

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

Типы объектов в R и манипуляция с ними

Г. Мороз

Что такое R?

- среда для статистического анализа, обработки и визуализации данных



язык программирования

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

Что такое R?

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

- среда для статистического анализа, обработки и визуализации данных



язык программирования (хотя многие скептически...)

- скачать R
- скачать RStudio
- для любителей командной строки есть **littler**
- R in Notepad ++, Vim-R-Tmux, RKWard...

Что такое R?

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции






- среда для статистического анализа, обработки и визуализации данных



язык программирования (хотя многие скептически...)

- скачать R
- скачать RStudio
- для любителей командной строки есть **littler**
- R in Notepad ++, Vim-R-Tmux, RKWard...

Не обязательно в R:

-  Matlab
-  Octave
-  STATISTICA (и русский блог об этой программе)
-  SPSS
-  Julia
- и др.

презентация доступна: <http://1drv.ms/1TDPR6m>

Преимущества R

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

- бесплатный
- с открытым исходным кодом
- идет под многими операционными системами
- с кучей пакетов (< 6500), к которым написан хэлп с примерами
- реализована вся статистика, визуализация и обработка данных
- понимает уйму форматов
- пишет во множество форматов
- можно проиллюстрировать все свои манипуляции с данными в R Markdown
- большое комьюнити \Rightarrow быстро ответят на вопрос, на многое уже отвечено, много книжек

Недостатки

- руководство не всегда на высшем уровне
- во многих функциях не уделяется большое внимание расходу памяти
- работа с некоторыми типами данных хуже (например с символами)

Посмотрите статью [Comparing Python and R for Data Science](http://1drv.ms/1TDPR6m)

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

Калькулятор

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

$(4+1)*7+9-4$

$57 \% \% 43$

$(35+7)/-\ln f$

2^2^3 или $2^{**2^{**3}}$

$\text{abs}(-5)$

$\log(100, 10)$

$\log_2(64), \log_{10}(100)$

$\text{sum}(1:6, 7, 10:8)$

$\text{prod}(1:6, 7, 10:8)$

$\text{factorial}(8)$

$\text{choose}(7, 3)$

$\# (4 + 1) \times 7 + 9 - 4$

$\#$ остаток от деления

$\# \frac{35+7}{-\infty}$

$\# 2^{2^3}$

$\# | - 5 |$

$\# \log_{10}(100)$

$\# \log_2(64), \log_{10}(100)$

$\# \sum_{n=1}^{10} n$

$\# \prod_{n=1}^{10} n$

$\# 8!$

$\# \binom{3}{7}$

презентация доступна: <http://1drv.ms/1TDPR6m>

Атомарные и базовые объекты

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

○ атомарные объекты

- numeric — число
- integer — целое число
- complex — комплексное число
- character — символ
- logical (TRUE, FALSE) — логический оператор
- NA (not available) — пропущенное значение
- NULL
- NaN (not a number)

по умолчанию

5L

7+2i

○ базовые объекты

- vector — вектор **объекты только одного класса**
- matrix — вектор с двумя измерениями (матрица)
- array — вектор с двумя и более измерений (многомер. матрица)
- list — список **может содержать объекты разного класса**
- dataframe — датафрейм

○ комментирование

- #

- В RStudio  +  +  или  +  + 

Векторы: базовое

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

```
a <- 3
```

 + 

```
a
```

смотрим значение переменной

```
a + 1
```

```
a
```

значение не изменилось

```
a + 3 -> a
```

```
a
```

```
b <- "ку-ку"
```

```
c <- 1:6
```

```
d <- c(1:3, 2, 2, 3)
```

```
e <- c(100:1)*2
```

```
f <- pi
```

```
g <- c(Маша = 1, Ваня = 2, Саша = 2, Аня = 2) # вектор с именами
```

Векторы: последовательности, генераторы чисел

Фрагменты арифметических последовательностей можно создавать при помощи функции `seq()`

```
3:23                                     seq(3, 23)
(3:23)*3   seq(9, 69, by = 3)   seq(9, 69, 3)   # от 9 до 69 по 3
seq(3, 2, length=100)           # сто элементов от 3 до 2
```

Можно создавать векторы из повторяющихся элементов:

```
rep(5, 2)                                     # 2 раза 5
rep(33:22, 4)                                 # 4 раза 33:22
rep(33:22, each = 4)                          # 4 раза каждый элемент 33:22
```

Можно включить генератор случайных чисел:

```
runif(1, 11, 37)          # случайное число в промежутке (11, 37)
runif(5, 11, 37)          # пять случайных чисел в промежутке (11, 37)
sample(11:37, 6)          # шесть случайных целых чисел из 11:37
sample(11:37, 6, replace=TRUE) # можно повторяться
sample(c("a", "u", "o", "y"), 6, replace=TRUE)
```

Матрицы

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

- `A <- matrix(c(2, 4, 3, 1, 5, 7),
ncol=3)`

```
  [,1] [,2] [,3]  
[1,]  2   3   5  
[2,]  4   1   7
```

данные
количество колонок

- `B <- matrix(c(2, 4, 3, 1, 5, 7),
nrow=2)`

```
  [,1] [,2] [,3]  
[1,]  2   3   5  
[2,]  4   1   7
```

данные
количество строк

- `C <- matrix(c(2, 4, 3, 1, 5, 7),
nrow=2,
byrow = TRUE)`

```
  [,1] [,2] [,3]  
[1,]  2   4   3  
[2,]  1   5   7
```

данные
количество строк
по строчкам

Матрицы

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

- $A*10$

```
[,1] [,2] [,3]  
[1,] 20 30 50  
[2,] 40 10 70
```

- транспозиция матрицы
 $t(C)$

```
[,1] [,2]  
[1,] 2 1  
[2,] 4 5  
[3,] 3 7
```

- $A+C$

```
[,1] [,2] [,3]  
[1,] 4 7 8  
[2,] 5 6 14
```

- поэлементное умножение
 $A*C$

```
[,1] [,2] [,3]  
[1,] 4 12 15  
[2,] 4 5 49
```

- умножение матриц
 $A \%*\% t(C)$

```
[,1] [,2]  
[1,] 31 52  
[2,] 33 58
```

- декомпозиция матрицы
 $c(A)$

```
[1] 2 4 3 1 5 7
```

Матрицы: размерности

У любой матрицы есть размерности и их сообщает функция `dim()`:
`dim(A)`

```
[1] 2 3
```

```
dim(t(B))
```

```
[1] 3 2
```

```
m <- c(2, 4, 3, 1, 5, 7)
dim(m) <- c(2, 3)
```

```
# создадим вектор
# изменим его размерности
```

```
      [,1] [,2] [,3]
[1,]    2    3    5
[2,]    4    1    7
```

Многомерные матрицы

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

array в R — это матрица с любым количеством размерностей (нет, это не массив):

```
A <- array(26:3, dim = c(3, 4, 2))
```

```
, , 1
  [,1] [,2] [,3] [,4]
[1,]  26  23  20  17
[2,]  25  22  19  16
[3,]  24  21  18  15
```

```
, , 2
  [,1] [,2] [,3] [,4]
[1,]  14  11   8   5
[2,]  13  10   7   4
[3,]  12   9   6   3
```

```
c(A)
```

```
# декомпозиция
```

```
[1] 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3
```

Списки

Векторы, матрицы и многомерные матрицы позволяют иметь лишь объекты одного класса. Чтобы совмещать единицы разного класса используется функция `list()`.

```
list("abc", TRUE, NA, c(666, 333))
```

```
[[1]]
```

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

```
[[2]]
```

```
[1] TRUE
```

```
[[3]]
```

```
[1] NA
```

```
[[4]]
```

```
[1] 666 333
```

- объектами в `list()` трудно манипулировать
- объекты в `list()` упорядочены
- объекты, содержащие объект `list()`, тоже относятся к типу `list()`

Датафреймы

Датафрейм (иногда называют *таблицей данных*) — одномерный список векторов одинаковой длины.

```
data.frame(num = c(2,3,8,1), tf = c(T,F,T,T), let = c("a "f"k "z"))
```

	num	tf	let
1	2	TRUE	a
2	3	FALSE	f
3	8	TRUE	k
4	1	TRUE	z

Факторы

В R существует отдельный класс векторов, для кодирования единиц из номинальных (род существительного) или порядковых шкал (оценки грамматичности предложения). Иначе векторы с("m", "n", "f") или c(5, 4, 3) воспринимаются R как строки или числа, что может в некоторых случаях вызвать ошибку. Для преобразования вектора в фактор используется следующая функция:

```
factor(c("f", "f", "m", "n", "n", "f", "m", "f"), levels = c("m", "n", "f"))
```

```
[1] f f m n n f m f  
Levels: m n f
```

Пакеты

Большинство дополнительных функции в R содержатся в дополнительных пакетах.

- `install.packages("pkg")` # скачиваем пакет
- `update.packages` # проверяет обновления
- `library("pkg")` # добавляет пакет в сессию
- `detach("package:pkg")` # удаляет пакет из сессии

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

Сравнение и логические операторы

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

$m < n$

меньше

$m > n$

больше

$m \leq n$

меньше или равно

$m \geq n$

больше или равно

$m == n$

равно

$m != n$

не равно

$m \& n$

и

$m | n$

или

$\text{xor}(m, n)$

исключительное или

Таблицы истинности

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

a <- c(TRUE, TRUE, FALSE)

b <- c(TRUE, FALSE, FALSE)

outer(a, b, "&")

и

[,1] [,2] [,3]

[1,] TRUE FALSE FALSE

[2,] TRUE FALSE FALSE

[3,] FALSE FALSE FALSE

outer(a, b, "|")

или

[,1] [,2] [,3]

[1,] TRUE TRUE TRUE

[2,] TRUE TRUE TRUE

[3,] TRUE FALSE FALSE

outer(a, b, "xor")

исключительное или

[,1] [,2] [,3]

[1,] FALSE TRUE TRUE

[2,] FALSE TRUE TRUE

[3,] TRUE FALSE FALSE

Логические операции

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

```
h <- 1:10
```

```
h
```

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

```
h > 4
```

```
[1] FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
h < 8
```

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE
```

```
h > 4 & h < 8 # и
```

```
[1] FALSE FALSE FALSE FALSE TRUE TRUE TRUE FALSE FALSE FALSE
```

```
h > 4 | h < 8 # или
```

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

```
xor(h > 4, h < 8) # исключительное или
```

```
[1] TRUE TRUE TRUE TRUE FALSE FALSE FALSE TRUE TRUE TRUE
```

Циклы

- `if(condition){cmd1} else{cmd2}`
`if(i %% 2 == 0){`
 `print("even")`
 `} else {print("odd")}`
- `ifelse(condition, cmd1, cmd2)`
`ifelse(i %% 2 == 0, "even", "odd")`
- `for(variable in x){cmd}`
`for(i in 10:20){`
 `print(exp(i))`
}
- `while(condition){cmd}`
`while(i < 7){`
 `print(exp(i))`
 `i <- i+1`
}

R считается медленным языком, так что осторожнее с циклами. В книге **H. Wickham Advanced R** говорится, почему так, и как бороться.

презентация доступна: <http://idrv.ms/1TDPR6m>

Пустые переменные

При работе с циклами часто бывает полезно заранее создать пустую переменную, в которую будут писаться значения.

- векторы

- `numeric(p)` # создает вектор типа `numeric` длины `p`
- `integer(p)` # создает вектор типа `integer` длины `p`
- `complex(p)` # создает вектор типа `complex` длины `p`
- `character(p)` # создает вектор типа `character` длины `p`
- `logical(p)` # создает вектор типа `logical` длины `p`
- `rep(NA_real_, p)` # создает вектор типа `integer` длины `p`
- `rep(NA_integer_, p)` # создает вектор типа `integer` длины `p`
- `rep(NA_complex_, p)` # создает вектор типа `complex` длины `p`
- `rep(NA_character_, p)` # создает вектор типа `character` длины `p`

- `vector(mode = "list", length = 10)` # 10-элементный список

- `array(NA, dim = c(p, q, ...))` # матрицы с измерениями `p, q, ...`

- датафреймы

Чтение

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

- команды `read.table` и `read.csv` почти идентичны

```
X <- read.table("/agricolamz/work/R/ALex.txt",          # путь
               sep = "\t",                             # тип разделителя
               header = TRUE)                          # первая строка — имена столбцов?
```

```
X <- read.csv("http://goo.gl/GkfYYt",                 # url
              sep = "\t",                             # тип разделителя
              header = TRUE)                          # первая строка — имена столбцов?
```

- можно при помощи функции `setwd()` установить рабочую директорию и обращаться к файлам по имени

```
setwd("/agricolamz/work/R")                          # путь
X <- read.table("ALex.txt",                            # имя файла
               sep = "\t",                             # тип разделителя
               header = TRUE)                          # первая строка — имена столбцов?
```

- для текстов есть команда `readLines()`

```
text <- readLines("test-2.txt",                       # имя файла
                  n = 10)                             # количество абзацев
```


Разведка

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

- ? ставится перед функцией
- apropos("x") ищет, все, содержащее "x"
- summary(x) для данных
- str(x) и для данных, и для функций
- head(x) первые 6 значений объекта
- tail(x) последние 6 значений объекта
- typeof(x) возвращает тип данных
- is.что-то(x) возвращает вектор логических значений
- table(x) возвращает таблицу сопряженности
- unique(x) возвращает данные без дублей
- max(x), min(x) возвращает максим./миним. значение
- length(x), nrow(x), ncol(x) количество элементов
- ls() возвращает все переменные данной сессии
- getAnywhere(x) возвращает код функции (не для всех)

Indexing: векторы

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

```
x <- c(43:21)
```

- `x[3]`

```
[1] 41
```

в квадратных скобках

Любители Python! Индексация начинается с 1!

- `c(43:21)[3]`

```
[1] 41
```

аналогично можно с любым вектором

- `x[x%%7 == 0]`

```
[1] 42 35 28 21
```

логическое выражение

- `x[x%%7 == 0 & x%%13 == 0]`

```
[1] 42 28
```

логическое выражение

- `x[-c(3:19)]`

```
[1] 43 42 24 23 22 21
```

все кроме элементов 3:19

```
x <- c(Anna = 23, Wanda = 24, Agnieszka = 28)
```

вектор с именами

- `x["Anna"]`

```
Anna
```

обращение по имени

```
23
```

презентация доступна: <http://1drv.ms/1TDPR6m>

Indexing: матрицы

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

```
x <- matrix(9:60, nrow = 4)
```

○ `x[3]` # третий элемент
[1] 11

○ `x[3,4]` # третий столбец, четвертая строчка
[1] 23

○ `x[3,]` # элементы третьей строчки
[1] 11 15 19 23 27 31 35 39 43 47 51 55 59

○ `x[, 3]` # элементы третьего столбца
[1] 17 18 19 20

○ `x[-c(1, 4),]` # все, кроме первой и четвертой строчек
[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
[1,] 10 14 18 22 26 30 34 38 42 46 50 54 58
[2,] 11 15 19 23 27 31 35 39 43 47 51 55 59

Indexing: списки и датафреймы

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

Объект `list()` содержит несколько векторов. Поэтому чтобы обратиться к какому-то из векторов объекта `list()` рекомендуется использовать двойные квадратные скобки:

```
y <- list(tf = c(T, F), num = c(82, 99, 21), com = 2 + 9i)
y[[2]]
```

второй вектор списка

```
[1] 82 99 21
```

Чтобы обратиться к элементу вектора следует использовать **только двойные скобки**, для обращения к векторы, и одинарные для обращения к элементу:

```
y[[2]][3]
```

третий элемент второго вектора

```
[1] 21
```

Все это работает рекурсивно, так как объекты `list()` могут быть вложены друг в друга. Это распространяется и на обращения по имени.

презентация доступна: <http://1drv.ms/1TDPR6m>

Indexing: датафреймы

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

Обращение к элементам датафрейма устроены так же, как и в `list()` и `matrix()`:

```
z <- data.frame(tf = c(T, F), num = c(82, 99), com = c(2 + 9i, 8 + 3i))
```

```
z[[2]]
```

```
[1] 82 99
```

```
z[1, 2]
```

```
[1] 82
```

К векторам можно обращаться по имени:

```
z$com
```

```
[1] 2+9i 8+3i
```

```
z$com[2]
```

```
[1] 8+3i
```

Кроме того, важно отметить, что к элементам можно обращаться, указывая лишь часть имени, например, первую букву:

```
z$c[2]
```

```
[1] 8+3i
```

Соединение

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафреймы

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

- векторы: `c(a, b)`
- матрицы: `rbind(a, b)`, `cbind(a, b)`, многомерные — `abind()`
- списки:

```
a <- list(2, 3)
b <- list(5, 6)
c(a, b)
```

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

```
a <- list(2, 3)
b <- list(5, 6)
c(list(a), list(b))
```

```
[[1]]
 [[1]][[1]]
  [1] 2
 [[1]][[2]]
  [1] 3
[[2]]
 [[2]][[1]]
  [1] 5
 [[2]][[2]]
  [1] 6
```

```
a <- list(2, 3)
b <- list(5, 6)
mapply(c, a, b,
        SIMPLIFY=F)
```

```
[[1]]
 [1] 2 5
[[2]]
 [1] 3 6
```

- датафреймы: `rbind.data.frame(a, b)`, `cbind.data.frame(a, b)`

Сортировки: вектора

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

```
a <- c(1, 3, 1, 2, 2, 7, 4, 4, 2, 2, 2, 3, 9, 10, 2, 3, 4, 6, 5)
rev(a)
```

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

```
sort(a)
```

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

```
sort(a, decreasing = T)
```

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

```
sort(a, index.return = T)
```

```
$x
```

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

```
$ix
```

```
[1] 1 3 4 5 9 10 11 15 2 12 16 7 8 17 19 18 6 13 14
```

Сортировки: матрицы

Так как матрицы — всего лишь вектора с измерениями, то все команды сортировки затрагивают их целиком и "убивают" измерения.

```
m <- matrix(c(4, 2, 3, 5, 1, 2, 6, 2, 6, 3, 1, 9), nrow = 4)
sort(m)
```

```
[1] 1 1 2 2 2 3 3 4 5 6 6 9
```

В связи с этим, если мы хотим сохранить структуру, измерения получившегося вектор следует изменить под старые:

```
p <- sort(m); dim(p) <- dim(m)
p
```

	[,1]	[,2]	[,3]
[1,]	1	2	5
[2,]	1	3	6
[3,]	2	3	6
[4,]	2	4	9

Сортировки: матрицы

В пакет BioPhysConnectoR реализована функция `mat.sort` позволяющая сортировать матрицы по выбранной колонке:
`library("BioPhysConnectoR")`

b	mat.sort(b)				mat.sort(b, 3)						
	[,1]	[,2]	[,3]		[,1]	[,2]	[,3]		[,1]	[,2]	[,3]
[1,]	4	1	6	[1,]	2	2	3	[1,]	3	6	1
[2,]	2	2	3	[2,]	3	6	1	[2,]	2	2	3
[3,]	3	6	1	[3,]	4	1	6	[3,]	4	1	6
[4,]	5	2	9	[4,]	5	2	9	[4,]	5	2	9

Сортировки: датафрейма

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

Для сортировки матрицы используется команда `order()`

```
df <- data.frame(
  name = c("Agnieszka", "Marek", "Marta", "Magda", "Jan"),      # имя
  age = c(24, 23, 19, 25, 21),                                   # возраст
  sex = c("f", "m", "f", "f", "m"),                             # пол
  lang = c(2, 2, 1, 4, 2))                                       # владение языками
```

```
df[order(df[4]),]
df[order(df$lang),]
```

```
df[order(-df[4], df[2]),]
df[order(-df$lang, df$age),]
```

	name	age	sex	lang
3	Marta	19	f	1
1	Agnieszka	24	f	2
2	Marek	23	m	2
5	Jan	21	m	2
4	Magda	25	f	4

	name	age	sex	lang
4	Magda	25	f	4
5	Jan	21	m	2
2	Marek	23	m	2
1	Agnieszka	24	f	2
3	Marta	19	f	1

Преобразования

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

```
df <- data.frame(
  name = c("Agnieszka", "Marek", "Marta", "Magda", "Jan"),      # имя
  age = c(24, 23, 19, 25, 21),                                     # возраст
  sex = c("f", "m", "f", "f", "m"),                               # пол
  lang = c(2, 2, 1, 4, 2))                                         # владение языками
```

```
table(df$sex)      table(df$sex, df$lang)      prop.table(table(df$sex, df$lang))
```

		4	1	2			
f	m	f	1	1	1	f	m
3	2	m	0	0	2	0.6	0.4

```
as.data.frame(table(df$sex, df$lang))
```

	Var1	Var2	Freq
1	f	1	1
2	m	1	0
3	f	2	1
4	m	2	2
5	f	4	1
6	m	4	0

Удаление NA

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

- вектор:

```
a <- c(1, 3, 5, NA, 1, 3, NA)
```

```
a[!is.na(a)] или a[complete.cases(a)]
```

```
[1] 1 3 5 1 3
```

- датафрейм:

```
z <- data.frame(tf = c(T, F, NA, T),
```

```
num = c(82, NA, 99, 56),
```

```
com = c(2 + 9i, 8 + 3i, 5 + 2i, 6 + 1i))
```

```
z[complete.cases(z),]
```

```
      tf num  com  
1 TRUE  82 2+9i  
4 TRUE  56 6+1i
```

Запись файлов

- `write.csv(df, file = "/agricolamz/work/R/calculi.csv")` # путь

Можно при помощи функции `setwd()` установить рабочую директорию и не прописывать весь путь:

- `write.table(df, file = "calculi.csv", sep = ";")` # разделитель

- для текстов есть команда `writeLines`

Расход времени и памяти

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

- `Sys.time()`
- `system.time()`

`library("pryr")`

- `object_size()`
- `mem_used()`
- `mem_change()`

Функции

Кроме встроенных функций пользователю также предоставлена возможность делать функции самостоятельно:

`function(аргументы){действия}`

`t <- function(x){factorial(x)}` # функция с одной переменной

`u <- function(x, y){x + y}` # функция с двумя переменными

`v <- function(x = 0){exp(x)}` # функция со значением по умолчанию

`v()`

`[1] 2.718282`

`v(4)`

`[1] 54.59815`

умолчание одной переменной может зависеть от другой

`w <- function(x=1, y = x*10){sum(x:y)}`

`w()`

`[1] 55`

`w(4)`

`[1] 814`

`w(4, 2)`

`[1] 9`

`(function(x){3*x})(10)`

анонимная функция

Функции

To understand computations in R, two slogans are helpful:

- Everything that exists is an object.
- Everything that happens is a function call.

— John Chambers

Косые кавычки ``...`` позволяют обращаться к именам всех функций, даже с нестандартным синтаксисом.

```
x <- 10; y <- 5; x + y  
`(x, y)
```

сложим два вектора
тот же результат

```
for (i in 9:6) print(i)  
`for`(i, 1:2, print(i))
```

цикл
тот же результат

```
c(5:8)[3]  
`[(c(5:8), 3)
```

третий элемент вектора
тот же результат

об R

калькулятор

объекты

векторы

матрицы

многомерные матрицы

списки

датафрейм

факторы

пакеты

if, else, for, while

манипуляции

чтение

разведка

indexing

соединение

сортировки

reshaping

удаление NA

запись

память и время

функции

Спасибо за внимание

Пишите письма

agricolamz@gmail.com