Язык R
и его применение в биоинформатике

Анастасия Александровна Жарикова

Дмитрий Дмитриевич Пензар

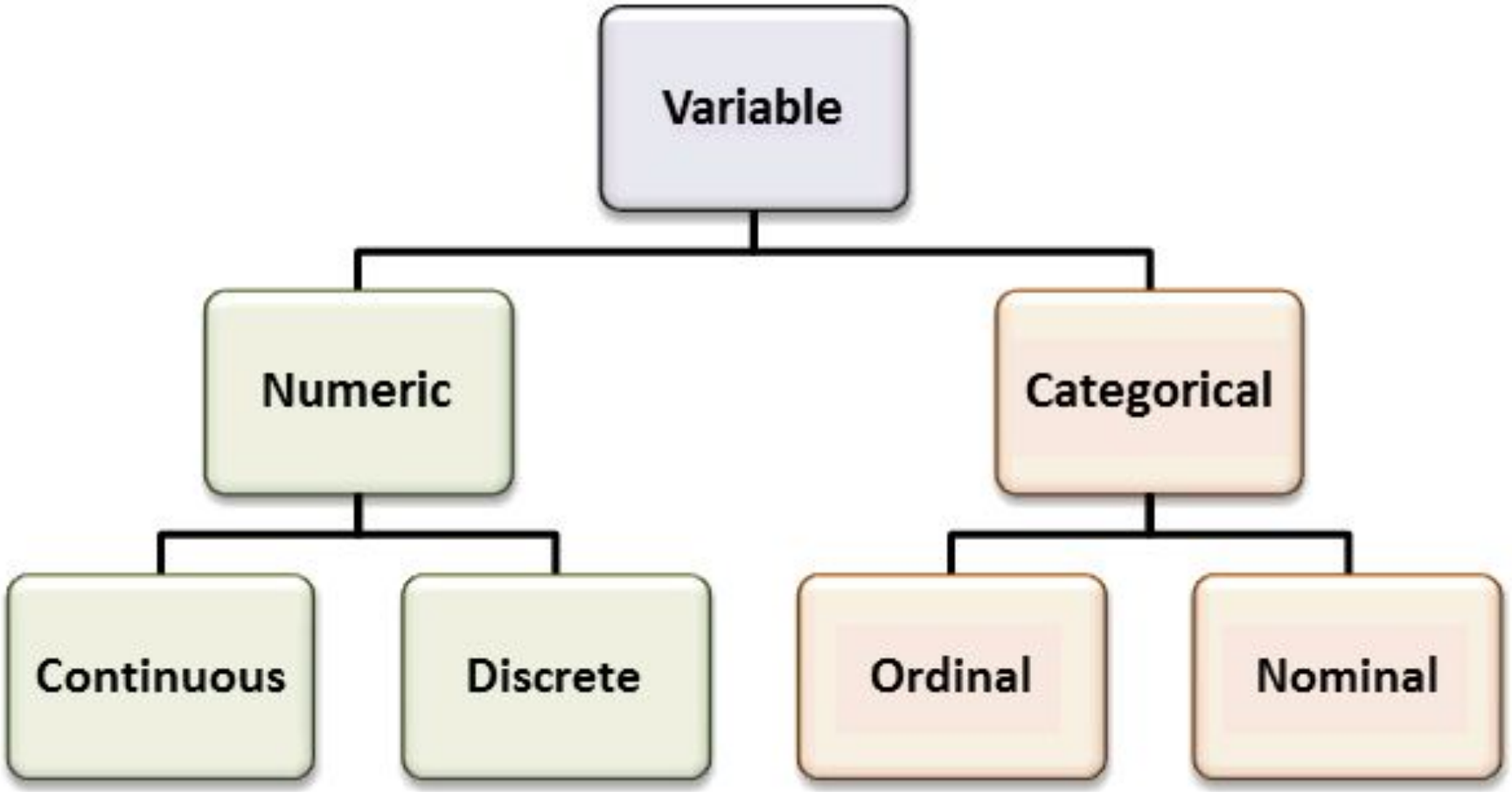
14\15 сентября 2020

Категориальные переменные

Какие еще бывают?

Категориальные переменные

Какие еще бывают?



Факторы

```
f <- factor(c("yes", "yes", "no", "yes", "no"))
f
```

```
## [1] yes yes no  yes no
## Levels: no yes
```

```
levels(f)
```

```
## [1] "no"  "yes"
```

Факторы

```
levels(f) <- c(levels(f), "maybe")
table(f)
```

```
## f
##    no   yes maybe
##     2     3     0
```

Факторы

Упорядочивание уровней фактора

```
f <- factor(c("yes", "yes", "no", "yes","no"), levels = c("yes", "no"))
f
```

```
## [1] yes yes no  yes no
## Levels: yes no
```

Факторы

```
a=read.table('in_data.tab',header=T)
head(a)
```

```
##   height weight gender
## 1    132     48   male
## 2    151     49   male
## 3    162     66 female
## 4    139     53 female
## 5    166     67   male
## 6    147     52 female
```

```
str(a)
```

```
## 'data.frame':    7 obs. of  3 variables:
##  $ height: int  132 151 162 139 166 147 122
##  $ weight: int  48 49 66 53 67 52 40
##  $ gender: Factor w/ 2 levels "female","male": 2 2 1 1 2 1 2
```

Факторы

Зачем такое поведение? (пытаемся угадать мысли автора)

```
library(rbenchmark)
string_vec <- sample(c("yes", "no"), size=10000, replace = TRUE)
factor_vec <- as.factor(string_vec)
benchmark("string"={sum(string_vec == "yes")},
          "factor_vec" = {sum(factor_vec == "yes")},
           replications = 1000,
           columns = c("test", "replications",
                       "relative"))
```

```
##         test replications relative
## 2 factor_vec         1000     1.00
## 1     string         1000     2.37
```

Факторы

```
a=read.table('in_data.tab',header=T,stringsAsFactors = F)
head(a)
```

```
##   height weight gender
## 1    132     48   male
## 2    151     49   male
## 3    162     66 female
## 4    139     53 female
## 5    166     67   male
## 6    147     52 female
```

```
str(a)
```

```
## 'data.frame':    7 obs. of  3 variables:
##  $ height: int  132 151 162 139 166 147 122
##  $ weight: int  48 49 66 53 67 52 40
##  $ gender: chr  "male" "male" "female" "female" ...
```

Матрицы

Внешне похожи на data.frame

Вектор с атрибутом dim

Все элементы одного типа

Математические операции работают быстрее

Способы создания матрицы

```
m <- matrix(c(1:6), nrow=2, ncol=3)
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
m <- matrix(c(1:6), nrow=2, ncol=3,byrow=T)
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

Способы создания матрицы

```
as.matrix(1:3)
```

```
##      [,1]
## [1,]    1
## [2,]    2
## [3,]    3
```

Способы создания матрицы

```
v <- 1:6
dim(v)
```

```
## NULL
```

```
dim(v) <- c(2,3)
v
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
is.matrix(v)
```

```
## [1] TRUE
```

Способы создания матрицы

```
a <- c(1,2,3)
b <- c(7,8,9)
cbind(a,b)
```

```
##      a b
## [1,] 1 7
## [2,] 2 8
## [3,] 3 9
```

```
rbind(a,b)
```

```
##   [,1] [,2] [,3]
## a    1    2    3
## b    7    8    9
```

Матрицы. Срезы

```
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

```
m[2,3]
```

```
## [1] 6
```

```
m[2,]
```

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

```
m[,3]
```

```
## [1] 3 6
```

Матрицы. Изменение значений

```
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

```
m[2,3] = 1
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    1
```

Создать пустую матрицу

```
m <- matrix(nrow=2, ncol=3)
m
```

```
##      [,1] [,2] [,3]
## [1,]   NA   NA   NA
## [2,]   NA   NA   NA
```

```
dim(m)
```

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

Матрицы

```
m
```

```
##      [,1] [,2] [,3]
## [1,]   NA   NA   NA
## [2,]   NA   NA   NA
```

```
rownames(m) <- c("r1","r2")
colnames(m) <- c("c1","c2","c3")
m
```

```
##    c1 c2 c3
## r1 NA NA NA
## r2 NA NA NA
```

Из data.frame в матрицу

```
df=data.frame(name=paste('gene',1:5,sep='_'),
              par1=1:5,
              par2=11:15,
              par3=21:25,stringsAsFactors = FALSE)
df
```

```
##     name par1 par2 par3
## 1 gene_1    1   11   21
## 2 gene_2    2   12   22
## 3 gene_3    3   13   23
## 4 gene_4    4   14   24
## 5 gene_5    5   15   25
```

```
str(df)
```

```
## 'data.frame':    5 obs. of  4 variables:
##  $ name: chr  "gene_1" "gene_2" "gene_3" "gene_4" ...
##  $ par1: int  1 2 3 4 5
##  $ par2: int  11 12 13 14 15
##  $ par3: int  21 22 23 24 25
```

Из data.frame в матрицу

```
row.names(df)=df$name
df$name=NULL
df.m=as.matrix(df)
df.m
```

```
##         par1 par2 par3
## gene_1     1   11   21
## gene_2     2   12   22
## gene_3     3   13   23
## gene_4     4   14   24
## gene_5     5   15   25
```

Из data.frame в матрицу

```
df=data.frame(name=paste('gene',c(1,1,3,4,5),sep='_'),
              par1=1:5,
              par2=11:15,
              par3=21:25,stringsAsFactors = FALSE)
df
```

```
##     name par1 par2 par3
## 1 gene_1    1   11   21
## 2 gene_1    2   12   22
## 3 gene_3    3   13   23
## 4 gene_4    4   14   24
## 5 gene_5    5   15   25
```

```
str(df)
```

```
## 'data.frame':    5 obs. of  4 variables:
##  $ name: chr  "gene_1" "gene_1" "gene_3" "gene_4" ...
##  $ par1: int  1 2 3 4 5
##  $ par2: int  11 12 13 14 15
##  $ par3: int  21 22 23 24 25
```

Из data.frame в матрицу

```
row.names(df)=df$name
```

```
## Warning: non-unique value when setting 'row.names': 'gene_1'
```

```
## Error in `.rowNamesDF<-`(x, value = value): duplicate 'row.names' are not allowed
```

Списки

Можно хранить данные разных типов

```
L <- list("A", c(1,2), 30)
L
```

```
## [[1]]
## [1] "A"
##
## [[2]]
## [1] 1 2
##
## [[3]]
## [1] 30
```

Списки

```
L1 <- list (L, 40)
L1
```

```
## [[1]]
## [[1]][[1]]
## [1] "A"
##
## [[1]][[2]]
## [1] 1 2
##
## [[1]][[3]]
## [1] 30
##
##
## [[2]]
## [1] 40
```

Списки

```
L [[4]] = 'new_element'
L
```

```
## [[1]]
## [1] "A"
##
## [[2]]
## [1] 1 2
##
## [[3]]
## [1] 30
##
## [[4]]
## [1] "new_element"
```

Списки

```
a <- list()
b <- c(a, 5)
print(b)
```

```
## [[1]]
## [1] 5
```

```
d <- list(4, 10)
print(c(b, d))
```

```
## [[1]]
## [1] 5
##
## [[2]]
## [1] 4
##
## [[3]]
## [1] 10
```

Списки

```
L [3]
```

```
## [[1]]
## [1] 30
```

```
L [[3]]
```

```
## [1] 30
```

Списки

```
L1 [[1]]
```

```
## [[1]]
## [1] "A"
##
## [[2]]
## [1] 1 2
##
## [[3]]
## [1] 30
```

```
L1 [[1]] [[2]]
```

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

Списки

```
L <- list (10,20)
L$abc <- 123
L
```

```
## [[1]]
## [1] 10
##
## [[2]]
## [1] 20
##
## $abc
## [1] 123
```

```
names(L)
```

```
## [1] ""    ""    "abc"
```

Списки

```
L[[3]]
```

```
## [1] 123
```

```
L$abc
```

```
## [1] 123
```

```
L[["abc"]]
```

```
## [1] 123
```

names

```
lst <- list(a = 1:10, b = "49", c = list(d=5, s=10), "New York"=5)
print(names(lst))
```

```
## [1] "a"        "b"        "c"        "New York"
```

```
names(lst)[1] <- "vec"
print(names(lst))
```

```
## [1] "vec"      "b"        "c"        "New York"
```

List - не словарь

```
library(rbenchmark)
size <- 100000
lst <- as.list(runif(size, 0, 1))
names(lst) <- paste("n", 1:size, sep = "")
names(lst)[1] <- "first"
names(lst)[length(lst)] <- "last"
benchmark(
"index_first"={
     a <- lst[[1]]
},
"name_first" = {
       a <- lst[["first"]]
},
"index_last"={
    a <- lst[length(lst)]
},
"name_last" = {
    a <- lst["last"]
},
  replications = 1000,
    columns = c("test", "replications",
                        "relative")
)
```

```
##          test replications relative
## 1 index_first         1000        1
## 3  index_last         1000        2
## 2  name_first         1000        2
## 4   name_last         1000     1477
```

Циклы

Циклы в R медленные!

for

```r
for (year in c(2010,2011,2012,2013,2014,2015)){
  print(paste("The year is", year))
}
```

```
## [1] "The year is 2010"
## [1] "The year is 2011"
## [1] "The year is 2012"
## [1] "The year is 2013"
## [1] "The year is 2014"
## [1] "The year is 2015"
```

for

```
mt <- mtcars
head(mt,2)
```

```
##               mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4      21   6  160 110  3.9 2.620 16.46  0  1    4    4
## Mazda RX4 Wag  21   6  160 110  3.9 2.875 17.02  0  1    4    4
```

```
for (i in 1:nrow(mt)){
  mt$NEW[i] <- 2^i
}
head(mt,4)
```

```
##                mpg cyl disp  hp drat    wt  qsec vs am gear carb NEW
## Mazda RX4     21.0   6  160 110 3.90 2.620 16.46  0  1    4    4   2
## Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4   4
## Datsun 710    22.8   4  108  93 3.85 2.320 18.61  1  1    4    1   8
## Hornet 4 Drive 21.4  6  258 110 3.08 3.215 19.44  1  0    3    1  16
```

Деление

```
5/2
```

```
## [1] 2.5
```

```
5%%2
```

```
## [1] 1
```

for + if

```
mt <- mtcars
for (i in 1:nrow(mt)){
  if ((i %% 2) == 0){
    mt$NEW[i] = i^2
    mt$type[i] = 'even'
  }
  else {
    mt$NEW[i] = i^3
    mt$type[i] = 'odd'
  }
}
head(mt)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb NEW
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4   1
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4   4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1  27
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1  16
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2 125
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1  36
##                   type
## Mazda RX4          odd
## Mazda RX4 Wag     even
## Datsun 710         odd
## Hornet 4 Drive    even
## Hornet Sportabout  odd
## Valiant           even
```

ifelse

`ifelse` returns a value with the same shape as `test` which is filled with elements selected from either `yes` or `no` depending on whether the element of `test` is `TRUE` or `FALSE`.

Usage

```
ifelse(test, yes, no)
```

Arguments

`test`	an object which can be coerced to logical mode.
`yes`	return values for true elements of `test`.
`no`	return values for false elements of `test`.

ifelse

```
x <- sample(1:1000, 1000, replace=T)
y <- ifelse(x %% 2 == 1, x ** 2, x ** 3)
```

ifelse

```
library(rbenchmark)
x <- rnorm(10000)
benchmark("ifbranch_gg"={y <- c()
    for (i in x){
        if (i > 0){
            y <- c(y, i)
        }else{
            y <- c(y, 0)
        }
    }
}, "ifbranch"={
    y <- double(length = length(x))
    for (i in 1:length(x)){
        if (x[i] > 0){
            y[i] <- x[i]
        }
    }
},
    "ifelse" = {
        y <- ifelse(x > 0, x, 0)
},
    replications = 10,
    columns = c("test", "replications",
                        "relative"))
```

```
##          test replications relative
## 2    ifbranch           10   12.333
## 1 ifbranch_gg           10  689.667
## 3      ifelse           10    1.000
```

Почему такой медленный первый вариант?

```
x <- 1:10
y <- x
y[5] <- 10
print(x)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
print(y)
```

```
##  [1]  1  2  3  4 10  6  7  8  9 10
```

Базовые объекты в R - неизменяемы. Любое изменение создает копию объекта

```
a <- data.frame(a=1:5, b=1:5)
b <- a
b$a[3] <- 20
print(a)
```

```
##   a b
## 1 1 1
## 2 2 2
## 3 3 3
## 4 4 4
## 5 5 5
```

```
print(b)
```

```
##    a b
## 1  1 1
## 2  2 2
## 3 20 3
## 4  4 4
## 5  5 5
```

Неизменяемость объектов дает ряд преимуществ и теоретически может быть очень эффективно реализована.
Однако она создает и ряд проблем, из-за которых делать полностью невозможным изменить объекты - спорное решение

Языки

C разрабатывался двумя трезвыми высококвалифицированными программистами, чтобы поиграть в игру Asteroids на новом компьютере компании, где они работали

Python разрабатывался высококвалифицированным программистом в Рождество в компании с кружкой глитвейна

C++, Rust и многие другие языки (не JavaScript) разрабатывались опытными программистами с мощной опорой на опыт языков до этого

R разрабатывался математиками, которые предпочли проигнорировать весь опыт программистов до них...

for + if + next

```
x <- 1:5
for (val in x) {
if (val == 3){
next
}
print(val)
}
```

```
## [1] 1
## [1] 2
## [1] 4
## [1] 5
```

Как заменить?

Как заменить?

```
library(rbenchmark)

benchmark("ifnext"={
    x <- sample(1:1000, 10000, rep=T)
    for (v in x){
        if (v  == 5){
            next
        }
    }
},
    "which" = {
        x <- sample(1:1000, 10000, rep=T)
        pos <- which(x != 5)
        y <- x[pos]
}
,
    replications = 100,
    columns = c("test", "replications",
                        "relative"))
```

```
##     test replications relative
## 1 ifnext          100    6.727
## 2  which          100    1.000
```

for + if + break

```
x <- 1:5
for (val in x) {
if (val == 3){
break
}
print(val)
}
```

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

Как заменить?

Как заменить?

```
library(rbenchmark)
x <- rnorm(10000)
benchmark("ifbreak"={
    x <- sample(1:1000, 10000, rep=T)
    for (i in 1:length(x)){
        if (x[i]  == 5){
            break
        }
    }
    pos <- ifelse(x[i] == 5, i, -1)
},
    "which" = {
        x <- sample(1:1000, 10000, rep=T)
        pos <- which(x == 5)
        pos <- ifelse(length(pos) > 0, pos[1], -1)
},
   "match" = {
       x <- sample(1:1000, 10000, rep=T)
       pos <- match(5, x, nomatch=-1)
},
    replications = 100,
    columns = c("test", "replications",
                        "relative"))
```

```
##      test replications relative
## 1 ifbreak          100    8.067
## 3   match          100    1.000
## 2   which          100    1.033
```

Как узнать крайнее правое вхождение?

Как узнать крайнее правое вхождение?

```
library(rbenchmark)
x <- rnorm(10000)
benchmark("ifbreak"={
    x <- sample(1:1000, 10000, rep=T)
    for (i in length(x):1){
        if (x[i]  == 5){
            break
        }
    }
    pos <- ifelse(x[i] == 5, i, -1)
},
    "which" = {
        x <- sample(1:1000, 10000, rep=T)
        pos <- which(x == 5)
        pos <- ifelse(length(pos) > 0, pos[length(pos)], -1)
},
   "match" = {
       x <- sample(1:1000, 10000, rep=T)
       pos <- match(5, rev(x), nomatch=-1)
       pos <- ifelse(pos == -1, -1, length(x) - pos + 1)
},
    replications = 100,
    columns = c("test", "replications",
                        "relative"))
```

```
##      test replications relative
## 1 ifbreak          100    7.900
## 3   match          100    1.133
## 2   which          100    1.000
```

while

```
i <- 1
while (i < 6) {
print(i)
i = i+1
}
```

Иногда заменить ничем нельзя. Надо смириться

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

repeat + if + break

```
x <- 1
repeat {
print(x)
x = x+1
if (x == 6){
break
}
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

repeat + if + break

```
x <- 1
repeat {
print(x)
x = x+1
if (x == 6){
break
}
}
```

Не используйте!

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

for

```
mt <- mtcars
head(mt,2)
```

```
##               mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4      21   6  160 110  3.9 2.620 16.46  0  1    4    4
## Mazda RX4 Wag  21   6  160 110  3.9 2.875 17.02  0  1    4    4
```

```
for (i in 1:nrow(mt)){
  mt$NEW[i] <- 2^i
}
head(mt,4)
```

```
##                mpg cyl disp  hp drat    wt  qsec vs am gear carb NEW
## Mazda RX4     21.0   6  160 110 3.90 2.620 16.46  0  1    4    4   2
## Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4   4
## Datsun 710    22.8   4  108  93 3.85 2.320 18.61  1  1    4    1   8
## Hornet 4 Drive 21.4  6  258 110 3.08 3.215 19.44  1  0    3    1  16
```

Случайная матрица

```
set.seed(123)
mt <- matrix(sample(1:5,10000,replace = T),ncol=10)
dim(mt)
```

```
## [1] 1000   10
```

```
head(mt)
```

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

Случайная матрица

```
mt[1:5,1:5]
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    2    2    1    2    2
## [2,]    4    3    1    5    5
## [3,]    3    1    1    2    3
## [4,]    5    5    3    4    5
## [5,]    5    5    3    1    2
```

Случайная матрица

```
colnames(mt) <- paste("D",1:ncol(mt),sep='_')
rownames(mt) <- paste("Stud",1:nrow(mt),sep='.')
head(mt)
```

```
##         D_1 D_2 D_3 D_4 D_5 D_6 D_7 D_8 D_9 D_10
## Stud.1   2   2   1   2   2   2   3   5   3    4
## Stud.2   4   3   1   5   5   5   2   3   3    5
## Stud.3   3   1   1   2   3   4   2   1   4    2
## Stud.4   5   5   3   4   5   4   3   5   2    3
## Stud.5   5   5   3   1   2   2   1   1   4    5
## Stud.6   1   3   4   4   1   1   4   5   5    1
```

Подсчитать среднее по столбцам

```
x <- rep(NA,ncol(mt))
length(x)
```

```
## [1] 10
```

```
x
```

```
##  [1] NA NA NA NA NA NA NA NA NA NA
```

Подсчитать среднее по столбцам

```
for (i in 1:ncol(mt)){
  x[i] <- mean(mt[,i])
}
x
```

```
##  [1] 2.989 2.999 2.994 3.033 2.951 3.040 2.990 2.985 2.940 3.009
```

Подсчитать сумму по строкам

```
y <- rep(NA,nrow(mt))
length(y)
```

```
## [1] 1000
```

Подсчитать сумму по строкам

```
for (i in 1:nrow(mt)){
  y[i] <- sum(mt[i,])
}
y[1:30]
```

```
##  [1] 26 36 23 39 29 29 27 25 31 29 38 37 34 33 28 33 30 29 33 28 40 39 33
## [24] 24 30 33 33 35 27 34
```

apply

```
x.2 <- apply(mt, 2, mean)
y.2 <- apply(mt, 1, sum)
```

```
x
```

```
##  [1] 2.989 2.999 2.994 3.033 2.951 3.040 2.990 2.985 2.940 3.009
```

```
x.2
```

```
##   D_1   D_2   D_3   D_4   D_5   D_6   D_7   D_8   D_9  D_10
## 2.989 2.999 2.994 3.033 2.951 3.040 2.990 2.985 2.940 3.009
```

apply

```
x.2 <- apply(mt, 2, mean)
y.2 <- apply(mt, 1, sum)
```

```
y[1:30]
```

```
##  [1] 26 36 23 39 29 29 27 25 31 29 38 37 34 33 28 33 30 29 33 28 40 39 33
## [24] 24 30 33 33 35 27 34
```

```
y.2[1:30]
```

```
##  Stud.1  Stud.2  Stud.3  Stud.4  Stud.5  Stud.6  Stud.7  Stud.8  Stud.9
##      26      36      23      39      29      29      27      25      31
## Stud.10 Stud.11 Stud.12 Stud.13 Stud.14 Stud.15 Stud.16 Stud.17 Stud.18
##      29      38      37      34      33      28      33      30      29
## Stud.19 Stud.20 Stud.21 Stud.22 Stud.23 Stud.24 Stud.25 Stud.26 Stud.27
##      33      28      40      39      33      24      30      33      33
## Stud.28 Stud.29 Stud.30
##      35      27      34
```

apply

```
library(rbenchmark)
mt <- matrix(sample(1:5, 1000000, replace=T), ncol=1000)
df_mt <- data.frame(mt)
benchmark("for"={
    y <- rep(NA, nrow(mt))
    for (i in 1:nrow(mt)){
        y[1] <- mean(mt[i,])
    }
},
    "apply1" = {
       mean1 <- apply(mt, 1, mean)
},
"apply2" = {
       mean2 <- apply(mt, 2, mean)
},
    "apply1_df" = {
       mean1 <- apply(df_mt, 1, mean)
},
"apply2_df" = {
       mean2 <- apply(df_mt, 2, mean)
},
  replications = 10,
    columns = c("test", "replications",
                        "relative")
)
```

```
##        test replications relative
## 2    apply1           10    1.049
## 4 apply1_df           10    2.260
## 3    apply2           10    1.000
## 5 apply2_df           10    1.480
## 1       for           10    1.252
```

Список

```r
x <- list(a = 1, b = 1:3, c = 10:100)
x
```

```
## $a
## [1] 1
##
## $b
## [1] 1 2 3
##
## $c
##  [1]  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26
## [18]  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43
## [35]  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
## [52]  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77
## [69]  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94
## [86]  95  96  97  98  99 100
```

lapply

Возвращает список

```
lapply(x, FUN = length)
```

```
## $a
## [1] 1
##
## $b
## [1] 3
##
## $c
## [1] 91
```

lapply

```
lapply(x, FUN = sum)
```

```
## $a
## [1] 1
##
## $b
## [1] 6
##
## $c
## [1] 5005
```

sapply

Пытается вернуть вектор. Если не получается - возвращает list

```
sapply(x, FUN = length)
```

```
##  a  b  c
##  1  3 91
```

```
sapply(x, FUN = sum)
```

```
##    a    b    c
##    1    6 5005
```

```
sapply(1:3, FUN=list)
```

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

vapply

Возвращает набор из значений, соответствующих определенному шаблону.
Если не получается - вылетит ошибка

```
vapply(1:10, FUN=sum, FUN.VALUE = double(1))
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
vapply(1:10, FUN=list, FUN.VALUE = double(1))
```

```
## Error in vapply(1:10, FUN = list, FUN.VALUE = double(
##  but FUN(X[[1]]) result is type 'list'
```

```
vapply(1:3, FUN=list, FUN.VALUE = list(1))
```

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

vapply

```
vapply(1:3, FUN=list, FUN.VALUE = list(1, 2))
```

```
## Error in vapply(1:3, FUN = list, FUN.VALUE = list(1, 2)): values must be length 2,
##  but FUN(X[[1]]) result is length 1
```

```
vapply(1:10, FUN=function(x){1:10}, FUN.VALUE = double(10))
```

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

mapply

```
x <- 1:100
y <- -1:-100
z <- sample(0:1, 100, replace = T)
mapply(FUN = sum, x, y, z)
```

```
##   [1] 0 1 0 1 1 0 0 0 0 0 0 1 0 0 1 1 0 1 1 0 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 0 0
##  [38] 0 0 1 1 0 0 0 1 0 0 0 1 1 1 0 0 1 0 0 1 0 0 1 1 0 1 1 0 0 0 1 1 1 0 1 1 0
##  [75] 1 0 1 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 1 1 0 1 1 1 0
```

f 2015

Jonathan Boothe

[redacted] is easy. It's like riding a bike and the bike is on fire and the ground is on fire and everything's on fire because you're in hell."