -0.07875375
[5,]  0.05762663  0.01798599  0.06213614 -0.07875375  2.53872973
 var(x1,y=x1)
            [,1]        [,2]        [,3]        [,4]        [,5]
```

```
[1,]  2.50877978 -0.07256056 -0.06449850  0.03971471  0.05762663
[2,] -0.07256056  2.66953353 -0.07131932  0.08868869  0.01798599
[3,] -0.06449850 -0.07131932  2.53917518 -0.04649650  0.06213614
[4,]  0.03971471  0.08868869 -0.04649650  2.85723223 -0.07875375
[5,]  0.05762663  0.01798599  0.06213614 -0.07875375  2.53872973
> cov(x1)
            [,1]        [,2]        [,3]        [,4]        [,5]
[1,]  2.50877978 -0.07256056 -0.06449850  0.03971471  0.05762663
[2,] -0.07256056  2.66953353 -0.07131932  0.08868869  0.01798599
[3,] -0.06449850 -0.07131932  2.53917518 -0.04649650  0.06213614
[4,]  0.03971471  0.08868869 -0.04649650  2.85723223 -0.07875375
[5,]  0.05762663  0.01798599  0.06213614 -0.07875375  2.53872973
> cor(x1)
            [,1]         [,2]        [,3]        [,4]         [,5]
[1,]  1.00000000 -0.028038299 -0.02555477  0.01483363  0.022834089
[2,] -0.02803830  1.000000000 -0.02739323  0.03211277  0.006908891
[3,] -0.02555477 -0.027393225  1.00000000 -0.01726240  0.024473139
[4,]  0.01483363  0.032112770 -0.01726240  1.00000000 -0.029240868
[5,]  0.02283409  0.006908891  0.02447314 -0.02924087  1.000000000
```

Преобразуем ковариационную матрицу в матрицу корреляций.

```
> V=cov(x1)
> cov2cor(V)
            [,1]         [,2]        [,3]        [,4]         [,5]
[1,]  1.00000000 -0.028038299 -0.02555477  0.01483363  0.022834089
[2,] -0.02803830  1.000000000 -0.02739323  0.03211277  0.006908891
[3,] -0.02555477 -0.027393225  1.00000000 -0.01726240  0.024473139
[4,]  0.01483363  0.032112770 -0.01726240  1.00000000 -0.029240868
[5,]  0.02283409  0.006908891  0.02447314 -0.02924087  1.000000000
```

*Рассмотрим случай, когда оба аргумента — **x** и **y** функции* **cov()** *— матрицы.*

```
> x2=matrix(x,ncol=5,byrow=T)
> cov(x1,y=x2)
             [,1]         [,2]        [,3]         [,4]         [,5]
[1,] -0.122986987  0.135512513 -0.18543043 -0.053756757  0.082110110
[2,]  0.002794795 -0.034890891  0.13449449  0.004388388 -0.107091091
[3,] -0.058202202 -0.076034034  0.12435435 -0.001639640  0.007095095
[4,]  0.213013013  0.007732733  0.04642142 -0.059334334  0.054954955
[5,] -0.040736737 -0.007304304 -0.08864364 -0.066483483  0.173265265
```

```
> cov(x1,y=x1)
            [,1]        [,2]        [,3]        [,4]        [,5]
[1,]  2.50877978 -0.07256056 -0.06449850  0.03971471  0.05762663
[2,] -0.07256056  2.66953353 -0.07131932  0.08868869  0.01798599
[3,] -0.06449850 -0.07131932  2.53917518 -0.04649650  0.06213614
[4,]  0.03971471  0.08868869 -0.04649650  2.85723223 -0.07875375
[5,]  0.05762663  0.01798599  0.06213614 -0.07875375  2.53872973
```

Функция cov.wt()

```
cov.wt(x, wt = rep(1/nrow(x), nrow(x)), cor = FALSE, center = TRUE,
method = c("unbiased", "ML"))
```

Данная функция позволяет вычислить взвешенные ковариационную и корреляционную матрицы.

Аргументы функции:

- **x** — числовая матрица или таблица данных. Как правило, строки - это наблюдения, а столбцы - переменные (случайные величины), т.е. столбец представляет из себя выборку.
- **wt** — неотрицательный и ненулевой вектор весов, его длина обязательно должна совпадать с количеством строк аргумента **x**.
- **cor** — логический аргумент — нужно ли помимо взвешенной ковариационной матрицы вычислять и взвешенную корреляционную матрицу.
- **center** — логический или числовой аргумент. Определяет центрирование при вычислении ковариации. Если **center** = **TRUE**, то используется взвешенное выборочное среднее, если же **center** = **FALSE**, то в качестве центра используется ноль. Если **center** — числовой аргумент, то это должен быть вектор с длиной, равной числу столбцов аргумента **x**.
- **method** — выбор метода для представления результатов. Если **method** = **«unbiased»**, то строится несмещённая ковариационная матрица (матрица делится на $1 - \sum_{i=1}^{n} w_i^2$, w_i — вес i-го элемента выборки , $i = \overline{1,n}$). В противном случае — **method** = **«ML»** — ковариационная матрица не делится на данный множитель, т.е. строится смещённая ковариационная матрица.

Стоит отметить, что если веса всех Элементов выборки одинаковы, то строится обычная матрица выборочных ковариаций.

Результат работы функции **cov.wt()** — это список, состоящий из следующих полей:

- **cov** — выборочная взвешенная ковариационная матрица;
- **center** — выборочные взвешенные средние;
- **n.obs** — размер выборки (число строк в **x**);
- **wt** — веса, использовавшиеся при вычислении; возвращаются, только если были заданы;
- **cor** — матрица выборочных взвешенных коэффициентов корреляции.

Пример 22.
```
> (xy <- cbind(x = 1:10, y = c(1:3, 8:5, 8:10)))
       x  y
 [1,]  1  1
 [2,]  2  2
 [3,]  3  3
 [4,]  4  8
 [5,]  5  7
 [6,]  6  6
 [7,]  7  5
 [8,]  8  8
 [9,]  9  9
[10,] 10 10
> w1 <- c(0,0,0,1,1,1,1,1,0,0)
> cov.wt(xy, wt = w1) # i.e. method = "unbiased"
$cov
     x    y
x  2.5 -0.5
y -0.5  1.7
$center
  x   y
6.0 6.8
$n.obs
[1] 10
$wt
 [1] 0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0

> cov.wt(xy, wt = w1, method = "ML", cor = TRUE)
$cov
     x     y
x  2.0 -0.40
y -0.4  1.36
```

```
$center
  x   y
6.0 6.8
$n.obs
[1] 10
$wt
 [1] 0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0
$cor
           x          y
x  1.0000000 -0.2425356
y -0.2425356  1.0000000
```

Функции quantile() и IQR()

```
quantile(x, probs = seq(0, 1, 0.25), na.rm = FALSE,
names = TRUE, type = 7, ...)
IQR(x, na.rm = FALSE)
```

Функция **quantile()** позволяет находить по выборке квантили Q_α заданного размера α $(0 \leq \alpha \leq 1)$.

Аргументы функции:

- **x** — числовой вектор — выборка. Для получения результатов не должен содержать **NA** и **NaN**.
- **probs** — числовой вектор — размеры необходимых квантилей. По умолчанию вычисляются квантили Q_α для $\alpha = 0, 0.25, 0.5, 0.75, 1$, т.е. квартили.
- **na.rm** — логический аргумент — нужно ли исключить из выборки элементы типа **NA** и (или) **NaN**. По умолчанию — нет.
- **names** — логический аргумент — нужно ли присваивать имена выводимым результатам (т.е. нужно ли указывать, что за квантили посчитаны)
- **type** — положительное целое число, принимающее значения от 1 до 9, тем самым определяющее алгоритм вычисления квантилей.

Алгоритмы вычисления квантилей. Все выборочные квантили вычисляются как взвешенные средние порядковых статистик. Выборочный квантиль $Q_\alpha(i)$ заданного размера α и типа i определяется по формуле:

$$Q_\alpha(p) = (1 - \gamma)x_j + \gamma x_{j+1},$$

где $1 \leq i \leq 9$, $\frac{j-m}{n} \leq \alpha < \frac{j-m+1}{n}$, x_j — j-я порядковая статистика, n — размер выборки, γ — функция от $j = \lfloor n\alpha + m \rfloor$ и $g = n\alpha + m - j$, m — константа, определяемая типом алгоритма расчёта квантиля.

Для типов 1–3 $Q_\alpha(i)$ является разрывной (дискретной) функцией от α и $m = 0$ для $i = 2$ или $i = 3$, и $m = -\frac{1}{2}$ для $i = 3$.

- **i=1**. $Q_\alpha(i)$ — обратная функция к эмпирической функции распределения. При $g = 0$ — $\gamma = 0$, для остальных значений g — γ принимает значение 1.

- **i=2**. Аналогично предыдущему случаю, но производится усреднение в точках разрыва. При $g = 0$ — $\gamma = 0.5$, для остальных значений g — γ принимает значение 1.

- **i=3**. Используется **SAS** определение — ближайшая чётная порядковая статистика (???? nearest even order statistic ????). При $g = 0$ и чётном j —- $\gamma = 0$, для остальных значений g — γ принимает значение 1.

Для алгоритмов 4–9 $Q_\alpha(i)$ является непрерывной функцией от α, при этом $\gamma = g$, константа m для каждого типа своя. Выборочный квантиль строится посредством линейной интерполяции между точками (α_k, x_k), x_k — k-я порядковая статистика.

- **i=4**. $m = 0$, $\alpha_k = \frac{k}{n}$. Интерполяция эмпирической функции распределения.

- **i=5**. $m = 0.5$, $\alpha_k = \frac{k-0.5}{n}$.

- **i=6**. $m = \alpha$, $\alpha_k = \frac{k}{n+1}$. Используется в **Minitab** и **SPSS**.

- **i=7**. $m = 1 - \alpha$, $\alpha_k = \frac{k-1}{n-1}$. Используется в **R** и **S**.

- **i=8**. $m = \frac{\alpha+1}{3}$, $\alpha_k = \frac{k-1/3}{n+1/3}$.

- **i=9**. $m = \alpha/4 + 3/8$, $\alpha_k = \frac{k-3/8}{n+1/4}$.

Пример 23. *Сначала вычислим квартили.*

```
> set.seed(0)
> quantile(x <- rnorm(1000))
         0%         25%         50%         75%        100%
-3.23638573 -0.70845589 -0.05887078  0.68763873  3.26641452
```

А затем квантили заданного размера

```
> quantile(x, probs = c(0.1, 0.5, 1, 2, 5, 10, 50, 75, 100, NA)/100)
       0.1%        0.5%          1%          2%          5%         10%
-3.08251764 -2.33505063 -2.08579744 -1.82308655 -1.59810034 -1.28895979
50%         75%        100%
-0.05887078  0.68763873  3.26641452          NA
```

Теперь для небольшой выборки размера 100 вычислим квантили, задавая различные типы алгоритмов.

```
> set.seed(0)
> x=rbinom(100,100,0.5);x
  [1] 52 49 47 51 48 50 57 50 52 46 46 51 56 39 54 49 50 52 51 51 55 38 58 52 41
 [26] 51 61 52 50 48 50 51 46 41 51 54 53 53 57 44 48 47 42 52 51 48 49 42 54 40
 [51] 46 42 49 49 52 57 50 51 54 46 47 56 47 58 55 50 54 50 45 59 50 59 54 50 44
 [76] 47 58 49 46 42 49 50 49 59 61 50 60 54 53 53 49 51 51 49 55 57 50 52 52 44
> quantile(x,probs=c(0,1,2,3,4)/4,type=1)
  0%  25%  50%  75% 100%
  38   48   50   53   61
> quantile(x,probs=c(0,1,2,3,4)/4,type=2)
  0%  25%  50%  75% 100%
38.0 48.0 50.0 53.5 61.0
> quantile(x,probs=c(0,1,2,3,4)/4,type=3)
  0%  25%  50%  75% 100%
  38   48   50   53   61
> quantile(x,probs=c(0,1,2,3,4)/4,type=4)
  0%  25%  50%  75% 100%
  38   48   50   53   61
> quantile(x,probs=c(0,1,2,3,4)/4,type=5)
  0%  25%  50%  75% 100%
38.0 48.0 50.0 53.5 61.0
> quantile(x,probs=c(0,1,2,3,4)/4,type=6)
  0%  25%  50%  75% 100%
  38   48   50   53   61
> quantile(x,probs=c(0,1,2,3,4)/4,type=7)
   0%   25%   50%   75%  100%
38.00 48.00 50.00 53.25 61.00
> quantile(x,probs=c(0,1,2,3,4)/4,type=8)
      0%      25%      50%      75%     100%
38.00000 48.00000 50.00000 53.58333 61.00000
> quantile(x,probs=c(0,1,2,3,4)/4,type=9)
     0%     25%     50%     75%    100%
38.0000 48.0000 50.0000 53.5625 61.0000
```

Функция **IQR(X)** вычисляет межквартильное расстояние, т.е. **IQR(x) = quantile(x,3/4) - quantile(x,1/4)**.

Пример 24. *Найдём межквартильное расстояние для заданной выборки.*

```
> set.seed(0)
> x=rbinom(100,100,0.5);x
  [1] 52 49 47 51 48 50 57 50 52 46 46 51 56 39 54 49 50 52 51 51 55 38 58 52 41
 [26] 51 61 52 50 48 50 51 46 41 51 54 53 53 57 44 48 47 42 52 51 48 49 42 54 40
 [51] 46 42 49 49 52 57 50 51 54 46 47 56 47 58 55 50 54 50 45 59 50 59 54 50 44
 [76] 47 58 49 46 42 49 50 49 59 61 50 60 54 53 53 49 51 51 49 55 57 50 52 52 44
> quantile(x,probs=c(0,1,2,3,4)/4,type=7)
   0%   25%   50%   75%  100%
38.00 48.00 50.00 53.25 61.00
> IQR(x)
[1] 5.25
```

Функции median() и mad()

```
median(x, na.rm = FALSE)
mad(x, center = median(x), constant = 1.4826, na.rm = FALSE,
low = FALSE, high = FALSE)
```

Функция **median()** находит медиану по заданной выборке **x**. Если объём выборки нечётный, то результат — единственное значение медианы, если объём выборки чётный, то результат — два значения.

Функция **mad()** вычисляет асимптотически нормальное абсолютное медианное отклонение. Её аргументы:

- **x** — выборка.
- **center** — числовой аргумент — цент, относительно которого вычисляется абсолютное отклонение (по умолчанию - медиана).
- **constant** — нормировочная константа, равная $1/\varphi_{0.75}$, $\varphi_{0.75}$ — 0.75-квантиль стандартного нормального закона. Эта константа нужна для того, чтобы абсолютное медианное отклонение было несмещённой оценкой среднего квадратичного отклонения для асимптотически нормальной выборки.
- **low** и **high** — логические аргументы. В случае чётной выборки берётся не среднее значение двух медиан, а либо меньшая медиана (**low = TRUTH**), либо верхняя (**high = TRUE**).

Пример 25. *Найдём медиану и абсолютное медианное отклонение выборки.*

```
> set.seed(0)
> x=rbinom(100,100,0.5);x
  [1] 52 49 47 51 48 50 57 50 52 46 46 51 56 39 54 49 50 52 51 51 55 38 58 52 41
 [26] 51 61 52 50 48 50 51 46 41 51 54 53 53 57 44 48 47 42 52 51 48 49 42 54 40
 [51] 46 42 49 49 52 57 50 51 54 46 47 56 47 58 55 50 54 50 45 59 50 59 54 50 44
 [76] 47 58 49 46 42 49 50 49 59 61 50 60 54 53 53 49 51 51 49 55 57 50 52 52 44
> median(x)
[1] 50
> mad(x)
[1] 4.4478
> mad(x,low=T)
[1] 4.4478
> mad(x,high=T)
[1] 4.4478
```

Функции summary() и fivenum()

```
summary(x, na.rm = TRUE)
fivenum(x, na.rm = TRUE)
```

Функции **summary** и **fivenum** выводят вектора, содержащие ряд характеристик выборки. **summary** — минимальное и максимальное значения выборки, медиана и среднее, первый и третий квартили. **fivenum** — максимальное и минимальное значение выборки, медиана, нижняя и верхняя отсечки — медиана значений левее (правее) медианы выборки (приблизительно первый и третий выборочные квартили).

Пример 26. *Построим выборку и найдём её характеристики при помощи* **summary** *и* **fivenum**.

```
set.seed(0)
x=rbinom(100,100,0.5)
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  38.00   48.00   50.00   50.44   53.25   61.00
> fivenum(x)
[1] 38.0 48.0 50.0 53.5 61.0
```

Функция ecdf()

```
ecdf(x)
```

Функция **ecdf(x)** по выборке **x** строит эмпирическую функцию распределения (ЭФР) $F_n^*(x) = \sum\limits_{x_i^* \le x} \nu_i^*$, где x_i^* — элементы статистического ряда по выборке **x**, а ν_i^* — частоты появления этих элементов.

Пример 27. *По выборке, ТФР которой подчиняется биномиальному закону, построим эмпирическую функцию распределения.*

```
> set.seed(0)
> x=rbinom(100,100,0.5);x
  [1] 52 49 47 51 48 50 57 50 52 46 46 51 56 39 54 49 50 52 51 51 55 38 58 52 41
 [26] 51 61 52 50 48 50 51 46 41 51 54 53 53 57 44 48 47 42 52 51 48 49 42 54 40
 [51] 46 42 49 49 52 57 50 51 54 46 47 56 47 58 55 50 54 50 45 59 50 59 54 50 44
 [76] 47 58 49 46 42 49 50 49 59 61 50 60 54 53 53 49 51 51 49 55 57 50 52 52 44
> Fn=ecdf(x);Fn
Empirical CDF
Call: ecdf(x)
 x[1:23] =     38,     39,     40,  ...,     60,     61
> summary(Fn)
Empirical CDF:    23 unique values with summary
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  38.00   44.50   50.00   49.78   55.50   61.00
> plot(Fn)
```

График ЭФР.

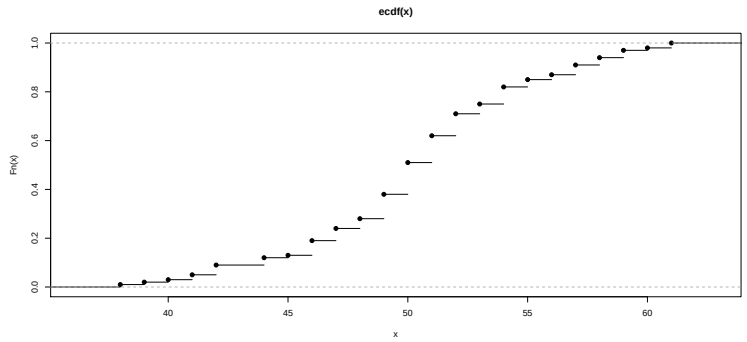

Рис. 2.7: График эмпирической функции распределения

Следует отметить, что функция **ecdf(x)** только вычисляет значения ЭФР, но не выводит. Для получения значений ЭФР нужно воспользоваться функцией **plot()**.

Чтобы узнать, какое значение ЭФР соответствует каждому элементу исходной выборки **x**, нужно некоторой переменной присвоить результат функции **ecdf(x)**, а затем вызвать эту переменную, но уже как функцию от выборки.

```
> Fn=ecdf(x);Fn
Empirical CDF
Call: ecdf(x)
 x[1:23] =     38,     39,     40,  ...,     60,     61
> Fn(x)
  [1] 0.71 0.38 0.24 0.62 0.28 0.51 0.91 0.51 0.71 0.19 0.19 0.62 0.87 0.02 0.82
 [16] 0.38 0.51 0.71 0.62 0.62 0.85 0.01 0.94 0.71 0.05 0.62 1.00 0.71 0.51 0.28
 [31] 0.51 0.62 0.19 0.05 0.62 0.82 0.75 0.75 0.91 0.12 0.28 0.24 0.09 0.71 0.62
 [46] 0.28 0.38 0.09 0.82 0.03 0.19 0.09 0.38 0.38 0.71 0.91 0.51 0.62 0.82 0.19
 [61] 0.24 0.87 0.24 0.94 0.85 0.51 0.82 0.51 0.13 0.97 0.51 0.97 0.82 0.51 0.12
 [76] 0.24 0.94 0.38 0.19 0.09 0.38 0.51 0.38 0.97 1.00 0.51 0.98 0.82 0.75 0.75
 [91] 0.38 0.62 0.62 0.38 0.85 0.91 0.51 0.71 0.71 0.12
```

Чтобы узнать значения выборки, по которым строится ЭФР (т.е. все различные элементы выборки), достаточно воспользоваться функцией **knots()**

```
> knots(Fn)
 [1] 38 39 40 41 42 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61
```

Можно улучшить график ЭФР.

```
plot(Fn, verticals=TRUE, col.points='blue', col.hor='red', col.vert='bisque')
```

Здесь **verticals** — аргумент, отвечающий за построение вертикальных линий, соединяющих ступени графика ЭФР; **col.points** — аргумент, отвечающий за цвет точек в узловых значениях ЭФР; **col.hor** и **col.vert** — аргументы, отвечающие за цвет горизонтальной и вертикальной составляющих «ступеней» графика функции.

Наконец, для построения графика ЭФР можно воспользоваться функцией **plot.ecdf()**:

```
set.seed(0)
x=rbinom(100,100,0.5)
Fn=ecdf(x)
op=par(mfrow=c(1,2))
plot.ecdf(Fn)
plot.ecdf(x)
```

Здесь неважно, используем ли мы переменную, которой присвоен результат функции **ecdf()**, или же саму выборку (см. 2.9).

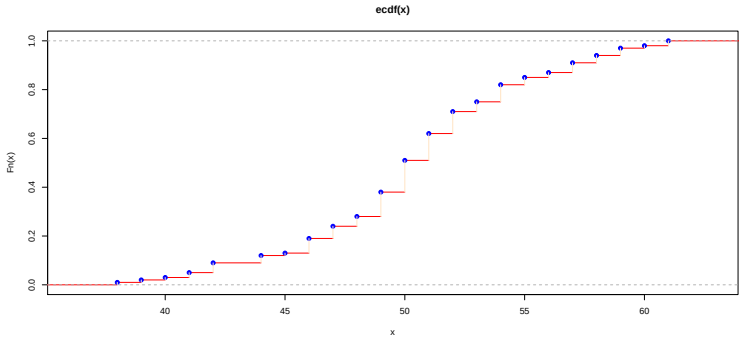

Рис. 2.8: Другой вид графика ЭФР

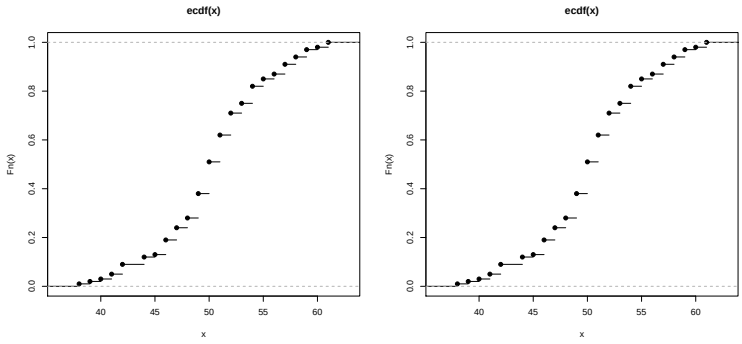

Рис. 2.9: Построение графика ЭФР при помощи **plot.ecdf()**

2.2.2 Графический анализ числовых данных. Функции hist(), boxplot(), qqnorm() и qqplot

Построение гистограммы

Гистограммы в **R** строятся при помощи следующей функции:

```
hist(x, breaks = "Sturges",
freq = NULL, probability = !freq, include.lowest = TRUE, right = TRUE,
density = NULL, angle = 45, col = NULL, border = NULL,
main = paste("Histogram of" , xname), xlim = range(breaks), ylim = NULL,
xlab = xname, ylab, axes = TRUE, plot = TRUE, labels = FALSE,
nclass = NULL, ...)
```

Рассмотрим подробно её аргументы (некоторые из них уже знакомы):

- **x** — числовой вектор (выборка), по которому строится гистограмма.

- **breaks** — параметр, отвечающий за разбиение на интервалы. Возможны следующие варианты:
  - числовой вектор, определяющий границы интервалов разбиения;
  - число, задающее количество интервалов разбиения;
  - символьная переменная, отвечающая за выбор алгоритма разбиения на интервалы;
  - функция, вычисляющая количество интервалов разбиения[2].
- **freq** и **probability** — два логических и альтернативных друг другу аргумента (не рекомендуется задать одновременно оба этих аргумента). Если **freq** = **TRUE** (или, соответственно, **probability** = **FALSE**), то строится гистограмма частот. Если же **freq** = **FALSE** (или, соответственно, **probability** = **TRUE**) — гистограмма относительных частот. Если аргумент **probability** не задан и интервалы разбиения одинаковые, то по умолчанию строится гистограмма частот, т.е. **freq** = **TRUE**.
- **include.lowest** — логический аргумент. Минимальный элемент выборки включается в первый интервал в качестве левой границы. Используется только в том случае, если **breaks** — числовой вектор. В противном случае выдаётся предупреждение.
- **right** — логический аргумент. Если **right** = **TRUE**, то интервалы разбиения имеют вид $(a_i; a_{i+1}]$.
- **density** и **angle** — числовые аргументы, отвечающие за плотность и угол наклона штриховки столбцов гистограммы.
- **col** — символьный или числовой аргумент, задающий либо цвет столбцов гистограммы (если не определены аргументы **density** и **angle**), либо цвет штриховки. По умолчанию значение **col** = **NULL**, т.е. столбцы гистограммы не закрашены.
- **border** — цвет границы столбцов гистограммы.
- **main** — основной заголовок гистограммы.
- **xlim** и **ylim** — границы осей гистограммы. По умолчанию для оси **X** в качестве границ используются границы разбиения, а границы оси **Y** — отсутствуют.
- **xlab** и **ylab** — названия осей.

[2]В последних трёх случаях рассматривается только число интервалов разбиения

- **axes** — логический аргумент — построение осей (если выводится и сама гистограмма).

- **plot** — логический аргумент — отвечает за построение гистограммы. Если **plot** = **TRUE**, то строится гистограмма. В противном случае функция **hist()** возвращает список из интервалов разбиения и числа элементов выборки, попавших в интервалы. В последнем случае выдаётся предупреждение, если были определены графические аргументы при задании функции **hist()**.

- **labels** — логический или символьный аргумент. Если **labels** = **TRUE**, то над каждым столбцом гистограммы выводится число (или доля) элементов выборки, попавших в данный интервал разбиения. Если **labels** — символьный аргумент (вектор), то над столбцами выводятся элементы этого аргумента.

- **nclass** — положительное целое число. Аргумент отвечает за количество интервалов разбиения и введён для совместимости функции с языком **S** (**S-PLUS**).

- **...** — дополнительные графические аргументы, отвечающие за заголовки и оси.

Пример 28. *Построим выборку объёма 500 с теоретической функцией распределения, подчиняющейся биномиальному закону с параметрами* $p = 0.5$ *и* $n = 20$. *Для этой выборки построим различные гистограммы (см.‘2.10).*

```
x=rbinom(500,20,0.5)
par(bg = "white")
split.screen(c(2,1))
split.screen(c(1,2), screen = 1)
screen(3)
hist(x,col="gray")
screen(4)
hist(x,breaks=0:20,freq=F, density=6:26,angle=45,col='red')
split.screen(c(1,2), screen = 2)
screen(5)
hist(x,breaks=21,col=1:21,border='green',xlim=c(0,21),
ylim=c(0,100),labels=T)
screen(6)
hist(x,breaks=21,freq=F, col=1:21,border='green',xlim=c(0,21),
ylim=c(0,0.4),labels=T)
close.screen(all = TRUE)
```

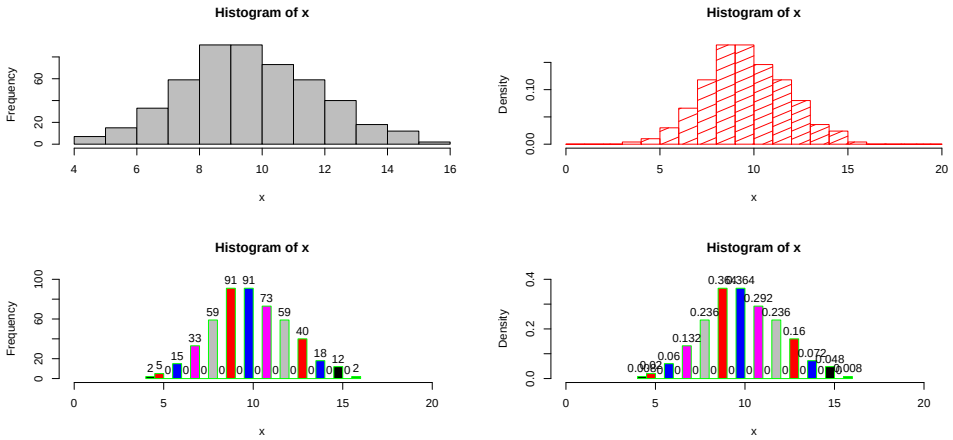

Рис. 2.10: Различные варианты задания гистограммы

Теперь воспользуемся функцией **hist()**, *не строя саму гистограмму.*

```
> hist(x,plot=F)
```

В результате получим список:

```
$breaks
 [1]  4  5  6  7  8  9 10 11 12 13 14 15 16

$counts
 [1]  7 15 33 59 91 91 73 59 40 18 12  2

$intensities
 [1] 0.014 0.030 0.066 0.118 0.182 0.182 0.146 0.118 0.080 0.036 0.024 0.004

$density
 [1] 0.014 0.030 0.066 0.118 0.182 0.182 0.146 0.118 0.080 0.036 0.024 0.004

$mids
 [1]  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5 14.5 15.5

$xname
[1] "x"

$equidist
[1] TRUE
```

```
attr(,"class")
[1] "histogram"
```

Элементы списка, возвращаемого функцией **hist(x,plot=F)** по выборке **x**:

- **breaks** — границы построенных интервалов разбиения.
- **counts** — количество элементов выборки, попавших в заданные интервалы разбиения.
- **intensities** — относительные частоты.
- **density** — аналогично **intensities**.
- **mids** — середины интервалов разбиения.
- **xname** — имя вектора с выборкой.
- **equidist** — равные ли интервалы разбиения.

attr(,"class") — указание класса, к которому принадлежит выводимый объект.

Построение коробчатой («коробка с усами») диаграммы — boxplot

.

```
boxplot(x, ..., range = 1.5, width = NULL, varwidth = FALSE,
notch = FALSE, outline = TRUE, names, plot = TRUE,
border = par("fg"), col = NULL, log = "",
pars = list(boxwex = 0.8, staplewex = 0.5, outwex = 0.5),
horizontal = FALSE, add = FALSE, at = NULL)
```

Аргументы функции:

- **x** — единственный обязательный аргумент — либо числовой вектор, либо список (элементы которого — числовые вектора), по которому строится график. Если **x** — список, то в одном графическом окне для всех элементов списка строятся графики (см.‘2.12).
- **...** — дополнительные числовые аргументы (выборки), по которым в рамках одного графического окна строятся графики (см.‘2.13).

- **range** — числовой аргумент, определяющий, насколько далеко от основной части диаграммы (прямоугольника)могут находиться «усы» (вертикальные линии, соответствующие первому и третьему квартилям). Положительное значение аргумента — множитель межквартильного расстояния — максимальный разброс «усов». Если **range** = **0**, то расположение «усов» определяется минимальным и максимальным значением выборки

- **width** — числовой аргумент, задающий ширину прямоугольника диаграммы. Если строится несколько «коробок с усами», то **width** — числовой вектор.

- **varwidth** — логический аргумент. Если **varwidth** = **TRUE**, то ширина прямоугольника (прямоугольников) пропорциональна корню квадратному объёму выборки (выборок).

- **notch** — логический аргумент. Если **notch** = **TRUE**, то по бокам диаграмм делаются выемки. Если эти выемки не пересекаются (находятся на разных уровнях), то можно говорить о несовпадении медиан.

- **outline** — логический аргумент — нужно ли рисовать на графике элементы выборки, лежащие вне «усов».

- **names** — названия отдельных диаграмм — либо символьный аргумент, либо типа **expression**.

- **plot** — логический аргумент — либо строится диаграмма, либо выводятся результаты обработки выборки (см.‘31).

- **border** — цвет границ прямоугольника (прямоугольников), «усов», а также значений, лежащих за «усами».

- **col** — цвет прямоугольника (прямоугольников).

- **log** — символьный аргумент — нужно ли оси переводить в логарифмические.

- **pars** — список дополнительных аргументов, отвечающих за масштаб прямоугольника (**boxwex**), масштаб разброса «усов» (**staplewex**) и значений за «усами» (**outwex**). Последние два аргумента пропорциональны ширине прямоугольника.

- **horizontal** — логический аргумент — нужно ли строить горизонтальные диаграммы (по умолчанию — вертикальные).

- **add** — логический аргумент — нужно ли добавлять диаграмму к уже существующим.
- **at** — числовой аргумент (вектор) — порядок построения диаграмм на одном графике.

boxplot суммирует информацию по выборке и визуально её представляет. Разберём на примере (см.‘2.11).

Пример 29. *Построим выборку объёма 500 с теоретической функцией распределения, подчиняющейся биномиальному закону с параметрами* $p = 0.5$ *и* $n = 20$ *и для её графического анализа воспользуемся функцией* **boxplot** *(см.‘2.11).*

```
set.seed(0)
x=rbinom(500,20,0.5)
par(mfrow=c(1,2))
boxplot(x,width=2,col="gray")
boxplot(x,range=0,col="gray",horiz=F)
```

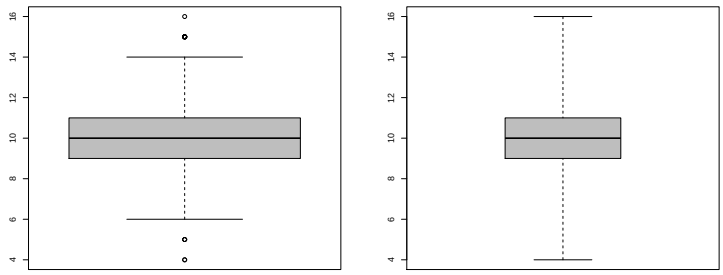

Рис. 2.11: Функция **boxplot**

Горизонтальная линия внутри закрашенного прямоугольника соответствует медиане выборки, верхняя и нижняя границы прямоугольника — это 0.75 и 0.25 квантили выборки (т.е. в прямоугольнике сосредоточено 50% выборки). Верхняя и нижняя вертикальные линии либо соответствуют максимальному и минимальному значению выборки, либо это есть отступы на полторы величины межквартильного расстояния (примерно два стандартных отклонения) вверх и вниз от медианы. Точки, лежащие вне этих линий, соответствуют экстремальным (наибольшим и наименьшим) значениям выборки. **boxplot** показывает распределение элементов выборки, а также её симметричность (асим-

метричность). Также **boxplot** полезен для выявления ошибок в предоставленных данных (как правило, ошибочные данные оказываются экстремальными значениями).

Пример 30. *Построим* **boxplot** *для выборок с различными ТФР используя список (см.‘2.12) или задавая несколько выборок (см.‘2.13)*

```
set.seed(0)
x=rbinom(500,20,0.5)
y1=runif(500,-2,2)
y2=rnorm(500,0,2)
y3=rexp(500,0.5)
z=list(x,y1,y2,y3)
boxplot(z,notch=TRUE)
```

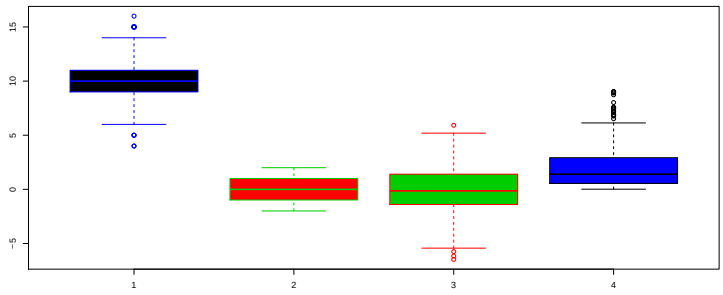

Рис. 2.12: **boxplot** для выборок с различными ТФР при помощи списка

```
set.seed(0)
x=rbinom(500,20,0.5)
y1=runif(500,-2,2)
y2=rnorm(500,0,2)
y3=rexp(500,0.5)
boxplot(x,y1,y2,y3)
```

Рассмотрим пример, когда функция **boxplot** представляет анализ выборки не графически, а выводит в консоли.

Пример 31. `set.seed(0)`
```
x=rbinom(500,20,0.5)
y1=runif(500,-2,2)
```

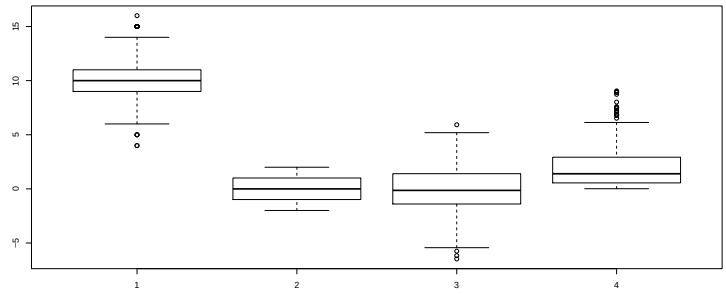

Рис. 2.13: **boxplot** для выборок с различными ТФР при помощи нескольких векторов

```
y2=rnorm(500,0,2)
y3=rexp(500,0.5)
z=list(x,y1,y2,y3)
boxplot(z,plot=F)

$stats
     [,1]        [,2]       [,3]       [,4]
[1,]    6 -1.99474137 -5.4318506 0.01392775
[2,]    9 -0.98401591 -1.3971417 0.55296942
[3,]   10 -0.00311991 -0.1388336 1.39749624
[4,]   11  1.00610008  1.4029322 2.92517649
[5,]   14  1.99972237  5.1915437 6.13467766

$n
[1] 500 500 500 500

$conf
         [,1]       [,2]        [,3]     [,4]
[1,]  9.85868 -0.1437410 -0.33668611 1.229877
[2,] 10.14132  0.1375012  0.05901893 1.565116

$out
 [1] 15.000000  5.000000 15.000000  5.000000 15.000000 15.000000 15.000000
 [8] 15.000000 16.000000 15.000000 15.000000 15.000000  5.000000 15.000000
[15] 15.000000  4.000000  4.000000 -5.750283  5.923487 -6.164727 -6.472771
[22]  7.078757  8.017729  8.921599  6.899314  9.044971  6.890356  8.911093
[29]  8.733675  7.591205  7.545854  6.524225  7.405684  6.790620  7.183892
```

```
[36]  9.018092  6.784458  7.503448  7.247282

$group
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
[39] 4

$names
[1] "1" "2" "3" "4"
```

Результат — список, состоящий из следующих полей:

- **stats** — матрица, число столбцов которой соответствует числу выборок, для которых применялась функция **boxplot**. Элементы каждого столбца (начиная с первого): минимальное значение выборки, первый квартиль, медиана, третий квартиль и максимальное значение выборки.
- **n** — объём выборки.
- **conf** — матрица, столбцы которой содержат верхнюю и нижнюю границу доверительного интервала для медианы (те значения, которые рисует аргумент **notch**).
- **out** — значения выборки, лежащие вне «усов», то есть до первого квартиля и после третьего квартиля.
- **group** — вектор той же длины, что и **out**, указывает к какой выборке принадлежат выходящие за «усы» значения.
- **names** — имена выборки (выборок).

Построение QQ-графиков

.

В этом подразделе рассмотрим функции, позволяющие графически проверять выборку на принадлежность нормальной генеральной совокупности, а также две выборки на однородность («квантиль-квантиль» графики).

Функция **qqnorm** сравнивает эмпирические квантили с теоретическими нормальными квантилями.

```
qqnorm(y, ylim, main = "Normal Q-Q Plot",
xlab = "Theoretical Quantiles", ylab = "Sample Quantiles",
plot.it = TRUE, datax = FALSE, ...)
```

Аргументы функции:

- **y** — выборка — единственный обязательный аргумент.
- **ylim** — графический параметр — границы оси **У**.
- **main** — название графика.
- **xlab** и **ylab** — подписи к осям.
- **plot.it** — логический аргумент — нужно ли строить график или нет.
- **datax** — логический аргумент — нужно ли выводить значения на оси **X**.
- `...` — дополнительные графические параметры.

Функция **qqline** по заданной выборке проводит линию на нормальном «квантиль-квантиль» графике через первый и третий квартили.

```
qqline(y, datax = FALSE, ...)
```

Пример 32. *Построим четыре выборки и проверим их при помощи функции* **qqnorm** *на нормальность.*

```
> set.seed(0)
> y=rnorm(500,2,2)
> y2=rcauchy(500)
> y3=rnorm(500)
> y4=runif(500,-5,5)
> par(mfrow=c(2,2))
> qqnorm(y)
> qqline(y)
> qqnorm(y2)
> qqline(y2)
> qqnorm(y3)
> qqline(y3)
> qqnorm(y4)
> qqline(y4)
```

Результаты представлены на рис.‘2.14.

Функция **qqplot** сравнивает эмпирические квантили двух выборок.

```
qqplot(x, y, plot.it = TRUE, xlab = deparse(substitute(x)),
ylab = deparse(substitute(y)), ...)
```

Её аргументы:

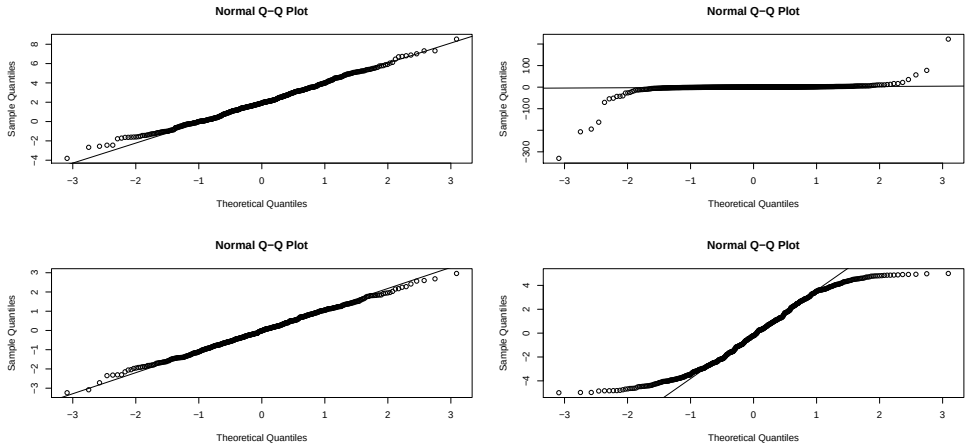

Рис. 2.14: Анализ выборок с помощью функции **qqnorm**

- **x** и **x** — две выборки.
- **plot.it** — логический аргумент — нужно ли строить график.
- **xlab** и **ylab** — подписи к осям — названия выборок.

Пример 33. *Построим различные выборки и сравним их при помощи функции* **qqplot**.

```
> set.seed(0)
> y2=rcauchy(500)
> y3=rnorm(500)
> y4=runif(500,-5,5)
> par(mfrow=c(2,2))
> qqplot(y2,y3)
> qqplot(y3,y4)
> qqplot(y2,y4)
> qqplot(y3,y3)
```

Результаты представлены на рис.‘2.15.

2.3 Оценки неизвестных параметров в R

В данном разделе рассмотрим, как с помощью пакета **R** находить точечные оценки неизвестных параметров, а именно реализации метода моментов и метода максимального правдоподобия (ММП). Сразу стоит отметить, что в базовой комплектации пакета **R** специально не предусмотрены функции для нахождения оценок неизвестных параметров.

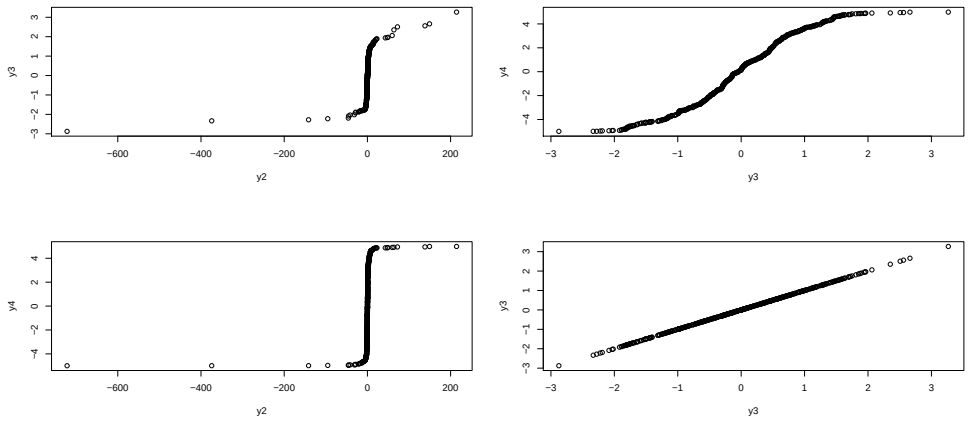

Рис. 2.15: Анализ выборок с помощью **qqplot**

2.3.1 Метод моментов. Функции uniroot, uniroot.all и multiroot

Рассмотрим два подхода к поиску в **R** неизвестных параметров эмпирической функции распределения генеральной совокупности методом моментов.

Первый подход связан с использованием функций, предназначенных для решения (нелинейных) уравнений и систем (нелинейных) уравнений. Нахождение оценки неизвестных параметров ТФР достаточно легко осуществимо. Для этого нужно составить систему уравнений (или одно уравнение — в зависимости от числа неизвестных параметров), приравняв выборочные моменты к соответствующим теоретическим.

Наиболее простой случай, это когда неизвестен только один параметр. Достаточно составить одно уравнение, решить аналитически и запрограммировать в **R**. Рассмотрим на примерах.

Пример 34. *Найдём неизвестные параметры для биномиальной выборки (вероятность успеха или число испытаний), пуассоновской и экспоненциальной выборок. Напомним, что теоретические моменты для указанных распределений имеют вид: np, λ и $\frac{1}{\lambda}$.*

Сгенерируем выборки размером 500.

```
> set.seed(0)
> x1=rbinom(500,20,0.6)
> x2=rpois(500,0.5)
> x3=rexp(500,0.5)
```

Составим уравнения:

$$p^* = \frac{\bar{m}}{n}, \quad n^* = \frac{\bar{m}}{p},$$

$$\lambda_1^* = \bar{m}, \quad \lambda_2^* = \frac{1}{\bar{m}},$$

здесь p^*, n^*, λ_1^* *и* λ_2^* *— оценки неизвестных параметров.*

```
> p=mean(x1)/20; p
[1] 0.5982
> n=mean(x1)/0.6; n
[1] 19.94
> lam1=mean(x2); lam1
[1] 0.536
> lam2=1/mean(x3); lam2
[1] 0.4780344
```

Если уравнение относительно неизвестного параметра распределения является нелинейным, то можно воспользоваться функцией **uniroot** пакета **stats** базовой комплектации **R** или функцией **uniroot.all** дополнительного пакета **rootSolve**. Если же есть система нелинейных уравнений, то она может быть решена с помощью функции **multiroot** пакета **rootSolve**. Эти функции рассмотрены в разделе «Решение нелинейных уравнений и систем нелинейных уравнений» главы 8 первой части данного пособия.

Функция uniroot

```
uniroot(f, interval, ...,
lower = min(interval), upper = max(interval),
f.lower = f(lower, ...), f.upper = f(upper, ...),
tol = .Machine$double.eps^0.25, maxiter = 1000)
```

Аргументы:

- **f** — функция, нуль которой (т.е. корень) вычисляется. Отметим, что нуль функции **f** ищется **только** по её первому аргументу.
- **interval** — числовой вектор — интервал, на котором ищется корень (необходимо, чтобы значения функции на концах этого интервала имели разные знаки).

- **lower** и **upper** — альтернативное задание интервала поиска **interval** через его начало и конец.
- **f.lower** и **f.upper** — граничные значения функции (по умолчанию значения функции **f** на границах интервала поиска).
- **tol** — желаемая точность.
- **maxiter** — максимальное число итераций.
- **...** — дополнительные аргументы.

Решение считается найденным, либо если значение функции в найденной точке x^* равняется нулю ($f(x^*) == 0$), либо если изменение значения x^* на следующей итерации меньше заданной точности **tol**. Если достигнут максимум итераций, а решение не найдено, то выдаётся предупреждение.

Результатом функции **uniroot** является список из четырёх компонент: искомое решение x^* — *root*, значение функции в найденной точке $f(x^*)$ — *f.root*, число итераций *iter* и точность решения *estim.prec*. Если x^* совпадает с одним из концов заданного интервала поиска, то тогда значение *estim.prec* — **NA**.

Существенный недостаток функции **uniroot** — это то, что ищется только одно решение на заданном интервале. Если существует несколько нулей функции, то будет выводиться только первый найденный.

Пример 35. *Найдём решение уравнения* $5x^2 - 10x = 0$ *сначала на отрезке* $[1; 3]$*, а потом на* $[0; 3]$.

```
> f=function(x,a=5,b=10){f=a*x^2-b*x}
> uniroot(f,c(1,3))
$root
[1] 2.000000

$f.root
[1] -2.678252e-06

$iter
[1] 6

$estim.prec
[1] 6.535148e-05

> uniroot(f,c(0,3))
```

```
$root
[1] 0

$f.root
[1] 0

$iter
[1] 0

$estim.prec
[1] 0
```

Во втором случае в качестве решения находится только $x = 0$.

Функция uniroot.all

Функция **uniroot.all**[3] устраняет недостатки **uniroot**, позволяя вычислять несколько корней на заданном интервале.

```
uniroot.all(f, interval, lower=min(interval), upper=max(interval),
  tol=.Machine$double.eps^0.2, maxiter=1000, n=100, ...)
```

Отличие заключается в введении аргумента **n** — число подинтервалов, на которые делится исходный интервал, и на каждом из этих подинтервалов ищется нуль функции. Результат вызова функции – вектор с найденными решениями.

Пример 36. *Вернёмся к предыдущему примеру и найдём решение уравнения* $5x^2 - 10x = 0$ *на отрезке* $[0; 3]$.

```
> f=function(x,a=5,b=10){f=a*x^2-b*x}
> uniroot.all(f,c(0,3),n=10)
[1] 0.000000 2.000000
```

Найдены оба решения. Отметим, что если задать слишком большое число подинтервалов поиска, то возможны погрешности в решении.

```
> uniroot.all(f,c(0,3))
[1] 0.000000 1.999999
```

Здесь $n = 100$ *(по умолчанию).*

[3]Пакет **rootSolve**

Функция multiroot

Функция **multiroot** позволяет найти решение (если оно существует) системы из n уравнений с n неизвестными.

```
multiroot(f, start, maxiter=100,
  rtol=1e-6, atol=1e-8, ctol=1e-8,
  useFortran=TRUE, positive=FALSE,
  jacfunc=NULL, jactype="fullint",
  verbose=FALSE, bandup=1, banddown=1, ...)
```

Рассмотрим только часть аргументов, необходимых для работы:

- **f** — функция, в которой задана система уравнений. Должна возвращать вектор той же длины, что и **start**.
- **start** — вектор начальных значений для искомых неизвестных.Если элементам **start** присвоены имена, то эти имена будут использоваться при выводе решения.
- **maxiter** — максимальное допустимое число итераций.
- **rtol** и **atol** — относительная и абсолютная погрешности — скаляр или вектор. Если вектора, то каждый их элемент задаёт погрешности при поиске соответствующей неизвестной.
- **ctol** — текущая погрешность — скаляр. Если между двумя итерациями изменение неизвестной меньше этого заданного параметра, то считается, что решение найдено.
- **positive** — логический аргумент. Если **positive=TRUE**, то предполагается, что решения системы — положительны.

Результатом вызова функции является список из четырёх компонент: *root* — решения системы, *f.root* — значения уравнений системы в найденных точках, *iter* — число итераций, *estim.precis* — точность полученного решения.

Рассмотрим работу этой функции на примере — найдём оценки неизвестных параметром биномиального и гамма распределений.

Пример 37. *Создадим выборки, подчиняющиеся биномиальному и гамма распределениям и найдём выборочные средние и дисперсии.*

```
set.seed(0)
x1=rbinom(500,20,0.6)
x2=rgamma(500,0.5,2)
y11=mean(x1)
y12=var(x1)
y21=mean(x2)
y22=mean(x2)
```

Составив системы уравнений:

$$\begin{cases} np = \bar{m}, \\ np(1-p) = s^{2*}. \end{cases} \qquad \begin{cases} \dfrac{\gamma}{\lambda} = \bar{m}, \\ \dfrac{\gamma}{\lambda^2} = s^{2*}. \end{cases}$$

Решим первую систему в **R**:

```
> f=function(x){c(F1=x[1]*x[2]-y11,F2=x[1]*x[2]*(1-x[2])-y12)}
> multiroot(f,c(10,0.5))
$root
[1] 19.0636602  0.6275815

$f.root
           F1            F2
-1.456613e-13 -1.248779e-12

$iter
[1] 6

$estim.precis
[1] 6.9722e-13
```

Меняя значения вектора начальных приближений, можно либо улучшить, либо ухудшить результат.

Аналитически решение для биномиального распределения имеет вид:

$$p = 1 - \frac{s^{2*}}{\bar{m}}, \quad n = \frac{\bar{m}^2}{\bar{m} - s^{2*}}.$$

Решая в **R**, *получим:*

```
> (p=1-y12/y11)
[1] 0.6275815
> (n=y11^2/(y11-y12))
[1] 19.06366
```

Для гамма распределения.

```
> f1=function(x){c(F1=x[1]/x[2]-y21,F2=x[1]/x[2]^2-y22)}
> multiroot(f1,c(2,0.6))
$root
[1] 0.2513997 1.0000000

$f.root
          F1           F2
1.005335e-11 8.700218e-11

$iter
[1] 7

$estim.precis
[1] 4.852777e-11
```

Аналитическое решение:

$$\lambda = \frac{\bar{m}}{s^{2*}}, \quad \gamma = \frac{\bar{m}^2}{s^{2*}}.$$

Расчёт в **R***:*

```
> (lambda=y21/y22)
[1] 1
> (gamma=y21^2/y22)
[1] 0.2513997
```

Но в случае нескольких неизвестных параметров метод моментов не всегда даёт хорошие результаты. Более лучшие оценки предоставляет метод максимального правдоподобия.

2.3.2 Метод моментов. Функции mmedist и fitdist пакета fitdistrplus

В пакете **fitdistrplus** реализованы две функции — **mmedist** и **fitdist**, позволяющие получать оценки неизвестных параметров методом моментов.

Функция fitdist

```
fitdist(data, distr, method="mme")
```

Здесь:

- **data** — числовой аргумент — выборка, подчиняющаяся ТФР с оцениваемыми параметрами.

- **distr** — символьный аргумент — название распределения — **"name"**, для которого должны быть определены функция плотности — **dname**, функция распределения — **pname** и функция вычисления квантилей — **qname**. Также значением аргумента **distr** может быть непосредственно функция плотности распределения.

- **method** — символьный аргумент — реализуется метод моментов.

Следует отметить, что данная функция позволяет найти оценки неизвестных параметров только для следующих распределений: нормальное — **"norm"**, логнормальное — **"lnorm"**, пуассоновское — **"pois"**, экспоненциальное — **"exp"**, гамма — **"gamma"**, отрицательно биномиальное — **"nbinom"**, геометрическое — **"geom"**, бета — **"beta"**, равномерное — **"unif"** и логистическое — **"logis"**. Для однопараметрических распределений оценка параметра находится из уравнения, в котором приравнены теоретическое и выборочное средние. Для двупараметрических распределений оценки являются решением системы из двух уравнений: приравнивание теоретических и выборочных средних (первое уравнение системы) и дисперсий (второе уравнение).

Результатом работы функции является список из компоненты **estimate** — полученные оценки параметров.

Пример 38. *Пусть имеется выборка, которая принадлежит нормальному закону с неизвестными параметрами* m *и* σ*. Найдём оценки этих параметров.*

```
> x1 <- c(6.4,13.3,4.1,1.3,14.1,10.6,9.9,9.6,15.3,22.1,13.4,
+ 13.2,8.4,6.3,8.9,5.2,10.9,14.4)
> fitdist(x1,"norm",method='mme')
Fitting of the distribution ' norm ' by matching moments
Parameters:
      estimate
mean 10.411111
sd    4.747033
```

Также результаты функции **fitdist** *можно вывести при помощи* **print()**

```
> x1 <- c(6.4,13.3,4.1,1.3,14.1,10.6,9.9,9.6,15.3,22.1,13.4,
+ 13.2,8.4,6.3,8.9,5.2,10.9,14.4)
> f1 <- fitdist(x1,"norm",method='mme')
> print(f1)
```

```
Fitting of the distribution ' norm ' by matching moments
Parameters:
      estimate
mean 10.411111
sd    4.747033
```

или **summary()**

```
> summary(f1)
Fitting of the distribution ' norm ' by matching moments
Parameters :
      estimate
mean 10.411111
sd    4.747033
```

Функция **plot()** *выводит в графическом окне гистограмму с графиком плотности ТФР, PP-график (график сравнения теоретических и эмпирических вероятностей), QQ-график (график сравнения эмпирических и теоретических квантилей), график сравнения ЭФР и ТФР (см. рис.2.16)*

```
> plot(f1)
```

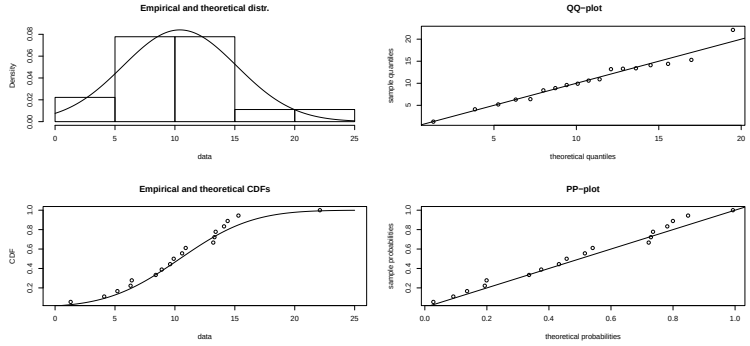

Рис. 2.16:

Рассмотрим ещё функцию **plotdistr()** пакета **fitdistrplus**. Её вид

```
plotdist(data,distr,para,breaks="default",discrete=FALSE,...)
```

Аргументы:

- **data** — числовая выборка.

- **distr** — символьный аргумент — **"name"** название распределения, для которого определены функции плотности, ФР и квантилей. Аргумент **distr** не задаётся, если не задаётся аргумент **para**.
- **para** — параметры распределения — список. Не задаётся, если не задан **distr**.
- **breaks** — аргумент, определяющий количество интервалов разбиения выборки при построении гистограммы. Определяется функцией **hist**.
- **discrete** — логический аргумент — если **discrete** = **TRUE**, распределение считается дискретным. Данный аргумент не играет роли, если задан аргумент **distr**.
- **...** — дополнительные графические аргументы.

Если аргументом функции **plotdistr()** является только выборка, то строятся два графика: гистограмма и график ЭФР. Если задано распределение и параметры распределения, то строятся гистограмма, QQ-график, PP-график и график ЭФР с ТФР.

Пример 39. *Рассмотрим работу функции* **plotdistr()**.

```
> x1 = c(6.4,13.3,4.1,1.3,14.1,10.6,9.9,9.6,15.3,22.1,13.4,
+ 13.2,8.4,6.3,8.9,5.2,10.9,14.4)
> plotdist(x1)
>plotdist(x1,col="red",type="b",pch=4)
>plotdist(x1,type="s")
```

Результат представлен на рис.2.17–2.19. Теперь рассмотрим случай, когда

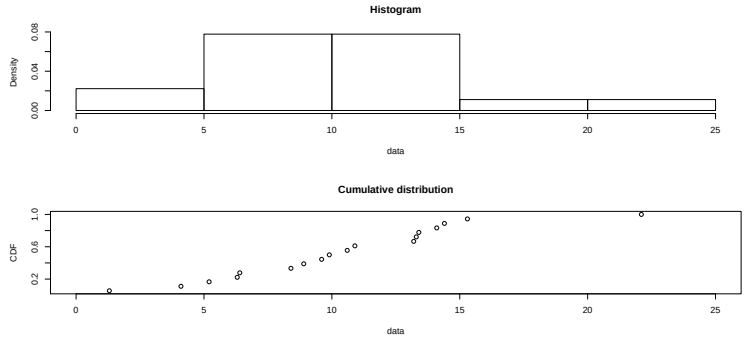

Рис. 2.17:

определено распределение — рис.2.20.

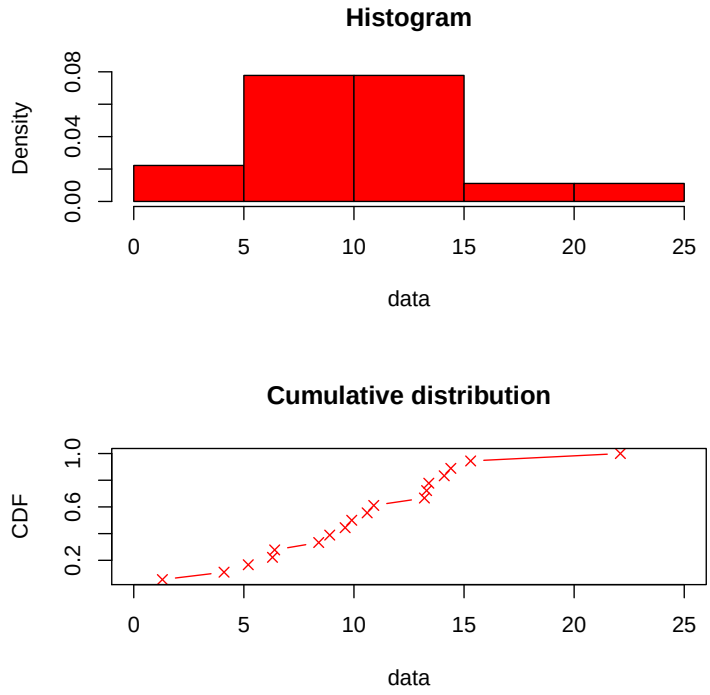

Рис. 2.18:

```
> xn<-rnorm(n=50,mean=5,sd=2)
> plotdist(xn,"norm",para=list(mean=mean(xn),sd=sd(xn)),pch=16,col="green")
```

Функция mmedist

Другая функция пакета **fitdistrplus**, реализующая метод моментов в **R**, это **mmedist**.

```
mmedist(data, distr)
```

Первый аргумент функции — числовая выборка, а второй — название распределения: **"norm"**, **"lnorm"**, **"exp"**, **"pois"** , **"gamma"**, **"logis"**, **"nbinom"**, **"geom"**, **"beta"** и **"unif"**.

Пример 40. *Найдём оценки неизвестных параметров ряда распределений.*

```
> x1=c(6.4,13.3,4.1,1.3,14.1,10.6,9.9,9.6,15.3,22.1,13.4,
+ 13.2,8.4,6.3,8.9,5.2,10.9,14.4)
> mmedist(x1,"norm")
```

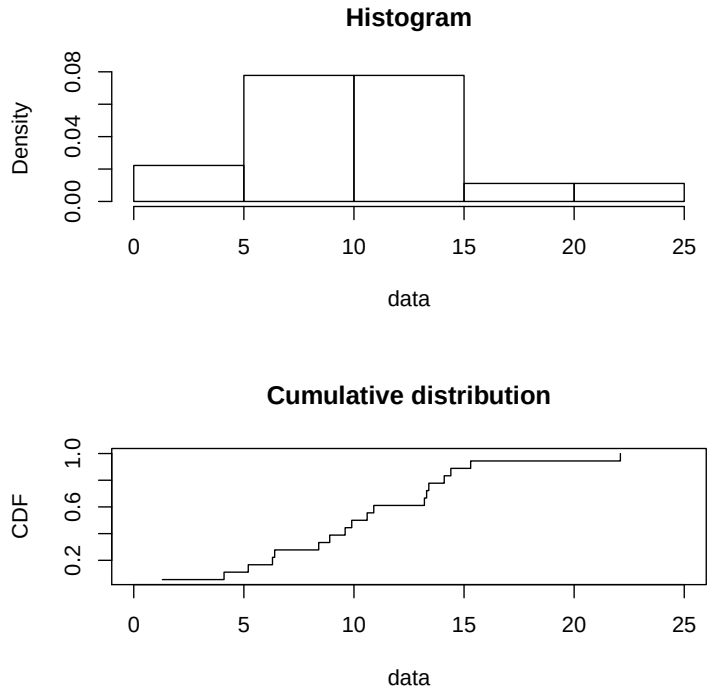

Рис. 2.19:

```
     mean        sd
10.411111  4.747033

> x2=c(rep(4,10),rep(2,30),rep(1,70),rep(0,120))
> mmedist(x2,"pois")
   lambda
0.7391304

> x3=c(0.80,0.72,0.88,0.84,0.38,0.64,0.69,0.48,0.73,0.58,0.81,
+ 0.83,0.71,0.75,0.59)
> mmedist(x3,"beta")
  shape1   shape2
7.322649 3.208486
```

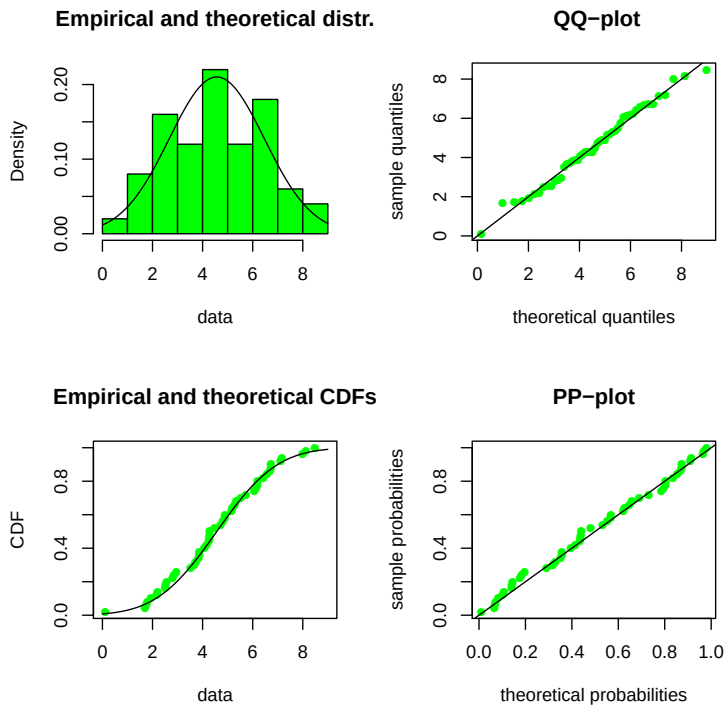

Рис. 2.20:

2.3.3 Метод максимального правдоподобия. Функции fitdist, mledist пакета fitdistrplus. Функция mle2 пакета bbmle. Функция mle пакета stats4. Функция maxLik пакета maxLik

В этом разделе будет разобрано несколько функций из различных пакетов, позволяющие вычислить оценки неизвестных параметров.

Функция fitdist пакета fitdistrplus

Снова обратимся к функции **fitdist**. Помимо метода моментов в ней реализован и метод максимального правдоподобия, более того, именно этот метод реализуется по умолчанию.

```
fitdist(data, distr, method="mle", start,...)
```

Рассмотрим только новый аргумент **start**. Это именованный[4] список, задающий начальные значения неизвестных параметров ТФР. Для распределений: **"norm"**, **"lnorm"**, **"exp"**, **"pois"**, **"cauchy"**, **"gamma"**, **"logis"**, **"nbinom"** (параметризация через **mu** и **size**), **"geom"**, **"beta"** и **"weibull"** можно не задавать значения **start** параметров (находятся автоматически), однако для улучшения результатов рекомендуется использовать начальные значения. Для равномерного распределения — **"unif"** — аргумент **start** нужно задавать обязательно.

Результат исполнения функции — оценки неизвестных параметров и стандартные ошибки (отклонения).

В качестве дополнительного аргумента можно ввести **optim.method**. Этот аргумент позволяет выбирать различные оптимизационные алгоритмы, реализованные в базовой функции **optim**.

Функция mledist

Функция **mledist** пакета **fitdistrplus** предназначена только для метода максимального правдоподобия.

```
mledist(data, distr, start, optim.method="default", lower=-Inf, upper=Inf, custom.o
```

Рассмотрим только последние пять аргументов:

- **optim.method** — символьный аргумент — выбор метода оптимизации базовой функции **optim**:
  - **"Nelder-Mead"** — базовый метод, использует только функцию. Медленный, но надёжный. Рекомендуется для недифференцируемых функций.
  - **"BFGS"** — используются функция и градиенты функции.
  - **"CG"** — метод сопряжённых градиентов.
  - **"L-BFGS-B"** — реализация метода **"BFGS"** с использованием ограничений.
  - **"SANN"** — стохастическая оптимизация, используется только оптимизируемая функция, можно использовать для недифференцируемых функций.
- **lower** и **upper** — нижняя и верхняя граница для метода **"L-BFGS-B"**.

[4] Т.е. названия компонент списка, которым присваиваются начальные значения параметров, должны **полностью** совпадать с названиями параметров распределений в **R**

- **custom.optim** — функция, отвечающая за оптимизацию (отличная от **optim**).
- **...** — дополнительные аргументы к **optim** или **custom.optim**.

Что касаемо аргумента **start**, то всё аналогично функции **fitdist**.
Результат вызова функции **mledist** — список из следующих компонент:

- **estimate** — оценка неизвестных параметров.
- **convergence** — сообщение о завершении оптимизации: **0** — успешно найдены оценки параметров, **10** — превышение лимита итераций, остальные коды — сообщения об ошибках.
- **loglik** — значение логарифма функции правдоподобия при найденной оценке.
- **hessian** — значения гессиана.
- **optim.function** — используемая оптимизационная функция.

Функция mle пакета stats4

Функция **mle** пакета **stats4** находит оценки неизвестных параметром ТФР методом максимального правдоподобия. Основное отличие от рассмотренных выше функций состоит в том, что надо в явном виде задать логарифм функции правдоподобия.

```
mle(minuslogl, start = formals(minuslogl), method = "BFGS",
    fixed = list(), ...)
```

Аргументы:

- **minuslogl** — функция — отрицательный логарифм функции правдоподобия ($-\log L(\Theta)$).
- **start** — именованный список — начальные значения оцениваемых параметров.
- **method** — используемый метод оптимизации.
- **fixed** — именованный список параметров, неизменных во время оптимизации.
- **...** — дополнительные аргументов, связанные с **optim**

Результатом работы функции является объект класса **mle-class** с найденными оценками неизвестных параметров. Чтобы получить числовой вектор с оценками параметров необходимо использовать **coef(mle())**

Функция **summary()** выводит суммарную информацию: оценки параметров и значение $-2\log L$ логарифма функции правдоподобия, умноженное на -2.

Функция **logLik()** выводит значение логарифма функции правдоподобия при найденных значениях неизвестных параметров.

Функция **confint()** вычисляет 95% доверительные интервалы для оцениваемых параметров.

Функция mle2 пакета bbmle

Функция **mle2** пакета **bbmle** представляет собой аналог функции **mle** пакета **stats4** с большим количеством аргументов.

```
mle2(minuslogl, start, method, optimizer,
    fixed = NULL, data=NULL,
    subset=NULL,
default.start=TRUE, eval.only = FALSE, vecpar=FALSE,
parameters=NULL,
parnames=NULL,
skip.hessian=FALSE,trace=FALSE,
transform=NULL,
gr,...)
```

Единственный важный аргумент функции — это **minuslogl** — минус логарифм функции правдоподобия.

Результат — объект класса **"mle2"**. Для того, чтобы получить вектор оценок неизвестных параметров ТФР, следует воспользоваться **coef(mle2())**.

Функция maxLik пакета maxLik

Функция

```
maxLik(logLik, grad = NULL, hess = NULL, start, method,
constraints=NULL, ...)
```

пакета **maxLik** предназначена для нахождения оценок неизвестных параметров ТФР методом максимального правдоподобия.

Первый основной аргумент — **logLik** — логарифм функции правдоподобия — функция, первым аргументом которой является оцениваемый параметр или вектор оцениваемых параметров. Вспомогательные аргументы, необязательные для задания, но ускоряющие работу — это

- **grad** — вектор или матрица градиентов логарифма функции правдоподобия.

- **hess** — гессиан (матрица Гессе) логарифма функции правдоподобия. Это матрица вторых частных производных:

$$H(f) = \begin{pmatrix} \frac{\partial f}{\partial x_1^2} & \frac{\partial f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial f}{\partial x_1 \partial x_n} \\ \frac{\partial f}{\partial x_2 \partial x_1} & \frac{\partial f}{\partial x_2^2} & \cdots & \frac{\partial f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial x_n \partial x_1} & \frac{\partial f}{\partial x_n \partial x_2} & \cdots & \frac{\partial f}{\partial x_n^2} \end{pmatrix}$$

- **start** — числовой вектор — второй обязательный аргумент — начальные значения оцениваемых параметров.

- **method** — символьный аргумент — название метода поиска оценки максимального правдоподобия. Возможные значения: **"Newton-Raphson"**, **"BFGS"**, **"BFGS-YC"**, **"BHHH"**, **"SANN"** или **"NM"** (Nelder-Mead). Возможны сокращения названий (если нет двоякого толкования) и запись названий методов прописными буквами. По умолчанию выбирается оптимальный метод.

- **constraints** — либо **NULL**, либо список с указанием ограничений для условной (ограниченной) оптимизации.

- **...** — дополнительные аргументы

Чтобы получить вектор оценок параметров необходимо воспользоваться функцией **coef()** (см. пример).

Примеры

Рассмотрим на примерах работу функций, реализующих метод максимального правдоподобия.

Пример 41. *Сначала рассмотрим работу функции* **fitdist** *на примере нормального закона.*

```
> library('fitdistrplus')
> x1 = c(6.4,13.3,4.1,1.3,14.1,10.6,9.9,9.6,15.3,22.1,13.4,
+ 13.2,8.4,6.3,8.9,5.2,10.9,14.4)
> fitdist(x1,"norm")
Fitting of the distribution ' norm ' by maximum likelihood
```

```
Parameters:
      estimate Std. Error
mean 10.411111   1.118886
sd    4.747033   0.791172
```

Затем зададим собственное распределение (распределение Гумбеля):

```
> #Распределение Гумбеля
> dgumbel <- function(x,a,b) 1/b*exp((a-x)/b)*exp(-exp((a-x)/b))
> pgumbel <- function(q,a,b) exp(-exp((a-q)/b))
> qgumbel <- function(p,a,b) a-b*log(-log(p))
> #Задаются начальные значения
> fitdist(x1,"gumbel",start=list(a=10,b=5))
Fitting of the distribution ' gumbel ' by maximum likelihood
Parameters:
  estimate Std. Error
a 8.094333  1.0924286
b 4.375401  0.7659003
```

Теперь найдём оценку параметра λ экспоненциального распределения, не используя и используя начальные значения оцениваемого параметра.

```
> x2=rexp(200,0.5)
> fitdist(x2,"exp", start=list(rate=1))
Fitting of the distribution ' exp ' by maximum likelihood
Parameters:
     estimate Std. Error
rate 0.477256 0.03374695
Предупреждения
1: In dexp(x, 1/rate, log) : созданы NaN
2: In dexp(x, 1/rate, log) : созданы NaN
3: In dexp(x, 1/rate, log) : созданы NaN
4: In dexp(x, 1/rate, log) : созданы NaN
> fitdist(x2,"exp")
Fitting of the distribution ' exp ' by maximum likelihood
Parameters:
      estimate Std. Error
rate 0.4772326 0.03374529
```

Как видно из примера, задание начальных значений может приводить к предупреждениям и не всегда лучшим результатам.

Нахождение ММП оценок при помощи функции **mledist**.

```
> library('fitdistrplus')
> x1 = c(6.4,13.3,4.1,1.3,14.1,10.6,9.9,9.6,15.3,22.1,13.4,
+ 13.2,8.4,6.3,8.9,5.2,10.9,14.4)
> mledist(x1,"norm")
$estimate
     mean        sd
10.411111  4.747033
$convergence
[1] 0
$loglik
[1] -53.57625
$hessian
          mean       sd
mean 0.7987816 0.000000
sd   0.0000000 1.597564
$optim.function
[1] "optim"

> #Распределение Гумбеля
> dgumbel <- function(x,a,b) 1/b*exp((a-x)/b)*exp(-exp((a-x)/b))
> pgumbel <- function(q,a,b) exp(-exp((a-q)/b))
> qgumbel <- function(p,a,b) a-b*log(-log(p))
> #Задаются начальные значения
> mledist(x1,"gumbel",start=list(a=10,b=5))
$estimate
       a        b
8.094333 4.375401
$convergence
[1] 0
$loglik
[1] -54.09525
$hessian
           a          b
a  0.9400408 -0.4418806
b -0.4418806  1.9124424
$optim.function
[1] "optim"

> x2=rexp(200,0.5)
> mledist(x2,"exp", start=list(rate=1))
```

```
$estimate
     rate
0.5758357
$convergence
[1] 0
$loglik
[1] -310.3868
$hessian
        rate
rate 603.164
$optim.function
[1] "optim"
Предупреждения
1: In dexp(x, 1/rate, log) : созданы NaN
2: In dexp(x, 1/rate, log) : созданы NaN
3: In dexp(x, 1/rate, log) : созданы NaN
4: In dexp(x, 1/rate, log) : созданы NaN
5: In dexp(x, 1/rate, log) : созданы NaN
6: In dexp(x, 1/rate, log) : созданы NaN
7: In dexp(x, 1/rate, log) : созданы NaN

> mledist(x2,"exp")
$estimate
    rate
0.575835
$convergence
[1] 0
$loglik
[1] -310.3868
$hessian
         rate
rate 603.1655
$optim.function
[1] "optim"
```

Найдём оценки неизвестных параметров m *и* σ^2 *нормального закона при помощи функции* **mle** *пакета* **stats4**.

```
> mu=20; sigma2=4 #дисперсия
> x=rnorm(100,mu,sqrt(sigma2))
> #Логарифм функции правдоподобия
> Log.L=function(mu.hat=15,sigma2.hat=6)
```

```
+ {n=length(x)
+ n/2*log(2*pi*sigma2.hat)+1/2*sum((x-mu.hat)^2/sigma2.hat)
+ }
> library('stats4')
> mle(Log.L)

Call:
mle(minuslogl = Log.L)

Coefficients:
    mu.hat sigma2.hat
 20.067841   4.678276
Предупреждение
In log(2 * pi * sigma2.hat) : созданы NaN
```

Начальные значения оцениваемых параметров были указаны при задании логарифма функции правдоподобия. Дополнительные функции:

```
> confint(fit1)
Profiling...
               2.5 %    97.5 %
mu.hat     19.639800 20.495884
sigma2.hat  3.589962  6.256223
> summary(fit1)
Maximum likelihood estimation

Call:
mle(minuslogl = Log.L)

Coefficients:
            Estimate Std. Error
mu.hat     20.067841  0.2162932
sigma2.hat  4.678276  0.6616081

-2 log L: 438.0807
> logLik(fit1)
'log Lik.' -219.0403 (df=2)
```

Получим вектор оценок:

```
> fit1=mle(Log.L)
Предупреждение
```

```
In log(2 * pi * sigma2.hat) : созданы NaN
> coef(fit1)
    mu.hat sigma2.hat
 20.067841   4.678276
```

Для примера рассмотренного выше (нормальная выборка с неизвестными средним и дисперсией) получим оценки ММП при помощи функции **mle2** *пакета* **bbmle**.

```
> mu=20;sigma2=4 #дисперсия
> x=rnorm(100,mu,sqrt(sigma2))
> #Логарифм функции правдоподобия
> Log.L=function(mu.hat=15,sigma2.hat=6)
+ {n=length(x)
+ n/2*log(2*pi*sigma2.hat)+1/2*sum((x-mu.hat)^2/sigma2.hat)
+ }
> library('bbmle')
> fit2=mle2(Log.L)
Предупреждение
In log(2 * pi * sigma2.hat) : созданы NaN
> fit2

Call:
mle2(minuslogl = Log.L)

Coefficients:
    mu.hat sigma2.hat
 20.221350   4.533344

Log-likelihood: -217.47
> coef(fit2)
    mu.hat sigma2.hat
 20.221350   4.533344
```

Снова для нормальной выборки (см. выше) найдём ОМП при помощи функции **maxLik** *пакета* **maxLik**.

Сначала найдём логарифм функции правдоподобия

```
> Log.L=function(param)
+ {mu.hat=param[1]
+ sigma2.hat=param[2]
+ n=length(x)
```

```
+ -n/2*log(2*pi*sigma2.hat)-1/2*sum((x-mu.hat)^2/sigma2.hat)
+ }
```

Затем оценки неизвестных параметров

```
> library('maxLik')
> fit3=maxLik(Log.L,start=c(15,6))
Предупреждения
1: In log(2 * pi * sigma2.hat) : созданы NaN
2: In log(2 * pi * sigma2.hat) : созданы NaN
3: In log(2 * pi * sigma2.hat) : созданы NaN
> fit3
Maximum Likelihood estimation
Newton-Raphson maximisation, 9 iterations
Return code 1: gradient close to zero
Log-Likelihood: -225.4695 (2 free parameter(s))
Estimate(s): 19.88166 5.32021
> coef(fit3)
[1] 19.88166  5.32021
```

2.4 Доверительные интервалы в R

2.4.1 Доверительные интервалы для выборок большого объёма

2.4.2 Доверительные интервалы для выборок небольшого объёма

Глава 3

Проверка статистических гипотез в R для одной выборки

3.1 Проверка гипотезы о нормальности выборки

3.1.1 Тест Шапиро–Уилка.

Функция **shapiro.test(x)** выполняет тест Шапиро–Уилка. Нуль-гипотеза заключается в том, что случайная величина, выборка x которой известна, распределена по нормальному закону. Объем выборки должен быть не меньше 3 и не больше 5000.

Объект, возвращаемый функцией **shapiro.test**, — это список со следующими полями:

- **statistics** — значение статистики Шапиро–Уилка,
- **p.value** — p-value,
- **method** — строка "Shapiro-Wilk normality test
- **data.name** — строка, содержащее имя данных, подвергнутых тесту.

Рассмотрим несколько примеров.

Пример 42. *Протестируем стандартные датчики распределений. Начнём с нормальноrо распределения.*

```
> set.seed(0)
> shapiro.test(rnorm(100, mean = 2, sd = 5))
Shapiro-Wilk normality test
```

```
data: rnorm(100, mean = 2, sd = 5)
W = 0.9896, p-value = 0.6303
```

При уровне значимости, например, $\alpha = 0.05$ *гипотеза должна быть принята, так как* **p-value** $> \alpha$.

Теперь рассмотрим равномерное распределение.

```
> set.seed(0)
> shapiro.test(runif(100, min = -10, max = 10))
Shapiro-Wilk normality test
data: runif(100, min = -10, max = 10)
W = 0.9561, p-value = 0.002126
```

При уровне значимости, например, $\alpha = 0.05$ *гипотеза должна быть отвергнута, так как* **p-value** $< \alpha$.

Рассмотрим еще один пример.

Пример 43. *Таблица данных* **trees** *из библиотеки* **datasets** *содержит замеры диаметра, высоты и объёма вишнёвых деревьев. Проверим гипотезу о том, что высоты деревьев распределены по нормальному закону.*

```
> colnames(trees)
[1] "Girth" "Height" "Volume"
> x <- trees[, "Height"]
hist(x, col = "green", xlab = "Tree heights",
main = "Tree height frequencies")
> shapiro.test(x)
Shapiro-Wilk normality test
data: x
W = 0.9655, p-value = 0.4034
```

При уровне значимости, например, $\alpha == 0.05$ *гипотезу о нормальности распределения принимаем.*

3.2 Критерии согласия

3.2.1 Критерий Колмогорова–Смирнова

Общий вид:

```
ks.test(x, y, ...,alternative = c("two.sided", "less", "greater"),
exact = NULL)
```

Тест Колмогорова–Смирнова для одной или двух выборок. Аргументы:

- **x** — вектор, содержащий выборку.
- **y** — вектор, содержащий вторую выборку, или символьная строка с именем распределения.
- **...** — параметры распределения.
- **alternative** — символьный аргумент, обозначающих тип альтернативной гипотезы. Принимает одно из следующих значений: **"two.sided"** (по умолчанию), **"less"** или **"greater"**.
- **exact** — **NULL** или логическое значение, обозначающее требуется ли точное вычисление **p-value**. Не используется в двувыборочном тесте, если **alternative** ="**less**" или **alternative** = "**greater**".

Детали:

- Если **y** — числовой вектор, то выполняется двувыборочный тест Колмогорова–Смирнова, проверяющий нуль-гипотезу о том, что **x** и **y** принадлежат одному и тому же непрерывному распределению.
- Если **y** — символьная переменная (имя непрерывного распределения), то выполняется одновыборочный тест Колмогорова–Смирнова, проверяющий нулевую гипотезу о том, что **x** принадлежит заданному распределению.
- Возможные значения **"two.sided"**, **"less"** и **"greater"** параметра **alternative** определяют альтернативную гипотезу, заключающуюся в том, что эмпирическая функция распределения выборки **x** не совпадает с теоретической (**"two.sided"**), не больше ее (**"less"**) или не меньше ее (**"greater"**).

Точное значение **p-value** не вычисляется в двувыборочном тесте, если **alternative** = **"less"** или **alternative** = **"greater"**. В одновыборочном тесте параметры гипотетического распределения должны быть известны точно, а не вычисляться по выборке **x**. Вариант теста Колмогорова–Смирнова с оценкой параметров не поддерживается.

Объект, возвращаемый функцией **ks.test()**, — это список со следующими полями:

- **statistics** — значение статистики Колмогорова—Смирнова;
- **p.value** — ;

- **alternative** — символьный аргумент — описание альтернативной гипотезы;
- **method** — символьная аргумент — название используемого метода;
- **data.name** — название массива данных, подвергнутых тесту.

Рассмотрим пример.

Пример 44. *Фрейм данных* **randu** *из библиотеки* **datasets** *содержит 400 троек псевдо-случайных чисел из интервала* $[0; 1]$. *Значения записаны в матрицу с тремя столбцами, называемыми именами* x, y, z.

```
> colnames(randu)
[1] "x" "y" "z"
> nrow(randu)
[1] 400
> attach(randu)
> detach(randu)
> ks.test(x, y)
Two-sample Kolmogorov-Smirnov test
data: x and y
D = 0.085, p-value = 0.1111
alternative hypothesis: two-sided
Warning message:
cannot compute correct p-values with ties in: ks.test(x, y)

> ks.test(x, z)
Two-sample Kolmogorov-Smirnov test
data: x and z
D = 0.0875, p-value = 0.09353
alternative hypothesis: two-sided

> ks.test(y, z)
Two-sample Kolmogorov-Smirnov test
data: y and z
D = 0.0475, p-value = 0.7576
alternative hypothesis: two-sided

> ks.test(x, punif)
One-sample Kolmogorov-Smirnov test
data: x
```

```
D = 0.0555, p-value = 0.1697
alternative hypothesis: two-sided

> ks.test(y, punif)
One-sample Kolmogorov-Smirnov test
data: y
D = 0.0357, p-value = 0.6876
alternative hypothesis: two-sided

> ks.test(z, punif)
One-sample Kolmogorov-Smirnov test
data: z
D = 0.0455, p-value = 0.3782
alternative hypothesis: two-sided
> detach(randu)
```

3.2.2 Критерий согласия χ^2 Пирсона

Описание:

```
chisq.test(x, y = NULL, correct = TRUE,
p = rep(1/length(x), length(x)), rescale.p = FALSE,
simulate.p.value = FALSE, B = 2000)
```

Функция реализует критерий согласия χ^2 Пирсона для простых гипотез и тест на проверку независимости признаков.

Аргументы функции:

- **x** — вектор или матрица.
- **y** — вектор. Игнорируется, если **x** — матрица.
- **correct** — логическое значение, указывающее, требуется ли применять непрерывную коррекцию для 2×2 матриц.
- **p** — вектор, содержащий вероятности. Должен иметь такую же длину, что и **x**.
- **rescale.p** — логическое значение. Если **TRUE**, то **p** при необходимости нормируется так, чтобы сумма его компонентов была равна 1.
- **simulate.p.value** — логическое значение. Если **TRUE**, то **p-value** вычисляется с помощью метода Монте-Карло, в противном случае используется χ^2 -распределение

- **B** — количество испытаний в методе Монте-Карло.

Детали:

- Если **x** — вектор или матрица с одним столбцом (или одной строкой), а вектор **y** не задан, то **x** рассматривается как статистический ряд (одномерная таблица сопряжённости признаков). Т. е. i-я компонента вектора **x** содержит количество элементов выборки, попавших в i-й интервал группировки. В этом случае выполняется тест на проверку соответствия (согласия) выборки заданным вероятностям **p**. Таким образом, основная (нулевая) гипотеза заключается в том, что вероятность попадания в i-й интервал группировки равна i-й компоненте вектора **p**. По умолчанию, задаются равные вероятности.

- Если **x** — матрица не менее чем с 2 строками и 2 столбцами, то x рассматривается как двумерная таблица сопряжённости признаков и выполняется тест на проверку их независимости

Таким образом, обратим внимание, что если аргумент **y** не задан, то функция **chisq.test** работает со статистическим рядом или таблицей сопряжённости, а не непосредственно с самой выборкой.

- Если **x** и **y** числовые векторы или факторы одной и той же длины (числовые векторы будут преобразованы в факторы), то соответствующие пары их компонентов рассматриваются как реализации двумерной случайной величины $(X; Y)$ и выполняется тест на проверку независимости признаков X и Y .

Результат работы функции. Функция **chisq.test** возвращает список, состоящий из следующих полей:

- **statistics** — значение χ^2 -статистики Пирсона

- **parameter** — число степеней свободы распределения χ^2; равно **NA**, если для отыскания **p-value** использовался метод Монте-Карло,

- **p-value** —

- **method** — символьная строка с названием используемой модификации теста, а также с указанием того, использовались ли непрерывная коррекция и метод Монте-Карло,

- **data.name** — строка, содержащее имя (имена) данных, подвергнутых тесту.

- **observed** — число точек, попавших в i-й интервал группировки.
- **expected** — теоретическое число точек (в предположении выполнения гипотезы), попадающих в i-й интервал группировки.
- **residuals** — остатки Пирсона:

$$\frac{observed - expected}{\sqrt{expected}}$$

Пример 45. *Рассмотрим классический пример с бросанием монеты. Бюффон бросал монету 4040 раз, при этом герб выпал 2048 раз.*

Используя критерий согласия χ^2, проверим, что монета симметрична. Итак, основная гипотеза заключается в том, что вероятность выпадания герба равна $p_1 = 1/2$, вероятность выпадания решки — $p_2 = 1/2$.

```
chisq.test(c(2048, 1992))
```

Результат:

```
Chi-squared test for given probabilities
data: c(2048, 1992)
X-squared = 0.7762, df = 1, p-value = 0.3783
```

Пусть, например, был выбран уровень значимости $\alpha = 0.05$. Так как $\alpha <$ **p-value***, то гипотезу принимаем.*

Пример 46. *Рассмотрим ещё один простор пример проверки независимости двух генеральных совокупностей, если известны выборки* $\mathbf{x}$ *и* $\mathbf{y}$*. Соответствующие пары компонентов векторов будем рассматривать как реализации двумерной случайной величины $(X; Y)$.*

```
set.seed(0)
x <- rnorm(100)
y <- runif(100)
```

Проверяем на независимость:

```
chisq.test(x, y)
```

Выводимые результаты:

```
        Pearson's Chi-squared test

data:  x and y
X-squared = 9900, df = 9801, p-value = 0.239

Предупреждение
In chisq.test(x, y) :
  аппроксимация на основе хи-квадрат может быть неправильной
```

Пусть, например, был выбран уровень значимости $\alpha = 0.05$. *Так как* $\alpha <$ **p-value**, *то гипотезу о независимости случайных признаков можно принять.*

И ещё один пример.

Пример 47. *Таблица* **HairEyeColor** *из библиотеки* **datasets** *содержит информацию о поле, цвете волос и глаз у 592 студентов. Таблица имеет 3 размерности:*

```
"Hair": HairEyeColor["Black", ,], HairEyeColor["Brown", ,],
HairEyeColor["Red", ,], HairEyeColor["Blond", ,],
"Eye": HairEyeColor[, "Brown",], HairEyeColor[, "Blue",],
HairEyeColor[, "Hazel",], HairEyeColor[, "Green",],
"Sex": HairEyeColor[, , "Male"], HairEyeColor[, , "Female"].
```

Элементы таблицы — количество человек из данной группы. Проверим гипотезу о том, что для мужчин цвет глаз не зависит от цвета волос мужчин.

Сначала построим таблицу сопряжённости признаков

```
men <- HairEyeColor[, , "Male"]
```

и выведем её

```
> men
       Eye
Hair    Brown Blue Hazel Green
  Black    32   11    10     3
  Brown    53   50    25    15
  Red      10   10     7     7
  Blond     3   30     5     8
```

Построим мозаичную диаграмму,

```
mosaicplot(men,col = c("chocolate", "cornflowerblue", "salmon", "green"))
```

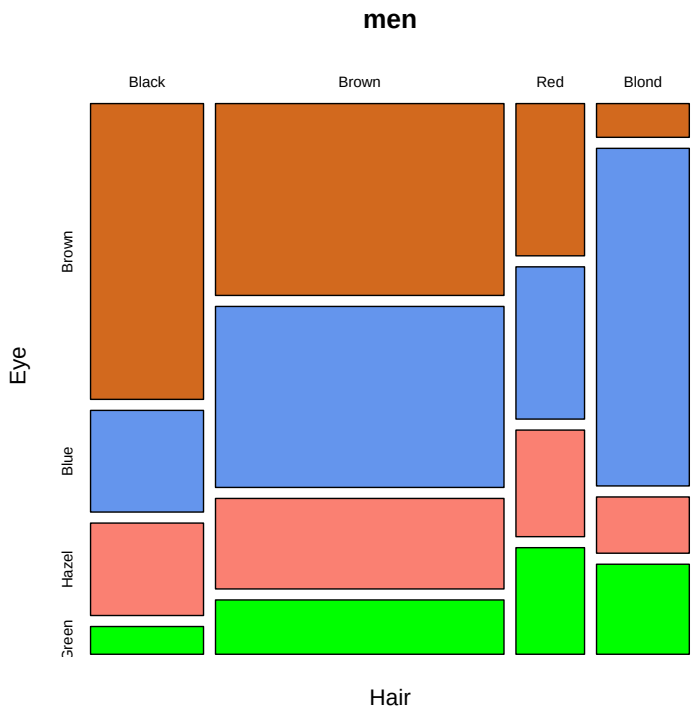

Рис. 3.1: Диаграмма цвета волос и глаз у мужчин

представленную ниже на рис.4.1

Проверим теперь на независимость.

```
chisq.test(men, simulate.p.value = TRUE)
```

Результат:

```
Pearson's Chi-squared test with simulated p-value (based on 2000
        replicates)

data:  men
X-squared = 41.2803, df = NA, p-value = 0.0004998
```

При уровне значимости, например, $\alpha = 0.05$*, гипотезу следует отклонить и признаки считать зависимыми.*

Проведём аналогичное исследование для женщин. Строим таблицу сопряжённости признаков.

```
> female <- HairEyeColor[, , "Female"]
> female
       Eye
Hair    Brown Blue Hazel Green
  Black    36    9     5     2
  Brown    66   34    29    14
  Red      16    7     7     7
  Blond     4   64     5     8
```

Мозаичная диаграмма,

```
mosaicplot(female,col = c("chocolate", "cornflowerblue", "salmon", "green"))
```

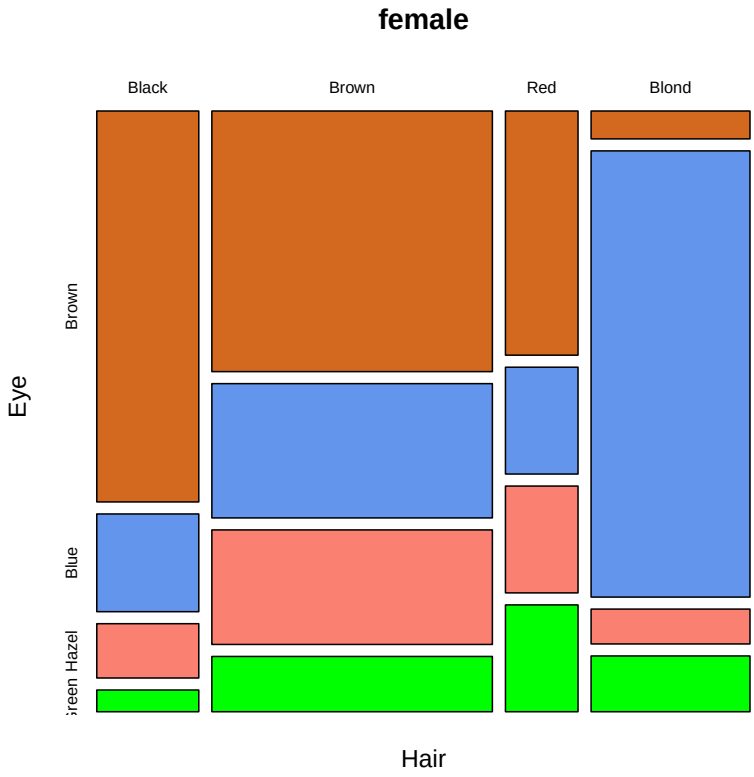

Рис. 3.2: Диаграмма цвета волос и глаз у женщин

Проверка на независимость

```
chisq.test(female, simulate.p.value = TRUE)
```

и её результат

```
        Pearson's Chi-squared test with simulated p-value (based on 2000
        replicates)

data:  female
X-squared = 106.6637, df = NA, p-value = 0.0004998
```

При уровне значимости $\alpha = 0.05$, гипотезу о независимости следует отклонить и признаки (цвет волос и глаз) считать зависимыми.

3.3 t-тест Стьюдента

Вид функции:

```
t.test(x, ...)
```

```
t.test(x, y = NULL, alternative = c("two.sided", "less", "greater"),
mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.95, ...)
```

```
t.test(formula, data, subset, na.action, ...)
```

Выполняется одно или двувыборочный тест Стьюдента.

Одновыборочный t-тест предназначен для проверки равенства среднего значения выборки из нормально распределённой генеральной совокупности некоторому заданному значению в предположении, что дисперсия не известна.

Двувыборочный тест служит для сравнения двух средних значений выборок из нормально распределённых генеральных совокупностей в предположении, что их дисперсии равны, хотя и не известны.

Аргументы:

- **x** — числовой вектор, содержащий элементы первой выборки.
- **y** — числовой вектор, содержащий элементы второй выборки.
- **alternative** — символьный аргумент, определяющий тип альтернативной гипотезы. Возможные значения: **"two.sided"** — средние значения не равны (по умолчанию), **"less"** или **"greater"**.
- **exact** — либо **NULL**, либо логический аргумент. Отвечает за точное вычисление **p-value**. Не используется в двувыборочном тесте, если **alternative** = **"less"** или **alternative** = **"greater"**.

- **mu** — математическое ожидание или разность математических ожиданий, если задано две выборки.
- **paired** — логическое значение, указывающее, требуется ли выполнить спаренный t-тест.
- **var.equal** — логическое значение, считающее, считать ли разбросы равными. Если **TRUE**, то вычисляется разброс для объединённой выборки.
- **conf.level** — доверительный уровень.
- **formula** — формула вида **lhs**$\sim$ **rhs**, где lhs — числовой вектор, а rhs — фактор с двумя классами. Только для двувыборочного теста.
- **data** — матрица или фрейм данных, из которых берутся данные.
- **subset** — вектор, определяющий используемое подмножество наблюдений.
- **na.action** — функция, которая вызывается, как только в данных встретилось значение NA.

Детали:

- Если **y** и **formula** не заданы, то выполняется одновыборочный тест, проверяющий, что выборка **x** имеет среднее, равное 0.
- Если **paired** = **TRUE**, то должны быть определены и иметь одинаковую длину векторы **x** и **y**.
- Значения **NA** и **NaN** из данных удаляются (если **paired** = **TRUE**, то при этом удаляется соответствующее значение из второй выборки).
- Если **var.equal** = **TRUE**, то для оценки отклонения используется объединённая выборка. По умолчанию **var.equal** = **FALSE** и отклонение оценивается отдельно для каждой выборки. При этом происходит надлежащая корректировка числа степеней свободы.

Возвращаемое значение. Объект, возвращаемый функцией **t.test**, — список со следующими полями:

- **statistics** — значение t-статистики Стьюдента.
- **parameter** — число степеней свободы.
- **p.value** —

- **conf.int** — доверительный интервал для математического ожидания.
- **estimate** — оценка математического ожидания для одновыборочного теста или разности математических ожиданий для двувыборочного теста.
- **null.value** — предполагаемое математическое ожидание или разность пред- полагаемых математических ожиданий для двувыборочного теста (входной параметр **mu**).
- **alternative** — символьная строка с описанием альтернативной гипотезы.
- **method** — символьная строка с названием используемой модификации метода.
- **data.name** — строка, содержащее имя (имена) данных, подвергнутых тесту.

Пример 48. *В качестве простых примеров сравним математические ожидания у двух выборок, полученных с помощью функции* **rnorm***:*

```
> set.seed(0)
> x <- rnorm(100, mean = 0, sd = 4)
> y <- rnorm(100, mean = 1, sd = 4)
```

Итак, **x** *и* **y** *— две выборки из нормальных генеральных совокупностей с одинаковыми среднеквадратичными отклонениями, но с разными средними. Применим к выборкам t-тест, полагая в качестве нуль-гипотезы, что генеральные совокупности имеют одинаковое математическое ожидание. Альтернативная гипотеза — средние значения совокупностей не равны.*

```
> t.test(x, y)
Welch Two Sample t-test
data: x and y
t = -1.3896, df = 196.428, p-value = 0.1662
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-1.7590435 0.3048036
sample estimates:
mean of x mean of y
0.0906738 0.8177938
```

Дадим разъяснение полученным результатам. Найдено значение t-статистики, число степеней свободы **df**, *величина* **p-value**. *Указаны границы 95% доверительного интервала для разности мат. ожиданий распределений первой и второй выборки. Приведены оценки математических ожиданий*

для каждого распре- деления. Пусть уровень значимости равен $\alpha = 0.1$. *Так как* **p-value** $> \alpha$, *то гипотезу о том, что математические ожидания у распределений равны, принимаем.*

Изменим теперь альтернативную гипотезу. Пусть альтернативная гипотеза постулирует, что вторая генеральная совокупность имеет большее математическое ожидание, чем первая:

```
> t.test(x, y, alternative = "less")
Welch Two Sample t-test
data: x and y
t = -1.3896, df = 196.428, p-value = 0.08311
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
-Inf 0.1376404
sample estimates:
mean of x mean of y
0.0906738 0.8177938
```

Теперь при уровне значимости $\alpha = 0.1$, *так как* **p-value** $< \alpha$, *то нуль-гипотезу отклоняем и принимаем альтернативную гипотезу.*

Пример 49. *В качестве еще одного примера рассмотрим данные об измерениях скорости света, полученные А.А. Майкельсоном и Э.У. Морли во время знаменитого эксперимента 1887 г.*

Данные содержатся во фрейме **morley** *библиотеки* **datasets**. *Фрейм содержит три столбца:* **Expt** *— номер эксперимента (от 1 до 5),* **Run** *— номер испытания (каждый эксперимент состоял из 20 испытаний),* **Speed** *— скорость света минус 299000 (в км/с). Примем во внимание только последний столбец.*

Предположим, что генеральная совокупность (замеры скорости света) имеет нормальное распределение. Сформулируем нуль-гипотезу: математическое ожидание генеральной совокупности равно 299792.458 *(принятое в настоящее время значение скорости света).*

```
> t.test(light - 792.458)
One Sample t-test
data: light - 792.458
t = 7.5866, df = 99, p-value = 1.824e-11
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
44.26459 75.61941
sample estimates:
```

```
mean of x
59.942
```

Дадим разъяснение полученным результатам. Найдено значение статистики, число степеней свободы **df**, *величина* **p-value**. *Указаны границы 95% доверительного интервала для оценки мат. ожидания выборки* **light** —-792.458. *Приведены оценки математических ожиданий для каждой группы. Пусть уровень значимости равен* $\alpha = 0.05$. *Так как* **p-value** $< \alpha$, *гипотезу отклоняем.*

Глава 4

Проверка статистических гипотез в R для двух и более выборок

4.1 Критерий Колмогорова–Смирнова

Общий вид:

```
ks.test(x, y, ...,alternative = c("two.sided", "less", "greater"),
exact = NULL)
```

Тест Колмогорова–Смирнова для одной или двух выборок. Аргументы:

- **x** — вектор, содержащий выборку.
- **y** — вектор, содержащий вторую выборку, или символьная строка с именем распределения.
- **...** — параметры распределения.
- **alternative** — символьный аргумент, обозначающих тип альтернативной гипотезы. Принимает одно из следующих значений: **"two.sided"** (по умолчанию), **"less"** или **"greater"**.
- **exact** — **NULL** или логическое значение, обозначающее требуется ли точное вычисление **p-value**. Не используется в двувыборочном тесте, если **alternative** ="less" или **alternative** = "greater".

Детали:

- Если **y** — числовой вектор, то выполняется двувыборочный тест Колмогорова–Смирнова, проверяющий нуль-гипотезу о том, что **x** и **y** принадлежат одному и тому же непрерывному распределению.

- Если **y** — символьная переменная (имя непрерывного распределения), то выполняется одновыборочный тест Колмогорова–Смирнова, проверяющий нулевую гипотезу о том, что **x** принадлежит заданному распределению.

- Возможные значения **"two.sided"**, **"less"** и **"greater"** параметра **alternative** определяют альтернативную гипотезу, заключающуюся в том, что эмпирическая функция распределения выборки **x** не совпадает с теоретической (**"two.sided"**), не больше ее (**"less"**) или не меньше ее (**"greater"**).

Точное значение **p-value** не вычисляется в двувыборочном тесте, если **alternative** = **"less"** или **alternative** = **"greater"**. В одновыборочном тесте параметры гипотетического распределения должны быть известны точно, а не вычисляться по выборке **x**. Вариант теста Колмогорова–Смирнова с оценкой параметров не поддерживается.

Объект, возвращаемый функцией **ks.test()**, — это список со следующими полями:

- **statistics** — значение статистики Колмогорова—Смирнова;

- **p.value** — ;

- **alternative** — символьный аргумент — описание альтернативной гипотезы;

- **method** — символьная аргумент — название используемого метода;

- **data.name** — название массива данных, подвергнутых тесту.

Рассмотрим пример.

Пример 50. *Фрейм данных* **randu** *из библиотеки* **datasets** *содержит 400 троек псевдо-случайных чисел из интервала* $[0; 1]$. *Значения записаны в матрицу с тремя столбцами, называемыми именами* x, y, z.

```
> colnames(randu)
[1] "x" "y" "z"
> nrow(randu)
[1] 400
> attach(randu)
> detach(randu)
> ks.test(x, y)
Two-sample Kolmogorov-Smirnov test
```

```
data: x and y
D = 0.085, p-value = 0.1111
alternative hypothesis: two-sided
Warning message:
cannot compute correct p-values with ties in: ks.test(x, y)

> ks.test(x, z)
Two-sample Kolmogorov-Smirnov test
data: x and z
D = 0.0875, p-value = 0.09353
alternative hypothesis: two-sided

> ks.test(y, z)
Two-sample Kolmogorov-Smirnov test
data: y and z
D = 0.0475, p-value = 0.7576
alternative hypothesis: two-sided

> ks.test(x, punif)
One-sample Kolmogorov-Smirnov test
data: x
D = 0.0555, p-value = 0.1697
alternative hypothesis: two-sided

> ks.test(y, punif)
One-sample Kolmogorov-Smirnov test
data: y
D = 0.0357, p-value = 0.6876
alternative hypothesis: two-sided

> ks.test(z, punif)
One-sample Kolmogorov-Smirnov test
data: z
D = 0.0455, p-value = 0.3782
alternative hypothesis: two-sided
> detach(randu)
```

4.2 t-тест Стьюдента

Вид функции:

```
t.test(x, ...)

t.test(x, y = NULL, alternative = c("two.sided", "less", "greater"),
mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.95, ...)

t.test(formula, data, subset, na.action, ...)
```

Выполняется одно или двувыборочный тест Стьюдента.

Одновыборочный t-тест предназначен для проверки равенства среднего значения выборки из нормально распределённой генеральной совокупности некоторому заданному значению в предположении, что дисперсия не известна.

Двувыборочный тест служит для сравнения двух средних значений выборок из нормально распределённых генеральных совокупностей в предположении, что их дисперсии равны, хотя и не известны.

Аргументы:

- **x** — числовой вектор, содержащий элементы первой выборки.
- **y** — числовой вектор, содержащий элементы второй выборки.
- **alternative** — символьный аргумент, определяющий тип альтернативной гипотезы. Возможные значения: **"two.sided"** — средние значения не равны (по умолчанию), **"less"** или **"greater"**.
- **exact** — либо **NULL**, либо логический аргумент. Отвечает за точное вычисление **p-value**. Не используется в двувыборочном тесте, если **alternative** = **"less"** или **alternative** = **"greater"**.
- **mu** — математическое ожидание или разность математических ожиданий, если задано две выборки.
- **paired** — логическое значение, указывающее, требуется ли выполнить спаренный t-тест.
- **var.equal** — логическое значение, считающее, считать ли разбросы равными. Если **TRUE**, то вычисляется разброс для объединённой выборки.
- **conf.level** — доверительный уровень.
- **formula** — формула вида **lhs**$\sim$ **rhs**, где lhs — числовой вектор, а rhs — фактор с двумя классами. Только для двувыборочного теста.
- **data** — матрица или фрейм данных, из которых берутся данные.

- **subset** — вектор, определяющий используемое подмножество наблюдений.
- **na.action** — функция, которая вызывается, как только в данных встретилось значение NA.

Детали:

- Если **y** и **formula** не заданы, то выполняется одновыборочный тест, проверяющий, что выборка **x** имеет среднее, равное 0.
- Если **paired** = **TRUE**, то должны быть определены и иметь одинаковую длину векторы **x** и **y**.
- Значения **NA** и **NaN** из данных удаляются (если **paired** = **TRUE**, то при этом удаляется соответствующее значение из второй выборки).
- Если **var.equal** = **TRUE**, то для оценки отклонения используется объединённая выборка. По умолчанию **var.equal** = **FALSE** и отклонение оценивается отдельно для каждой выборки. При этом происходит надлежащая корректировка числа степеней свободы.

Возвращаемое значение. Объект, возвращаемый функцией **t.test**, — список со следующими полями:

- **statistics** — значение t-статистики Стьюдента.
- **parameter** — число степеней свободы.
- **p.value** —
- **conf.int** — доверительный интервал для математического ожидания.
- **estimate** — оценка математического ожидания для одновыборочного теста или разности математических ожиданий для двувыборочного теста.
- **null.value** — предполагаемое математическое ожидание или разность пред- полагаемых математических ожиданий для двувыборочного теста (входной параметр **mu**).
- **alternative** — символьная строка с описанием альтернативной гипотезы.
- **method** — символьная строка с названием используемой модификации метода.
- **data.name** — строка, содержащее имя (имена) данных, подвергнутых тесту.

Пример 51. *В качестве простых примеров сравним математические ожидания у двух выборок, полученных с помощью функции* **rnorm***:*

```
> set.seed(0)
> x <- rnorm(100, mean = 0, sd = 4)
> y <- rnorm(100, mean = 1, sd = 4)
```

Итак, **x** *и* **y** *— две выборки из нормальных генеральных совокупностей с одинаковыми среднеквадратичными отклонениями, но с разными средними. Применим к выборкам t-тест, полагая в качестве нуль-гипотезы, что генеральные совокупности имеют одинаковое математическое ожидание. Альтернативная гипотеза — средние значения совокупностей не равны.*

```
> t.test(x, y)
Welch Two Sample t-test
data: x and y
t = -1.3896, df = 196.428, p-value = 0.1662
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-1.7590435 0.3048036
sample estimates:
mean of x mean of y
0.0906738 0.8177938
```

Дадим разъяснение полученным результатам. Найдено значение t-статистики, число степеней свободы **df**, *величина* **p-value**. *Указаны границы 95% доверительного интервала для разности мат. ожиданий распределений первой и второй выборки. Приведены оценки математических ожиданий для каждого распре- деления. Пусть уровень значимости равен* $\alpha = 0.1$. *Так как* **p-value** $> \alpha$, *то гипотезу о том, что математические ожидания у распределений равны, принимаем.*

Изменим теперь альтернативную гипотезу. Пусть альтернативная гипотеза постулирует, что вторая генеральная совокупность имеет большее математическое ожидание, чем первая:

```
> t.test(x, y, alternative = "less")
Welch Two Sample t-test
data: x and y
t = -1.3896, df = 196.428, p-value = 0.08311
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
-Inf 0.1376404
sample estimates:
```

```
mean of x mean of y
0.0906738 0.8177938
```

Теперь при уровне значимости $\alpha = 0.1$, *так как* **p-value** $< \alpha$, *то нуль-гипотезу отклоняем и принимаем альтернативную гипотезу.*

Пример 52. *В качестве еще одного примера рассмотрим данные об измерениях скорости света, полученные А.А. Майкельсоном и Э.У. Морли во время знаменитого эксперимента 1887 г.*

Данные содержатся во фрейме **morley** *библиотеки* **datasets**. *Фрейм содержит три столбца:* **Expt** *— номер эксперимента (от 1 до 5),* **Run** *— номер испытания (каждый эксперимент состоял из 20 испытаний),* **Speed** *— скорость света минус 299000 (в км/с). Примем во внимание только последний столбец.*

Предположим, что генеральная совокупность (замеры скорости света) имеет нормальное распределение. Сформулируем нуль-гипотезу: математическое ожидание генеральной совокупности равно 299792.458 *(принятое в настоящее время значение скорости света).*

```
> t.test(light - 792.458)
One Sample t-test
data: light - 792.458
t = 7.5866, df = 99, p-value = 1.824e-11
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
44.26459 75.61941
sample estimates:
mean of x
59.942
```

Дадим разъяснение полученным результатам. Найдено значение статистики, число степеней свободы **df**, *величина* **p-value**. *Указаны границы 95% доверительного интервала для оценки мат. ожидания выборки* **light** *—* 792.458. *Приведены оценки математических ожиданий для каждой группы. Пусть уровень значимости равен* $\alpha = 0.05$. *Так как* **p-value** $< \alpha$, *гипотезу отклоняем.*

4.3 F-тест Фишера

Функции:

```
var.test(x, y, ratio = 1,
```

```
alternative = c("two.sided", "less", "greater"),
conf.level = 0.95, ...)

var.test(formula, data, subset, na.action, ...)
```

Выполняется F-тест Фишера для проверки на равенство стандартных отклонений двух нормально распределённых генеральных совокупностей.

Аргументы:

- **x, y** — числовые векторы, содержащие выборки из разных генеральных совокупностей, или линейные модели (возвращаемые функцией **lm**)
- **ratio** — предполагаемая величина отношения стандартных отклонений в первой и второй генеральных совокупностях
- **alternative** — одно из следующих значений: **"two.side"** (по умолчанию), **"less"**, **"greater"**, обозначающих тип альтернативной гипотезы.
- **conf.level** — доверительный уровень для возвращаемого доверительного интервала.
- **formula** — формула вида **lhs** $\sim$ **rhs**, где **lhs** — числовой вектор, а **rhs** — фактор с двумя классами.
- **data** — матрица или фрейм данных, из которых берутся данные для **formula**.
- **subset** — вектор, определяющий используемое подмножество наблюдений.
- **na.action** — функция, которая вызывается, как только в данных встретилось значение **NA**.

Нулевая гипотеза постулирует, что отношение стандартных отклонений генеральных совокупностей, из которых выбраны x и y соответственно, равно отношению **ratio**.

Возвращаемое значение. Объект, возвращаемый функцией **var.test**, — это список со следующими полями:

- **data** — используемые выборки (названия переменных, которым присвоены значения выборок)
- **F** — значение F-статистики Фишера.
- **num df** — число степеней свободы.

- **denom df**
- **p.value** —
- **alternative hypothesis** —
- **95 percent confidence interva** — .
- **sample estimates** — .

Пример 53. *Рассмотрим две выборки из разных нормальных генеральных совокупностей:*

```
set.seed(0)
x <- rnorm(50, mean = 0, sd = 2)
y <- rnorm(50, mean = 10, sd = 2)
```

Проверим, что генеральные совокупности имеют одинаковое стандартное от- клонение.

```
 var.test(x, y)
```

Результат:

```
F test to compare two variances
data: x and y
 F= 0.7353, num df = 49, denom df = 49, p-value = 0.2852
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
0.4172619 1.2957270
sample estimates:
ratio of variances
0.7352942
```

4.3.1 Критерий χ^2 Пирсона проверки независимости двух выборок

Описание:

```
chisq.test(x, y = NULL, correct = TRUE,
p = rep(1/length(x), length(x)), rescale.p = FALSE,
simulate.p.value = FALSE, B = 2000)
```

Функция реализует критерий χ^2 Пирсона на проверку независимости признаков.

Аргументы функции:

- **x** — вектор или матрица.
- **y** — вектор. Игнорируется, если **x** — матрица.
- **correct** — логическое значение, указывающее, требуется ли применять непрерывную коррекцию для 2×2 матриц.
- **p** — вектор, содержащий вероятности. Должен иметь такую же длину, что и **x**.
- **rescale.p** — логическое значение. Если **TRUE**, то **p** при необходимости нормируется так, чтобы сумма его компонентов была равна 1.
- **simulate.p.value** — логическое значение. Если **TRUE**, то **p-value** вычисляется с помощью метода Монте-Карло, в противном случае используется χ^2 -распределение
- **B** — количество испытаний в методе Монте-Карло.

Детали:

- Если **x** — вектор или матрица с одним столбцом (или одной строкой), а вектор **y** не задан, то **x** рассматривается как статистический ряд (одномерная таблица сопряжённости признаков). Т. е. i-я компонента вектора **x** содержит количество элементов выборки, попавших в i-й интервал группировки. В этом случае выполняется тест на проверку соответствия (согласия) выборки заданным вероятностям **p**. Таким образом, основная (нулевая) гипотеза заключается в том, что вероятность попадания в i-й интервал группировки равна i-й компоненте вектора **p**. По умолчанию, задаются равные вероятности.
- Если **x** — матрица не менее чем с 2 строками и 2 столбцами, то x рассматривается как двумерная таблица сопряжённости признаков и выполняется тест на проверку их независимости

Таким образом, обратим внимание, что если аргумент **y** не задан, то функция **chisq.test** работает со статистическим рядом или таблицей сопряжённости, а не непосредственно с самой выборкой.

- Если **x** и **y** числовые векторы или факторы одной и той же длины (числовые векторы будут преобразованы в факторы), то соответствующие пары их компонентов рассматриваются как реализации двумерной случайной величины $(X; Y)$ и выполняется тест на проверку независимости признаков X и Y .

Результат работы функции. Функция **chisq.test** возвращает список, состоящий из следующих полей:

- **statistics** — значение χ^2 -статистики Пирсона
- **parameter** — число степеней свободы распределения χ^2; равно **NA**, если для отыскания **p-value** использовался метод Монте-Карло,
- **p-value** —
- **method** — символьная строка с названием используемой модификации теста, а также с указанием того, использовались ли непрерывная коррекция и метод Монте-Карло,
- **data.name** — строка, содержащее имя (имена) данных, подвергнутых тесту.
- **observed** — число точек, попавших в i-й интервал группировки.
- **expected** — теоретическое число точек (в предположении выполнения гипотезы), попадающих в i-й интервал группировки.
- **residuals** — остатки Пирсона:

$$\frac{observed - expected}{\sqrt{expected}}$$

Пример 54. *Рассмотрим классический пример с бросанием монеты. Бюффон бросал монету 4040 раз, при этом герб выпал 2048 раз.*

Используя критерий согласия χ^2, проверим, что монета симметрична. Итак, основная гипотеза заключается в том, что вероятность выпадания герба равна $p_1 = 1/2$, вероятность выпадания решки — $p_2 = 1/2$.

```
chisq.test(c(2048, 1992))
```

Результат:

```
Chi-squared test for given probabilities
data: c(2048, 1992)
X-squared = 0.7762, df = 1, p-value = 0.3783
```

Пусть, например, был выбран уровень значимости $\alpha = 0.05$. Так как $\alpha <$ **p-value***, то гипотезу принимаем.*

Пример 55. *Рассмотрим ещё один простор пример проверки независимости двух генеральных совокупностей, если известны выборки* $\mathbf{x}$ *и* $\mathbf{y}$. *Соответствующие пары компонентов векторов будем рассматривать как реализации двумерной случайной величины* $(X;Y)$.

```
set.seed(0)
x <- rnorm(100)
y <- runif(100)
```

Проверяем на независимость:

```
chisq.test(x, y)
```

Выводимые результаты:

```
        Pearson's Chi-squared test

data:  x and y
X-squared = 9900, df = 9801, p-value = 0.239

Предупреждение
In chisq.test(x, y) :
  аппроксимация на основе хи-квадрат может быть неправильной
```

Пусть, например, был выбран уровень значимости $\alpha = 0.05$. *Так как* $\alpha <$ **p-value**, *то гипотезу о независимости случайных признаков можно принять.*

И ещё один пример.

Пример 56. *Таблица* **HairEyeColor** *из библиотеки* **datasets** *содержит информацию о поле, цвете волос и глаз у 592 студентов. Таблица имеет 3 размерности:*

```
"Hair": HairEyeColor["Black", ,], HairEyeColor["Brown", ,],
HairEyeColor["Red", ,], HairEyeColor["Blond", ,],
"Eye": HairEyeColor[, "Brown",], HairEyeColor[, "Blue",],
HairEyeColor[, "Hazel",], HairEyeColor[, "Green",],
"Sex": HairEyeColor[, , "Male"], HairEyeColor[, , "Female"].
```

Элементы таблицы — количество человек из данной группы. Проверим гипотезу о том, что для мужчин цвет глаз не зависит от цвета волос мужчин.

Сначала построим таблицу сопряжённости признаков

```
men <- HairEyeColor[, , "Male"]
```

и выведем её

```
> men
       Eye
Hair    Brown Blue Hazel Green
  Black    32   11    10     3
  Brown    53   50    25    15
  Red      10   10     7     7
  Blond     3   30     5     8
```

Построим мозаичную диаграмму,

```
mosaicplot(men,col = c("chocolate", "cornflowerblue", "salmon", "green"))
```

представленную ниже на рис.4.1

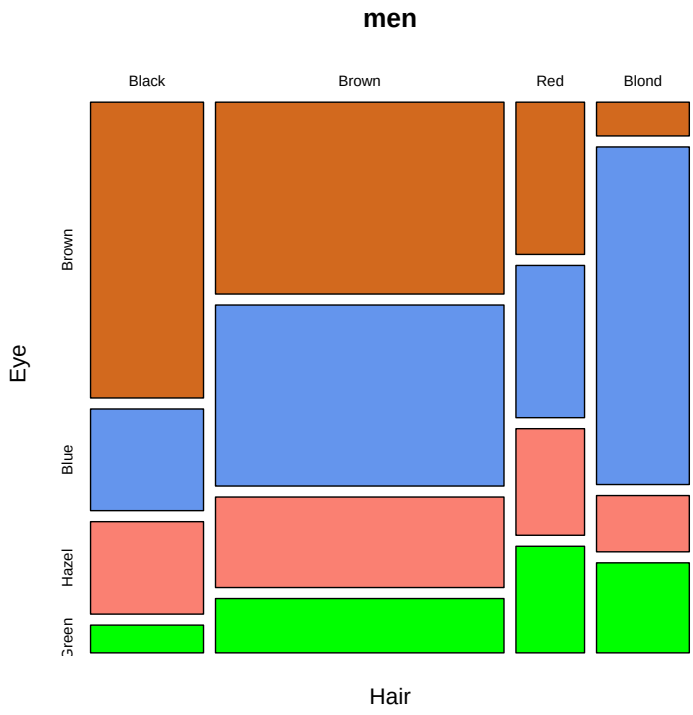

Рис. 4.1: Диаграмма цвета волос и глаз у мужчин

Проверим теперь на независимость.

```
chisq.test(men, simulate.p.value = TRUE)
```

Результат:

```
Pearson's Chi-squared test with simulated p-value (based on 2000
        replicates)

data:  men
X-squared = 41.2803, df = NA, p-value = 0.0004998
```

При уровне значимости, например, $\alpha = 0.05$, гипотезу следует отклонить и признаки считать зависимыми.

Проведём аналогичное исследование для женщин. Строим таблицу сопряжённости признаков.

```
> female <- HairEyeColor[, , "Female"]
> female
       Eye
Hair    Brown Blue Hazel Green
  Black    36    9     5     2
  Brown    66   34    29    14
  Red      16    7     7     7
  Blond     4   64     5     8
```

Мозаичная диаграмма,

```
mosaicplot(female,col = c("chocolate", "cornflowerblue", "salmon", "green"))
```

Проверка на независимость

```
chisq.test(female, simulate.p.value = TRUE)
```

и её результат

```
        Pearson's Chi-squared test with simulated p-value (based on 2000
        replicates)

data:  female
X-squared = 106.6637, df = NA, p-value = 0.0004998
```

При уровне значимости $\alpha = 0.05$, гипотезу о независимости следует отклонить и признаки (цвет волос и глаз) считать зависимыми.

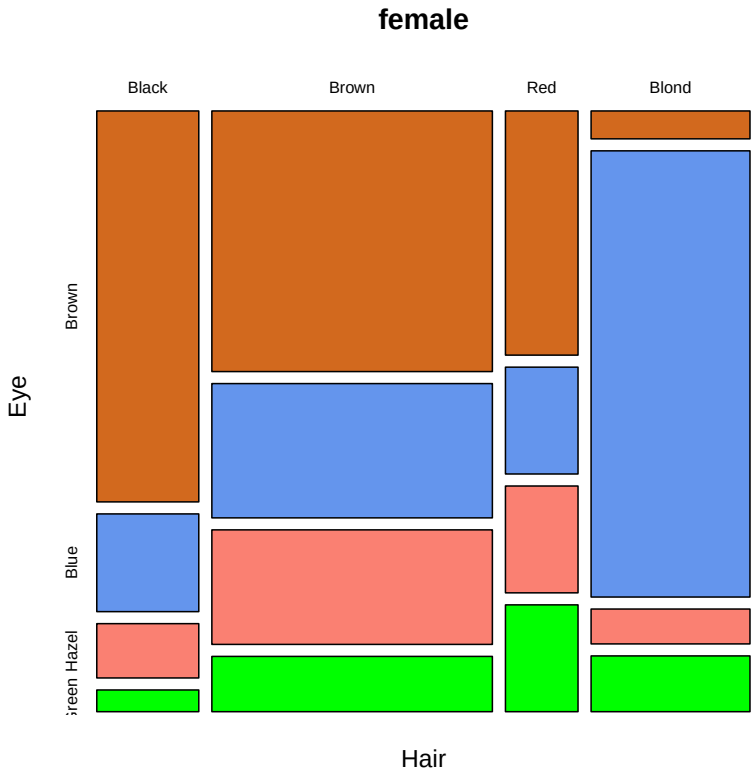

Рис. 4.2: Диаграмма цвета волос и глаз у женщин

Литература

[1] R Development Core Team (2009). **R: A language and environment for statistical computing.** R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

[2] Brian S. Everitt and Torsten Hothorn (2005). **A Handbook of Statistical Analyses Using R**, London and Erlangen, http://www.R-project.org.

[3] Michael J. Crawley (2007). **The R Book**, John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester. ISBN-13: 978-0-470-51024-7.

[4] John M. Chambers (2008). **Software for Data Analysis. Programming with R**, Springer Science+Business Media, LLC, USA, ISBN: 978-0-387-75935-7

[5] Phil Spector (2008). **Data Manipulation with R**, Springer Science+Business Media, LLC, USA, ISBN 978-0-387-74730-9

[6] Yosef Cohen and Jeremiah Y. Cohen (2008). **Statistics and Data with R**, John Wiley & Sons Ltd,The Atrium, Southern Gate, Chichester, ISBN 978-0-470-75805-2

[7] J. H. Maindonald (2001). **Using R for Data Analysis and Graphics An Introduction.** Statistical Consulting Unit of the Graduate School, Australian National University.

[8] John Verzani. **simpleR — Using R for Introductory Statistics**, http://cran.R-project.org/other-docs.html.

[9] Jim Lemon. **Kickstarting R**, http://cran.R-project.org/other-docs.html.

[10] Зарядов И.С.(2010) **Введение в статистический пакет R: типы переменных, структуры данных, чтение и запись информации, графика.** Учебно-методическое пособие, РУДН, 2010.

[11] Золотых Н. Ю., Половинкин А.Н. (2007). **Машинное обучение. Лабораторный практикум**, http://www.uic.unn.ru/ zny/ml/

[12] Меретилов М.А. (2006). **Методические указания к лабораторным работам по курсу «Методы анализа данных»**, http://gis-lab.info/docs/r-metoda-2006.10.23.pdf

[13] Шипунов А.Б. **R - объектно-ориентированная статистическая среда**, http://herba.msu.ru/shipunov/software/r/r-ru.htm