

Липецкий государственный технический университет

Кафедра прикладной математики

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ МАТЕМАТИЧЕСКИХ ИССЛЕДОВАНИЙ

Лекция 2

Структуры языка R. Пользовательские функции.

Решение задач линейной алгебры с помощью R.

Описательная статистика

Составитель - Сысоев А.С., к.т.н., доц.

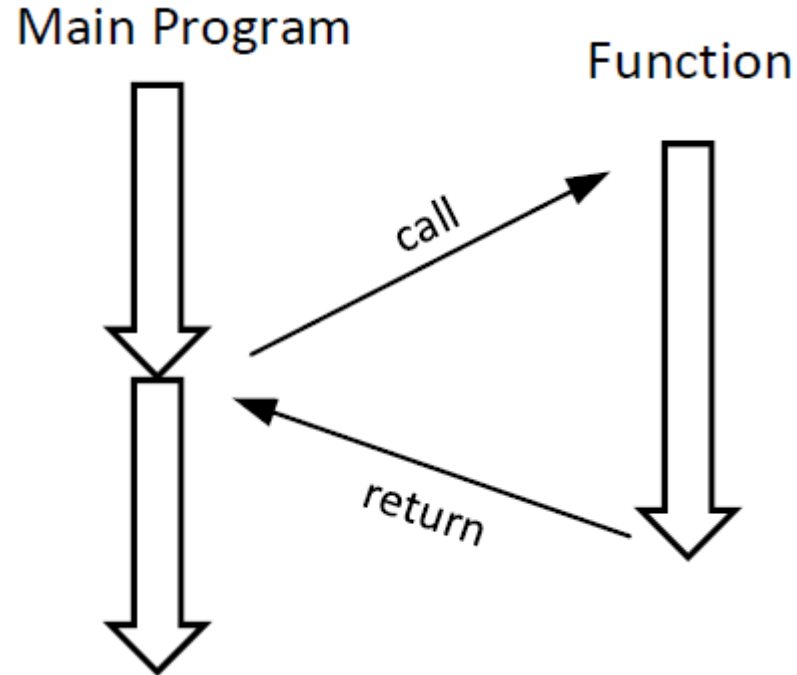
Липецк – 2019

Outline

- 3.1. Функции. Пользовательские функции
 - 3.1.1. Синтаксис
 - 3.1.2. Примеры функций
- 3.2. Математические выражения
- 3.3. Циклы
 - 3.3.1. Условие If-Else. Использование ifelse
 - 3.3.2. Цикл For
 - 3.3.3. Цикл While
 - 3.3.4. Пример сложной функции
- 3.4. Задачи линейной алгебры
 - 3.4.1. Произведение векторов и матриц. Единичная матрица
 - 3.4.2. Обращение и псевдообращение матриц
 - 3.4.3. Определитель матрицы. Собственные числа и собственные векторы
 - 3.4.4. Определенность матриц
 - 3.4.5. Гессиан. Аппроксимация функции разложением в ряд Тейлора.
- 4.1. Основные статистические функции
- 4.2. Стандартизация данных
- 4.3. Статистические распределения
 - 4.3.1. Дискретные статистические распределения
 - 4.3.2. Непрерывные статистические распределения
 - 4.3.3. Воспроизводимость результатов при использовании ГПСЧ
- 4.4. Подгонка статистического распределения
- 4.5. Проверка распределения на нормальность

3.1. Функции. Пользовательские функции

Функции создаются для того, чтобы выполнять определенные стандартные действия более одного раза.



Примеры: 1) стандартные функции для работы с векторами и матрицами

`sum(x)` `mean(x)` `max(x)` `min(x)`

2) стандартные тригонометрические функции

`cos(x)` `sin(x)` `tan(x)` ...

Тригонометрические функции определены для углов в радианах!!!

`sin(pi/2)` `sin(90)`

3.1. Функции. Пользовательские функции

3.1.1. Синтаксис

R предоставляет большое количество встроенных функций с возможностью гибкой пользовательской настройки.

ФУНКЦИЯ в R может содержать

- ✓ **Имя**, через которое происходит обращение (не должно совпадать с именами существующих функций)
- ✓ **Тело функции**, содержащие основные операции
- ✓ **Аргументы**, определяющие основные параметры ее выполнения
- ✓ **Значение**, которое необходимо вернуть в результате выполнения функции

Простой пример:

```
f <- function(x,y) {  
  return(2*x + y^2)  
}  
f(-3, 5)  
## [1] 19
```

3.1. Функции. Пользовательские функции

3.1.1. Синтаксис

СИНТАКСИС

```
functionname <- function(argument1, argument2, ...) {  
  function_body  
  return(value)  
}
```

- ✓ Возвращаемое значение (по умолчанию) - результат, **последнего** выполненного действия
- ✓ В случае, если функция содержит только одно выражение, фигурные скобки могут быть опущены (однако этого делать не рекомендуется)
- ✓ Порядок аргументов имеет значение
- ✓ Промежуточные значения вычислений внутри функций не выводятся в консоль, однако, `print(...)` может решить эту проблему

```
function_0 <- function(x,y,z){  
  x <- y + 3;  
  z <- x^3;  
  k <- x + y + z;  
  return(k)}  
> function_0(10,3,4)  
[1] 225
```

```
function_1 <- function(x,y,z){  
  x <- y + 3;  
  z <- x^3; prom <- c(x,z); print(prom);  
  k <- x + y + z;  
  return(k)}  
> function_1(10,3,4)  
[1] 6 216  
[1] 225
```

3.1. Функции. Пользовательские функции

3.1.2. Примеры функций

```
square <- function(x) x*x  
> square(5)  
[1] 25
```

```
hello <- function() print("Привет!")  
> hello()  
[1] "Привет!"
```

```
cubic <- function(x){  
  print(c("Значение: ", x, "Куб: ", x*x*x))  
> cubic(5)  
[1] "Значение: " "5"      "Куб: "      "125"
```

```
my_mean <- function(x){  
  return(sum(x)/length(x))  
> my_mean(seq(10,20))
```

Переменные, созданные в функции, существуют только в ней, они **ЛОКАЛЬНЫ**.

```
> x <- "A"  
g <- function(x){  
  x <- "B"; return(x)}  
> x <- "C"  
> g(x)      [1] ???  
> x         [1] ???
```

ПРИМЕР ПОЛЬЗОВАТЕЛЬСКОЙ ФУНКЦИИ

Задача: На автомагистрали могут возникать транспортные заторы. Существует множество программных продуктов, способных моделировать это. У автомагистрали есть определенная пропускная способность, экспериментально доказано, что это случайное значение, которое можно описать распределением Вейбулла. Определить, какое распределение имеет резерв транспортного потока (разность пропускной способности и интенсивности поступления).

3.1. Функции. Пользовательские функции

3.1.2. Примеры функций

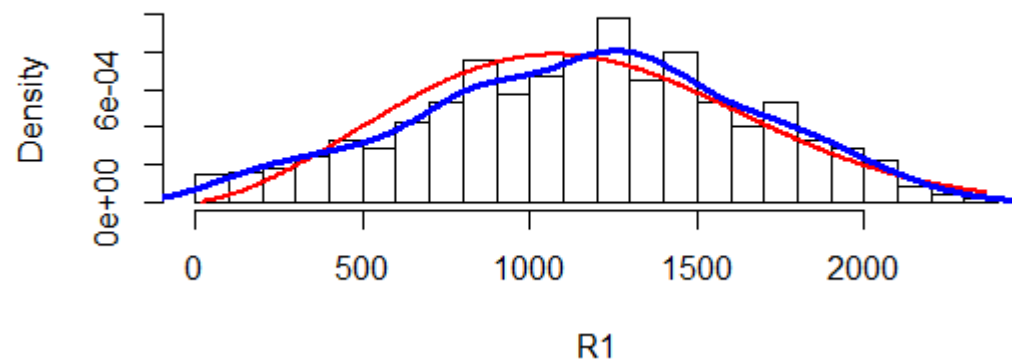
```
reserve<-function(num, fl_rate, c_hbs){
  library(nortest); library(MASS);
  flow_rate<-fl_rate*rnorm(n=num,mean=1,sd=4/sqrt(fl_rate));
  capacity<-rweibull(n=num,shape=15,scale=1.275*c_hbs);
  raz<-capacity-flow_rate;
  raz.1 <- split(raz[raz>0],cumsum(raz==0)[raz>0]); R1<-sapply(raz.1, as.numeric);
  raz.2 <- split(raz[raz<0],cumsum(raz==0)[raz<0]); R2<-sapply(raz.2, as.numeric);
  opar <- par(no.readonly=TRUE); par(mfrow=c(2,1));
  hist(R1, freq=FALSE, breaks=20, main="Undersaturated flow, Weibull");
  xfit<-seq(min(R1), max(R1), length=length(R1));
  fit.weibull <- fitdistr(R1, "weibull");
  yfit<-dweibull(xfit, shape= fit.weibull$estimate["shape"],
+ scale=fit.weibull$estimate["scale"]);
  lines(xfit, yfit, lwd=2, col="red"); lines(density(R1), col="blue", lwd=3);
  hist(R2, freq=FALSE, breaks=20, main="Oversaturated flow, Normal");
  xfit<-seq(min(R2), max(R2), length=length(R2));
  fit.normal <- fitdistr(R2, "normal");
  yfit<-dnorm(xfit, mean=fit.normal$estimate["mean"], sd=fit.normal$estimate["sd"])
  lines(xfit, yfit, lwd=2, col="red"); lines(density(R2), col="blue", lwd=3);
  par(opar); print("Undersaturated flow, Weibull distribution"); print(fit.weibull);
  print(ks.test(R1, "pweibull", scale=fit.weibull$estimate["scale"],
+ shape=fit.weibull$estimate["shape"]));
  print("Oversaturated flow, Normal distribution");
  print(fit.normal); print(lillie.test(raz)) }
```

3.1. Функции. Пользовательские функции

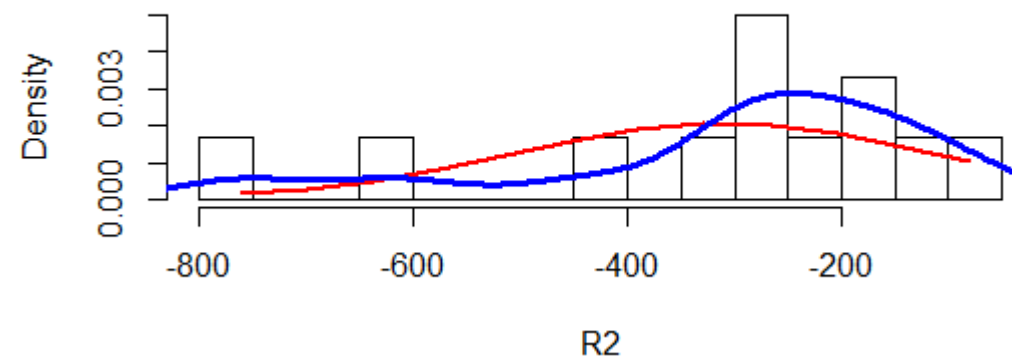
3.1.2. Примеры функций

> reserve(500,4500,4560)

Undersaturated flow, Weibull



Oversaturated flow, Normal



3.1. Функции. Пользовательские функции

3.1.2. Примеры функций

```
> reserve(500,4500,4560)
[1] "Undersaturated flow, Weibull distribution"
      shape      scale
 2.550446e+00  1.305352e+03
(9.459161e-02) (2.424581e+01)
```

One-sample Kolmogorov-Smirnov test

```
data: R1
D = 0.055762, p-value = 0.09616
alternative hypothesis: two-sided
```

```
[1] "Oversaturated flow, Normal distribution"
      mean      sd
-310.62464  192.93086
( 55.69434) ( 39.38185)
```

Lilliefors (Kolmogorov-Smirnov) normality test

```
data: raz
D = 0.045931, p-value = 0.01357
```

3.2. Математические выражения

В R выражения могут хранить в себе символьные математические структуры, которые в последствии могут быть модифицированы (например, вычислены их частные производные).

Для задания такого рода объекта существует функция `expression()`

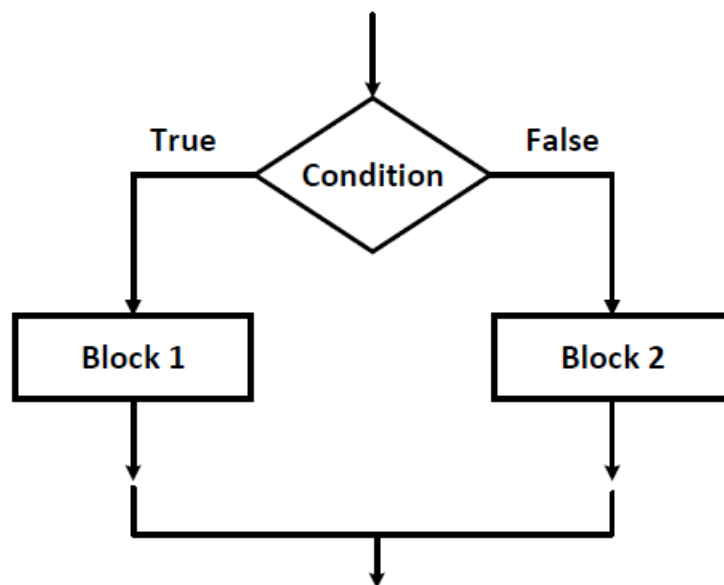
```
f <- expression(x^3 + 3*y - y^3 - 3*x)
> f
expression(x^3 + 3 * y - y^3 - 3 * x)
```

Если необходимо вычислить значение созданного выражения при определенных значениях переменных, входящих в него, используют функцию `eval()`

```
> x <- 2
> y <- 3
> eval(f)
[1] -16
```

3.3. Циклы

3.3.1. Условие If-Else. Использование ifelse



Условие If

```
if (condition) {  
  statement1  
}
```

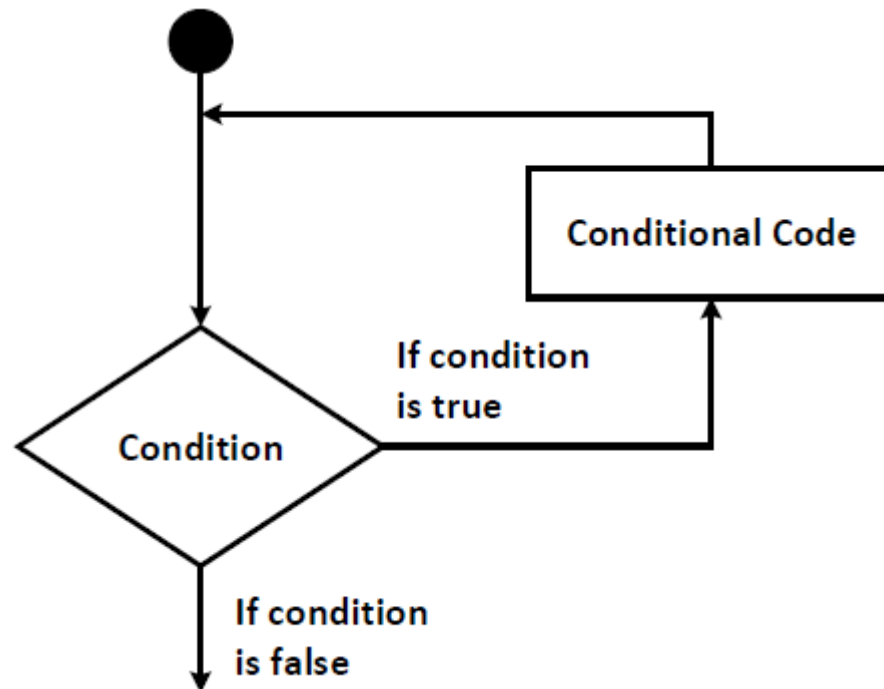
Условие If-Else

```
if (condition) {  
  statement1  
} else {  
  statement2  
}
```

```
grades <- c(1, 2, 3, 4, 5)  
ifelse(grades <= 4, "Passed", "Failed")  
## [1] "Passed" "Passed" "Passed" "Passed" "Failed"
```

3.3. Циклы

3.3.2. Цикл For



```
for (i in 4:7) {  
  print(i)  
}
```

```
## [1] 4  
## [1] 5  
## [1] 6  
## [1] 7
```

```
a <- c()  
for (i in 1:3){  
  a[i] <- sqrt(i)  
}  
a  
## [1] 1.000000 1.414214 1.732051
```

```
for (counter in looping_vector){  
  # code to be executed for each element in the sequence  
}
```

3.3. Циклы

3.3.3. Цикл While

```
> z <- 1
> while(z <= 4){
  print(z);
  z <- z + 1}
[1] 1
[1] 2
[1] 3
[1] 4
```

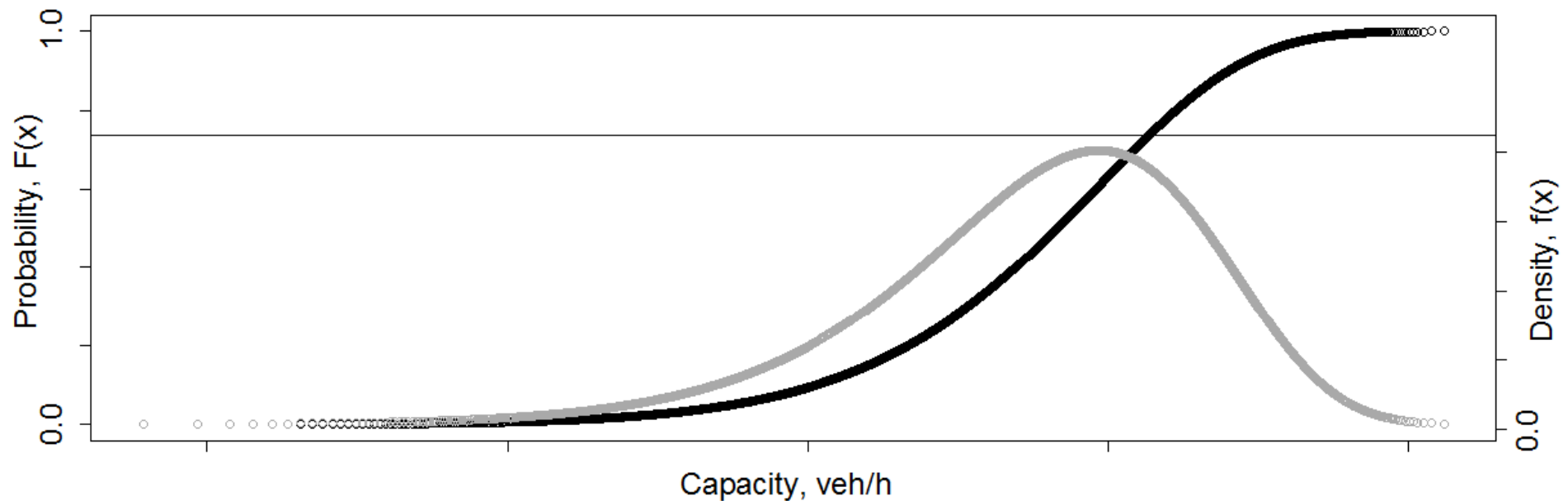
- ✓ Количество итераций определяется выполнением заданного условия
- ✓ Условия проверяется в начале каждой итерации

```
while (condition) {
  # code to be executed
}
```

3.3. Циклы

3.3.4. Пример сложной функции

Задача: При моделировании потока транспортных средств на автомагистрали используется распределение Вейбулла пропускной способности полосы. Но при компьютерном моделировании необходимо найти такие наименьшее и наибольшее значения случайного равномерно распределенного числа, используемого для моделирования, чтобы получить адекватные значения смоделированной пропускной способности.



3.3. Циклы

3.3.4. Пример сложной функции

```
poisk <- function(lambda,k,h1){
  B=0;i=0; ... # определяем локальные переменные
  for(i in 1:((1/h)-h)){
    point <- point + h; value[i] <- lambda*(-log(1-point))^(1/k); prob[i] <- i*h}
  for(i in 1:length(value)){
    z[i] <- k*(lambda^(-k))*(value[i]^(k-1))*exp(-(lambda/value[i])^(-k));
    t[i] <- (k*exp(-(value[i]/lambda)^k)*((value[i]/lambda)^k)*
      *(k-1-((value[i]/lambda)^2)*k))/(value[i]^2)}
  low <- which(t==max(t)); upp <- which(t==min(t));
  for(i in 1:length(value)){
    x[i] <- (k*exp(-(value[i]/lambda)^k)*(exp(-(value[i]/lambda)^k)*k^2-
      -3*exp(-(value[i]/lambda)^k)*k-3*exp(-(value[i]/lambda)^2*k)*k^2+2*exp(-
      (value[i]/lambda)^k)+3*exp(-(value[i]/lambda)^2*k)*k+exp(-
      -(value[i]/lambda)^3*k)*k^2))/(value[i]^3)}
  for(i in 2:(length(value))){
    diff[i] <- x[i]-x[i-1]; diff1[i-1] <- diff[i]; ind[i] = i; ind[1] = 1;
    diff1[length(value)]=diff1[length(value)-1]}
  fit_power <- nls(diff1 ~ a * ind^b, start = list(a=0.1, b=0.1))
  for(i in 2:length(ind)){
    deriv1[i] <- coef(fit_power)[1]*coef(fit_power)[2]*ind[i]^(coef(fit_power)[2]-1)
    deriv2[i] <- coef(fit_power)[1]*coef(fit_power)[2]*(coef(fit_power)[2]-1)*
      *ind[i]^(coef(fit_power)[2]-2);
    curve[i] <- abs(deriv2[i])/((1+deriv1[i]^2)^1.5)}
```

продолжение далее

3.3. Циклы

3.3.4. Пример сложной функции

продолжение

```
sred <- mean(curve)
for(i in 1:length(curve)){
  if(abs(curve[i]-sred) >
    0.1^abs(ceiling(log10(mean(curve)))))
    {number <- ind[i]} }
a1 <- value[low];
a2 <- pweibull(value[low], k, lambda);
a3 <- value[upp];
a4 <- pweibull(value[upp], k, lambda);
a5 <- value[number];
a6 <- pweibull(value[number], k, lambda)
B <- c(a1,a3)
return(B)
}
```

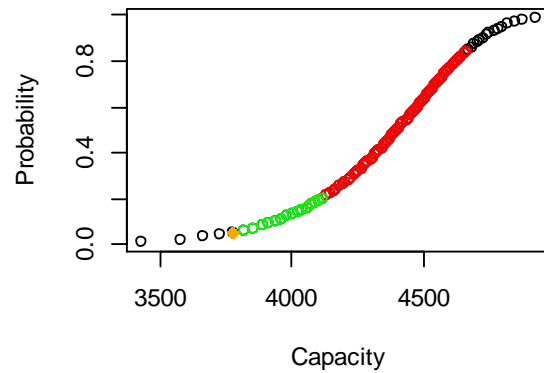
```
poisk(4500,17,100)
[1] "lower threshold"
[1] 4043.851
[1] "lower probability rate"
[1] 0.15
[1] "adjusted lower threshold"
[1] 3778.624
[1] "adjusted lower probability rate"
[1] 0.05
[1] "upper threshold"
[1] 4663.23
[1] "probability rate"
[1] 0.84
```

```
poisk(4500, 17, 100)
[1] 4043.851 4663.230
```

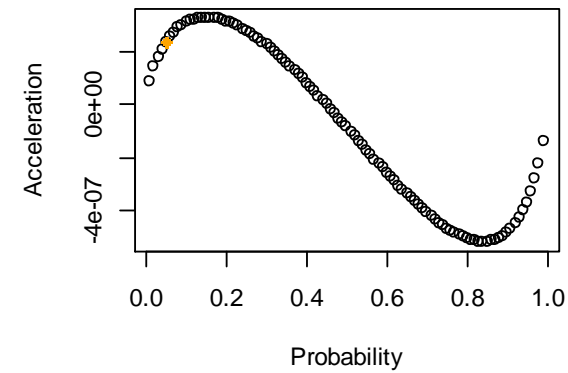

3.3. Циклы

3.3.4. Пример сложной функции

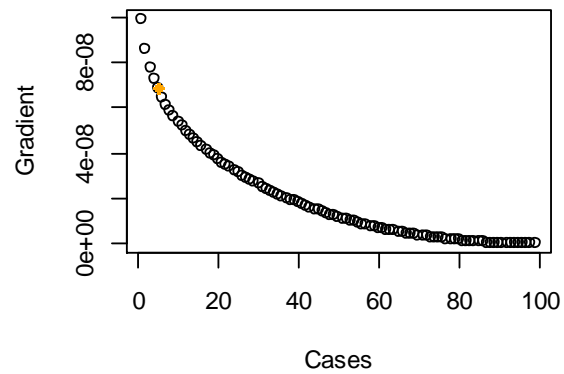
Capacity distribution



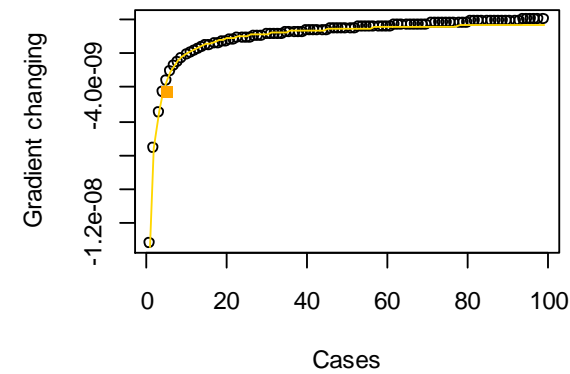
Acceleration graph



Acceleration gradient



Acceleration gradient changing



3.4. Задачи линейной алгебры

3.4.1. Произведение векторов и матриц. Единичная матрица

- Произведение вектора или матрицы на скаляр

```
5*c(1, 2, 3)
## [1]  5 10 15
m <- matrix(c(1,2, 3,4, 5,6), ncol=3)
```

- Произведение векторов или матриц

```
m %*% x
##           [,1]
## [1,]       22
## [2,]       28
```

- Единичная матрица

```
diag(3)
##           [,1] [,2] [,3]
## [1,]        1   0   0
## [2,]        0   1   0
## [3,]        0   0   1
```

3.4. Задачи линейной алгебры

3.4.2. Обращение и псевдообращение матриц

- Обратная матрица

```
sq.m <- matrix(c(1,2, 3,4), ncol=2)
sq.m

##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4

solve(sq.m)

##      [,1] [,2]
## [1,]   -2  1.5
## [2,]    1 -0.5

sq.m %*% solve(sq.m) - diag(2) # post check

##      [,1] [,2]
## [1,]    0    0
## [2,]    0    0
```

3.4. Задачи линейной алгебры

3.4.2. Обращение и псевдообращение матриц

- Псевдообратная матрица

```
library(MASS)
```

```
ginv(m)
```

```
##           [, 1]      [, 2]  
## [1, ] -1.3333333  1.0833333  
## [2, ] -0.3333333  0.3333333  
## [3, ]  0.6666667 -0.4166667
```

```
m %*% ginv(m)
```

```
##           [, 1] [, 2]  
## [1, ] 1.0000000e+00  0  
## [2, ] 2.664535e-15  1
```

3.4. Задачи линейной алгебры

3.4.3. Определитель матрицы. Собственные числа и собственные векторы

```
det(sq.m)
```

```
## [1] -2
```

```
sq.m
```

```
##      [,1] [,2]
```

```
## [1,]    1    3
```

```
## [2,]    2    4
```

```
e <- eigen(sq.m)
```

```
e$val # eigenvalues
```

```
## [1]  5.3722813 -0.3722813
```

```
e$vec # eigenvectors
```

```
##      [,1]      [,2]
```

```
## [1,] -0.5657675 -0.9093767
```

```
## [2,] -0.8245648  0.4159736
```

3.4. Задачи линейной алгебры

3.4.4. Определенность матриц

Матрица Q является *отрицательно определённой*, если $-Q$ есть положительно определённая матрица.

Матрица Q является *отрицательно полуопределённой*, если $-Q$ есть положительно полуопределённая матрица.

Матрица Q является *неопределённой*, если квадратичная форма $x^T Q x$ может принимать как положительные, так и отрицательные значения.

$$\begin{aligned} Q &= \begin{pmatrix} 1 & -1 \\ 1 & -2 \end{pmatrix}, \quad \varphi(x) = (x_1, x_2) \begin{pmatrix} 1 & -1 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \\ &= (x_1, x_2) \begin{pmatrix} x_1 - x_2 \\ x_1 - 2x_2 \end{pmatrix} = x_1(x_1 - x_2) + x_2(x_1 - 2x_2) = \\ &= x_1^2 - x_1x_2 + 2x_2^2 = x_1^2 - 2x_2^2. \end{aligned}$$

3.4. Задачи линейной алгебры

3.4.4. Определенность матриц

Для проверки определённости (полуопределённости) матрицы служат *критерии Сильвестра*:

1. Матрица положительно определена, если:
 - а) все диагональные элементы положительны;
 - б) все угловые миноры матрицы положительны.
2. Матрица отрицательно определена, если:
 - а) все диагональные элементы отрицательны;
 - б) все угловые миноры матрицы имеют чередующиеся знаки, начиная со знака «-».
3. Матрица положительно полуопределена, если значения диагональных элементов и главных миноров матрицы неотрицательны.

3.4. Задачи линейной алгебры

3.4.4. Определенность матриц

```
library(matrixcalc)
```

```
I <- diag(3)
I

##           [,1] [,2] [,3]
## [1,]         1    0    0
## [2,]         0    1    0
## [3,]         0    0    1

is.negative.definite(I)

## [1] FALSE

is.positive.definite(I)

## [1] TRUE
```

```
C <- matrix(c(-2,1,0, 1,-2,1, 0,1,-2),
            nrow=3, byrow=TRUE)
C

##           [,1] [,2] [,3]
## [1,]        -2    1    0
## [2,]         1   -2    1
## [3,]         0    1   -2

is.positive.semi.definite(C)

## [1] FALSE

is.negative.semi.definite(C)

## [1] TRUE
```


3.4. Задачи линейной алгебры

3.4.5. Гессиан. Аппроксимация функции разложением в ряд Тейлора

$$H(\mathbf{x}) = \nabla^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(\mathbf{x}) \end{bmatrix}$$

```
f <- function(x) (x[1]^3*x[2]^2-x[2]^2+x[1])
optimHess(c(3,2), f, control=(ndeps=0.0001))

##      [,1] [,2]
## [1,]   72  108
## [2,]  108   52
```

3.4. Задачи линейной алгебры

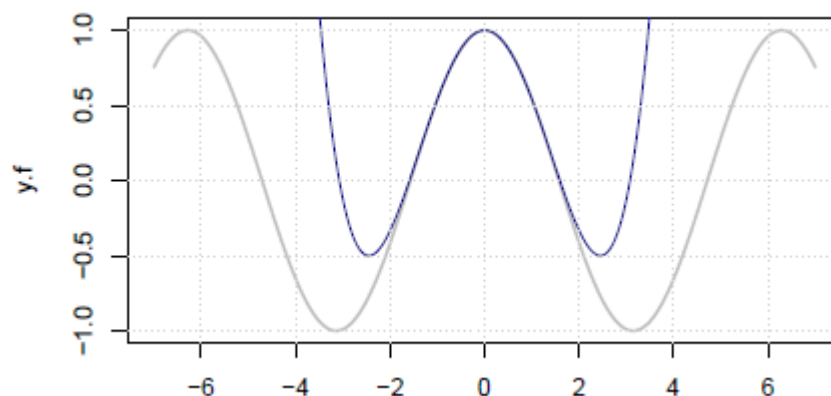
3.4.5. Гессиан. Аппроксимация функции разложением в ряд Тейлора

```
library(pracma)

f <- function(x) cos(x)
taylor.poly <- taylor(f, x0=0, n=4)
taylor.poly

## [1] 0.04166733 0.00000000 -0.50000000 0.00000000 1.00000000

x <- seq(-7.0, 7.0, by=0.01)
y.f <- f(x)
y.taylor <- polyval(taylor.poly, x)
plot(x, y.f, type="l", col="gray", lwd=2, ylim=c(-1, +1))
lines(x, y.taylor, col="darkblue")
grid()
```



4.1. Основные статистические функции

Функция	Описание
<code>mean(x)</code>	Среднее арифметическое <code>mean(c(1, 2, 3, 4))</code> равно 2.5
<code>median(x)</code>	Медиана <code>median(c(1, 2, 3, 4))</code> равно 2.5
<code>sd(x)</code>	Стандартное отклонение <code>sd(c(1, 2, 3, 4))</code> равно 1.29
<code>var(x)</code>	Дисперсия <code>var(c(1, 2, 3, 4))</code> равно 1.67
<code>mad(x)</code>	Абсолютное отклонение медианы <code>mad(c(1, 2, 3, 4))</code> равно 1.48
<code>quantile(x, probs)</code>	Квантили, где <i>x</i> – числовой вектор, для которого нужно вычислить квантили, а <i>probs</i> – числовой вектор с указанием вероятностей в диапазоне [0; 1] # 30-й и 84-й процентиля <i>x</i> <code>y <- quantile(x, c(.3, .84))</code>

Пример: 1) вычислить среднее арифметическое для всех элементов объекта *x*

```
y <- mean(x)
```

2) вычислить усеченное среднее, исключив 5% наибольших и 5% наименьших значений в выборке, не принимая при этом во внимание пропущенные значения.

```
z <- mean(x, trim = 0.05, na.rm=TRUE)
```

4.1. Основные статистические функции

Функция	Описание
<code>range(x)</code>	Размах значений <code>x <- c(1,2,3,4)</code> <code>range(x)</code> равно <code>c(1,4)</code> . <code>diff(range(x))</code> равно 3
<code>sum(x)</code>	Сумма <code>sum(c(1,2,3,4))</code> равно 10
<code>diff(x, lag=n)</code>	Разность значений в выборке, взятых с заданным интервалом (<code>lag</code>). По умолчанию интервал равен 1. <code>x <- c(1,5,23,29)</code> <code>diff(x)</code> равно <code>c(4, 18, 6)</code>
<code>min(x)</code>	Минимум <code>min(c(1,2,3,4))</code> равно 1
<code>max(x)</code>	Максимум <code>max(c(1,2,3,4))</code> равно 4
<code>scale(x, center=TRUE, scale=TRUE)</code>	Значения объекта <code>x</code> , центрованные (<code>center=TRUE</code>) или стандартизованные (<code>center=TRUE, scale=TRUE</code>) по столбцам.

4.1. Основные статистические функции

В системе R имеется возможность быстрого расчета основных параметров описательной статистики.

Функция общего назначения `summary()`:

```
summary(mtcars$mpg)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
10.40  15.35   19.20   20.00  22.15   33.90     1.00
```

Функция `describe()` пакета `Hmisc`:

```
# Пакет Hmisc, функция describe():
describe(mtcars)
mtcars
11 Variables      32 Observations
-----
mpg
  n missing  unique  Mean   .05   .10   .25   .50   .75   .90   .95
31      1     25    20 11.85 14.30 15.35 19.20 22.15 30.40 31.40
lowest : 10.4 13.3 14.3 14.7 15.0, highest: 26.0 27.3 30.4 32.4 33.9
-----
```

4.2. Стандартизация данных

СТАНДАРТИЗАЦИЯ (НОРМАЛИЗАЦИЯ) ДАННЫХ

По умолчанию функция `scale()` стандартизирует заданный столбец матрицы или таблицы данных так, чтобы его среднее арифметическое было равно нулю, а стандартное отклонение – единице.

Для преобразования каждого столбца так, чтобы его среднее арифметическое и стандартное отклонение приобрели заданные значения:

$$\text{newdata} \leftarrow \text{scale}(\text{mydata}) * \text{SD} + \text{M}$$

где M – это нужное значение среднего арифметического, а SD – стандартного отклонения.

Чтобы стандартизировать определенный столбец, а не всю матрицу или таблицу данных целиком:

$$\text{newdata} \leftarrow \text{transform}(\text{mydata}, \text{myvar} = \text{scale}(\text{myvar}) * 10 + 50).$$

4.3. Статистические распределения

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

дискретные:

- **биномиальное**;
- **пуассоновское**;
- геометрическое;
- **гипергеометрическое**;
- отрицательно биномиальное;
- **полиномиальное**;

непрерывные:

- бета-распределение;
- распределение Коши;
- **экспоненциальное**;
- χ^2 -распределение;
- распределение Фишера (f-распределение);
- гамма-распределение;
- логнормальное;
- логистическое;
- **нормальное**;
- распределение Стьюдента (t-распр.);
- равномерное;
- **распределение Вейбулла**.

ранговые распределения Вилкоксона

4.3. Статистические распределения

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.

4.3. Статистические распределения

4.3.1. Дискретные статистические распределения

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 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)
```

4.3. Статистические распределения

4.3.1. Дискретные статистические распределения

БИНОМИАЛЬНОЕ РАСПРЕДЕЛЕНИЕ (ПРОДОЛЖЕНИЕ)

Аргументы функций:

- x – целочисленный неотрицательный вектор – вектор значений случайной величины ξ ;
- q – неотрицательный вектор – вектор квантилей;
- p – вектор вероятностей;
- n – длина создаваемого вектора;
- $size$ – число испытаний Бернулли;
- $prob$ – вероятность «успеха» в одном испытании Бернулли;
- log – логарифмический аргумент (по умолчанию FALSE). Нужно ли вычислять логарифм вероятности;
- $log.p$ – аналогично;
- $lower.tail$ – логический аргумент. Если установлен в TRUE, то используется $P\{\xi \leq k\}$, в противном случае $P\{\xi > k\}$.

4.3. Статистические распределения

4.3.1. Дискретные статистические распределения

ПУАССОНОВСКОЕ РАСПРЕДЕЛЕНИЕ

Моделирует случайную величину, представляющую собой число событий, произошедших за фиксированное время, при условии, что данные события происходят с некоторой фиксированной средней интенсивностью и независимо друг от друга.

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

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

В R для пуассоновского распределения реализованы функции:

```
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)
```

4.3. Статистические распределения

4.3.1. Дискретные статистические распределения

ГИПЕРГЕОМЕТРИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ

В урне имеется m белых и n черных шаров. Из урны без возвращения вынимают k шаров ($0 < k < n + m$). Случайная величина ξ , описывающая число i ($0 \leq i \leq \min(k, m)$) вытянутых белых шаров, подчиняется гипергеометрическому распределению. (Пример: описывает вероятность того, что в выборке из n различных объектов, вытянутых из поставки, ровно k объектов являются бракованными.)

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

В R для гипергеометрического распределения реализованы функции:

```
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)
```

4.3. Статистические распределения

4.3.1. Дискретные статистические распределения

ПОЛИНОМИАЛЬНОЕ РАСПРЕДЕЛЕНИЕ

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

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

В R представлено всего двумя функциями:

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

4.3. Статистические распределения

4.3.2. Непрерывные статистические распределения

ЭКСПОНЕНЦИАЛЬНОЕ РАСПРЕДЕЛЕНИЕ

Моделирует время между двумя последовательными свершениями одного и того же события.

Случайная величина X имеет экспоненциальное распределение с параметром λ , если её плотность имеет вид

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

В R для экспоненциального распределения реализованы функции:

```
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)
```

.

4.3. Статистические распределения

4.3.2. Непрерывные статистические распределения

РАСПРЕДЕЛЕНИЕ ВЕЙБУЛЛА

Относится к двухпараметрическим распределениям, используется в демографических исследованиях, анализе дожития (исследовании смертности). Частным случаем распределения Вейбулла является экспоненциальное распределение.

Плотность распределения

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

```
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 числа.

4.3. Статистические распределения

4.3.2. Непрерывные статистические распределения

НОРМАЛЬНОЕ РАСПРЕДЕЛЕНИЕ

Плотность нормального распределения

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

где m – математическое ожидание, σ – среднее квадратическое 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)
```

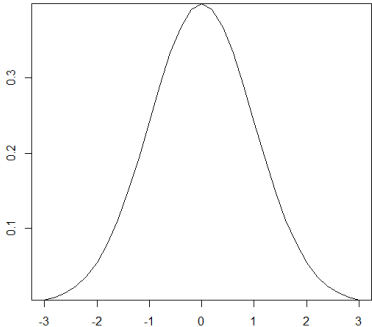
```
rnorm(n, mean = 0, sd = 1)
```

Замечания: 1) заданный порядок аргументов функций является обязательным;
2) 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.

4.3. Статистические распределения

4.3.2. Непрерывные статистические распределения

НОРМАЛЬНОЕ РАСПРЕДЕЛЕНИЕ (ПРОДОЛЖЕНИЕ)

ЗАДАЧА	РЕШЕНИЕ
<p>Как изобразить кривую стандартного нормального распределения в диапазоне значений $[-3, 3]$?</p> 	<pre>x <- pretty(c(-3,3), 30) y <- dnorm(x) plot(x, y, + type = "l", + xlab = "Normal Deviate", + ylab = "Density", + yaxs = "i")</pre>
<p>Какова площадь под кривой стандартного нормального распределения слева от $z=1.96$?</p>	<pre>pnorm(1.96) [1] 0.9750021</pre>
<p>Каково значение 90-го перцентиля нормального распределения со средним значением 500 и стандартным отклонением 100?</p>	<pre>qnorm(.9, mean=500, sd=100) [1] 628.1552</pre>
<p>Как создать 50 случайных чисел, принадлежащих нормальному распределению со средним значением 50 и стандартным отклонением 10?</p>	<pre>rnorm(50, mean=50, sd=10)</pre>

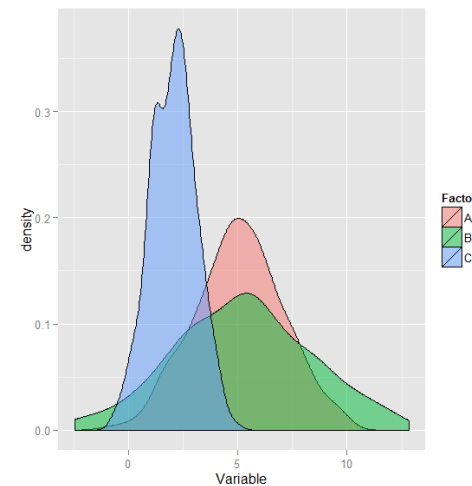
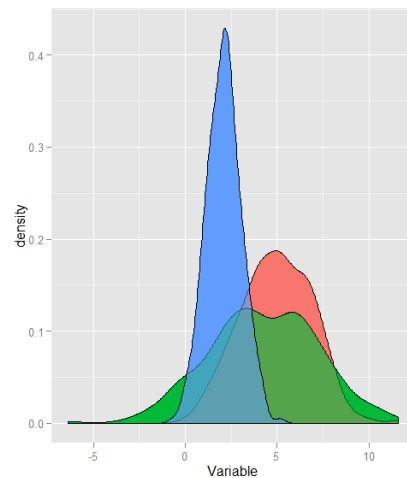
4.3. Статистические распределения

4.3.3. Воспроизводимость результатов при использовании ГПСЧ

Генератор псевдослучайных чисел (ГПСЧ) начинает свою работу с определенной точки в пространстве возможных чисел. Эта точка называется **начальным числом**.

Пример: создадим таблицу `example` с двумя столбцами. В первом столбце будут храниться коды уровней гипотетического фактора `Factor` (три уровня: A, B, и C). Для каждого из этих уровней сгенерируем (псевдо-)случайным образом по 300 нормально распределенных значений с разными средними и стандартными отклонениями.

```
example = data.frame(Factor = rep(c("A", "B", "C"), each = 300),  
+ Variable = c(rnorm(300, 5, 2), rnorm(300, 4, 3), rnorm(300, 2, 1)))
```



Выход - `set.seed(...)`

4.4. Подгонка статистического распределения

Можно выделить **4 шага при подборе распределений**:

- 1) Выбор модели: выдвигается гипотеза о принадлежности выборки некоторому семейству распределений;
- 2) Оценка параметров теоретического распределения;
- 3) Оценка качества приближения;
- 4) Проверка согласия между наблюдаемыми и ожидаемыми значениями с использованием статистических тестов.

ПРИНЦИП МАКСИМАЛЬНОГО ПРАВДОПОДОБИЯ

Принцип максимального правдоподобия состоит в том, что в качестве «наиболее правдоподобного» значения параметра берут значение Θ , максимизирующее вероятность получить при n опытах имеющуюся выборку $X = (x_1, \dots, x_n)$.

При оценке параметров в R могут использоваться функции `fitdistr()` из пакета MASS и `fitdist()` из пакета `fitdistrplus`.

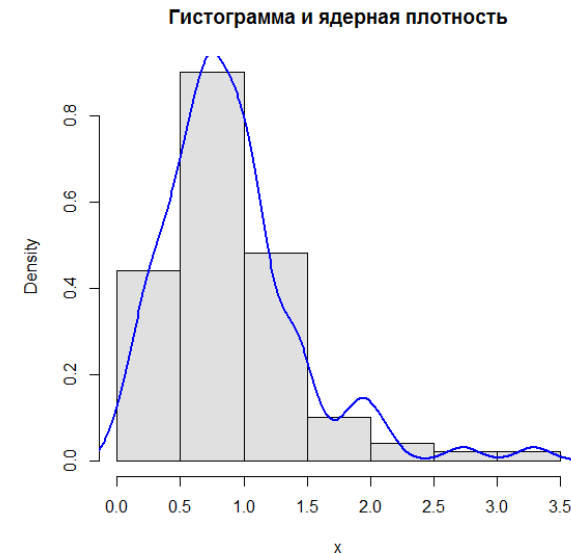
Пример (непрерывное распределение): рассмотрим имитацию случайной выборки из распределения Вейбулла

```
set.seed(1946)
x = sort(rweibull( 100, 2, (1 + 1.21*rbinom(100, 1, 0.05)) ))
```

4.4. Подгонка статистического распределения

График выборочной гистограммы и ядерной функции плотности распределения

```
hist(x, freq = FALSE, breaks=8,  
+ col="grey88", main = "Гистограмма и  
+ ядерная плотность")  
lines(density(x), lwd = 2, col="blue")
```



Рассмотрим в качестве моделей-претендентов три закона распределения:
нормальное, лог-нормальное и распределение Вейбулла.

Процедура подгонки эмпирического распределения состоит из трех шагов:

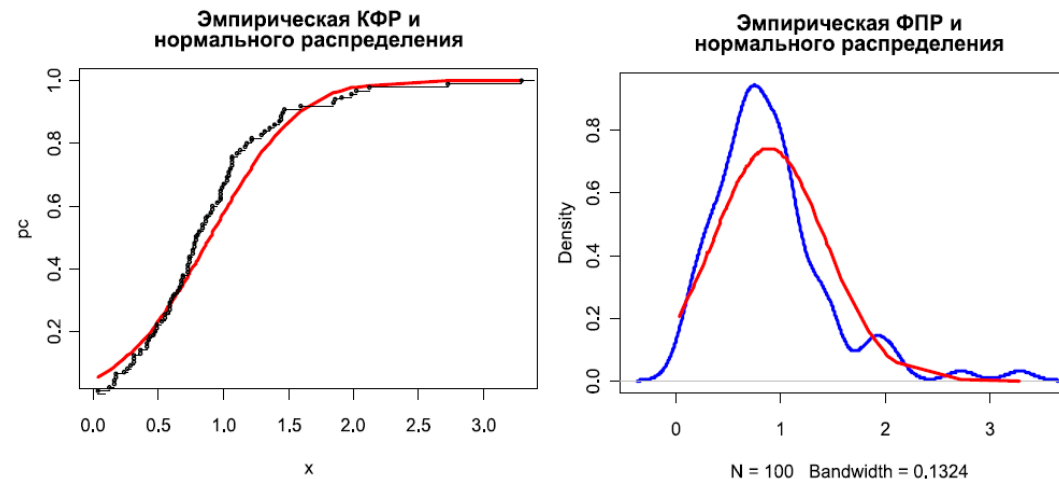
- оценка параметров распределения на основе метода максимального правдоподобия;
- проверка гипотезы о согласии эмпирического и теоретического распределений с использованием критерия Колмогорова-Смирнова;
- вывод графика (для удобства сопоставления показаны на одном рисунке).

4.4. Подгонка статистического распределения

График выборочной гистограммы и ядерной функции плотности распределения

```
##  оценка параметров нормального распределения
(dof = fitdistr(x,"normal"))
ep1=dof$estimate[1]; ep2=dof$estimate[2]
      mean      sd
0.89502201 0.53760487
(0.05376049) (0.03801440)

ks.test(x,pnorm, mean=ep1,sd=ep2)
      One-sample Kolmogorov-Smirnov test
data:  x
D = 0.1342, p-value = 0.05463
alternative hypothesis: two-sided
```

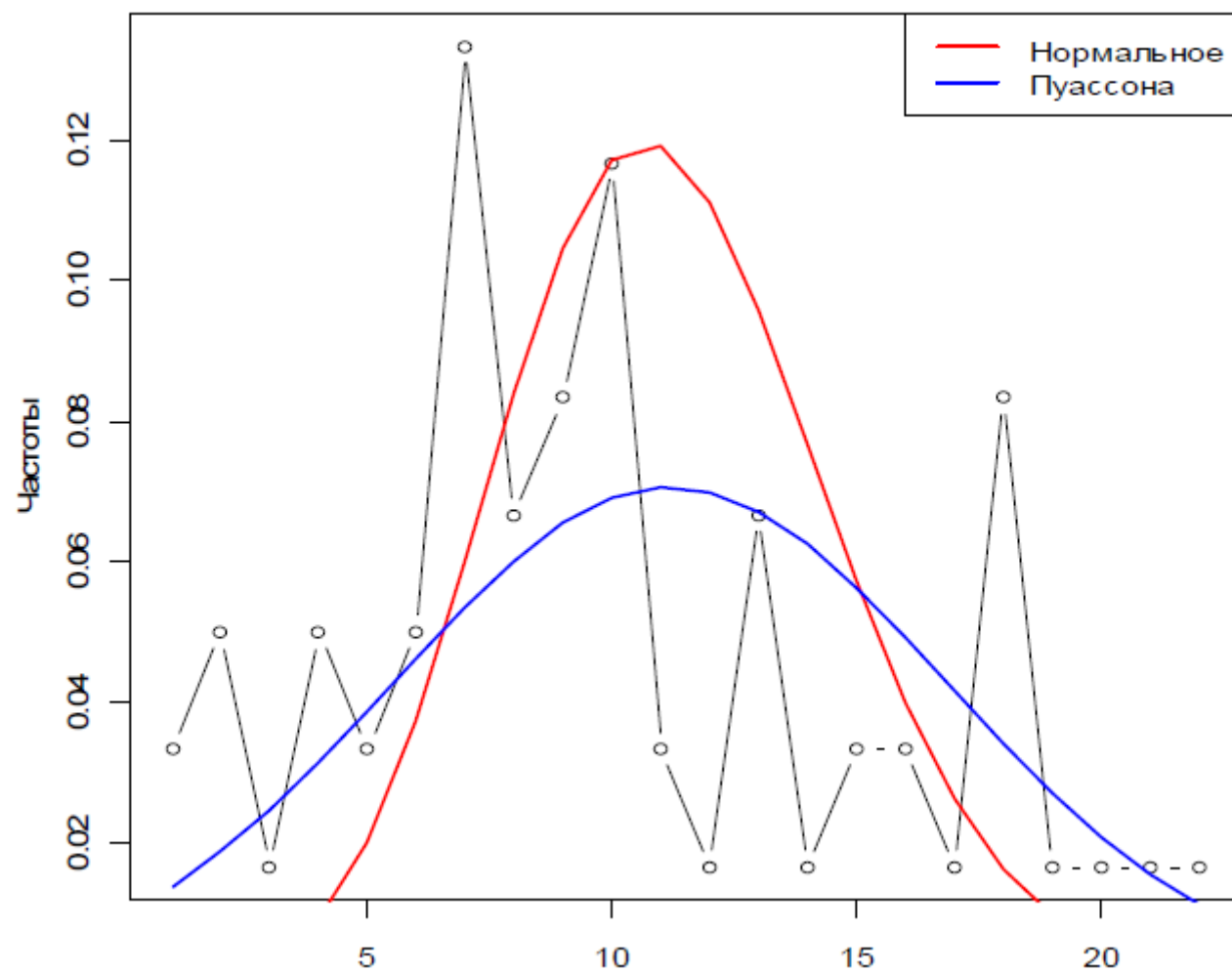


4.4. Подгонка статистического распределения

Пример (дискретное распределение): из реки было сделано 60 проб и подсчитывалось число обнаруженных видов донных организмов. Это число варьирует от 2 до 30 при среднем $x = 11.2$. Какое распределение является лучшим с формально-статистической точки зрения: Пуассона с $\lambda = 11.2$ или нормальное?

```
x <- c(12,20,19,19,18,10,19,30,16,10,8,11,10,11,16,3,7,6,5,11,
8,14,9,8,10,11,14,17,2,7,17,19,9,15,9,8,4,8,11,8,5,3,10,
14,22,11,8,7,3,5,8,11,14,2,13,9,12,6,19,21)
# Оценка параметров распределений нормального и Пуассона
n = length(x); p1 = mean(x) ; p2 = sqrt(var(x)*(n-1)/n)
# Создание векторов эмпирических и теоретических частот
pr_obs <- as.vector(table(x)/n) ; nr <- length(pr_obs)
pr_norm <- dnorm(1:nr, p1, p2) # Частоты нормального распр.
pr_pois <- dpois(1:nr, p1)      # Частоты распр. Пуассона
plot(pr_obs, type="b", ylab = "Частоты")
  lines(1:nr, pr_pois , col="red", lwd=2)
  lines(1:nr, pr_norm, col="blue", lwd=2)
  legend("topright", legend = c("Нормальное", "Пуассона"),
        lwd=2, col=c("red","blue"))
# Сравнение качества подгонки распределений
# Среднее абсолютное отклонение
c(sum(abs(pr_obs-pr_norm))/nr, sum(abs(pr_obs-pr_pois))/nr)
[1] 0.02314994 0.03176255
# Средняя квадратичная ошибка
c(sum((pr_obs-pr_norm)^2)/nr, sum((pr_obs-pr_pois)^2)/nr)
[1] 0.0009595203 0.0017446052
# Критерий согласия Колмогорова-Смирнова
c(ks.test(pr_obs, pr_norm)$statistic,
  ks.test(pr_obs, pr_pois)$statistic)
[1] 0.2272727 0.4090909
```

4.4. Подгонка статистического распределения



4.5. Проверка распределения на нормальность

СПОСОБЫ:

1. Графический (с помощью гистограмм, графиков квантилей и т.п.)
2. Формальные тесты (тест Шапиро-Уилка, тест Андерсона-Дарлинга, тест Крамера фон Мизеса, тест Колмогорова-Смирнова в модификации Лиллиефорса, тест Шапиро-Франсия)

```
# Тесты на нормальность
shapiro.test(x)
      Shapiro-Wilk normality test
data:  x
W = 0.8986, p-value = 1.219e-06

library(nortest)
ad.test(x)
      Anderson-Darling normality test
data:  x
A = 2.0895, p-value = 2.382e-05
cvm.test(x)
      Cramer-von Mises normality test
data:  x
W = 0.3369, p-value = 0.0001219
lillie.test(x)
      Lilliefors (Kolmogorov-Smirnov) normality test
data:  x
D = 0.1348, p-value = 0.0001225
sf.test(x)
      Shapiro-Francia normality test
data:  x
W = 0.8936, p-value = 3.617e-06
```


Список литературы

Кабаков Р. К. (2014) R в действии. Анализ и визуализация данных на языке R Издательство: ДМК Пресс, 580 с.

Numerical Analysis // Computational Economics Practice by Stefan Feuerrigel (Freiberg Universitaet)

Мастицкий С. Э., Шитиков В. К. (2014) Статистический анализ и визуализация данных с помощью R. - Электронная книга, 400 с

Зарядов И. С. (2010) Статистический пакет R: теория вероятностей и математическая статистика. Москва: Изд-во РУДНБ, 141 с.