

Анализ данных

Лекции 2–3

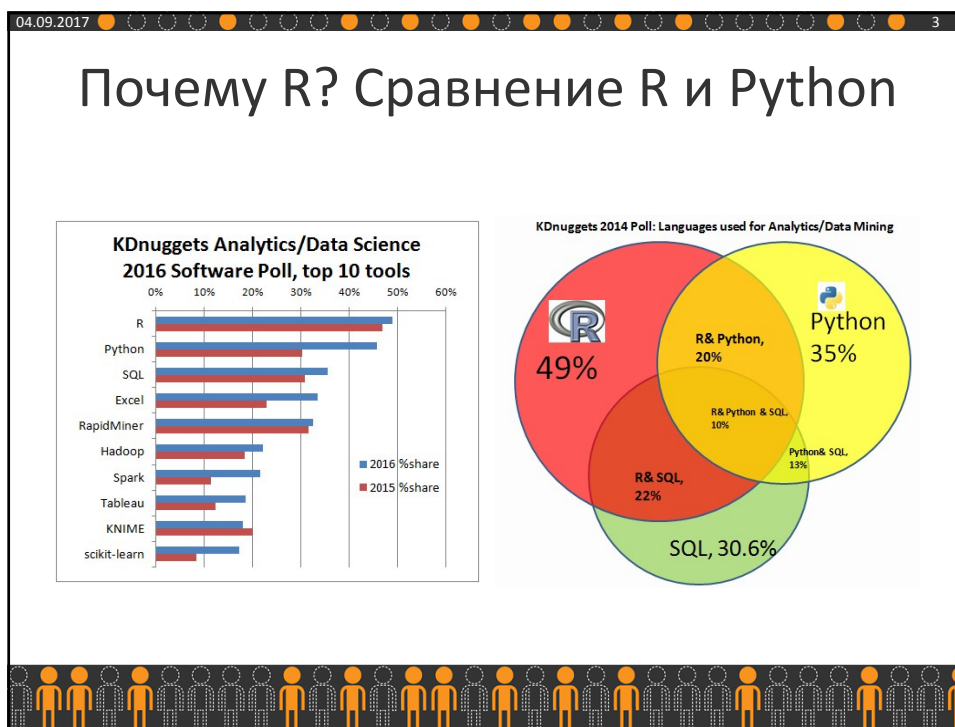
Программная среда анализа данных и язык программирования R

Гедранович Ольга Брониславовна,
старший преподаватель кафедры ИТ, МИУ
volha.b.k@gmail.com

04.09.2017 2

Вопросы лекции

- Почему R? Сравнение R и Python
- Обзор и история R
- Консоль R
- Интегрированная среда разработки RStudio
- Объекты и атрибуты. Пакеты и библиотеки
- Типы данных: векторы, матрицы, факторы, списки, блоки данных (data frames), tibble
- magrittr: Simplifying R code with pipes
- Чтение и запись данных. Форматы данных
- Представление даты и времени. Временные ряды
- Организация вычислений: функции, ветвления, циклы
- Векторизованные вычисления в R



04.09.2017 4

Почему R? Сравнение R и Python

Data Science Wars: R vs Python

<https://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis#gs.bsjqDmY>

[Zoom](#)

04.09.2017 5

Обзор и история R

Why Learn R? edureka!

Open Source Language

Cross-Platform Compatible

Advanced Statistical Language

Outstanding Graphs

Relates to other Languages

Vast Community

Supports Extensions

Extremely Comprehensive

Flexible 'n' Fun

<https://www.quora.com/What-is-it-that-Python-has-and-R-doesnt-when-using-them-for-Data-Science>

04.09.2017 6

Обзор и история R

- 1976 — **язык S** разработан в компании Bell Labs и был назван «по мотивам» языка C. Первая реализация S была написана на FORTRAN и работала под управлением операционной системы GCOS.
- 1980 — реализация была переписана под UNIX, и с этого момента S стал распространяться в основном в научной среде.
- 1988 — коммерческая реализация S — **S-Plus**. Распространялась компанией Insightful, а сейчас — компанией TIBCO Software. Версии S-Plus доступны под Windows и различные версии UNIX за плату (версия для UNIX стоит порядка \$6500).
- Августе 1993 — двое молодых новозеландских ученых Robert Gentleman и Ross Ihaka анонсировали свою новую разработку, которую они назвали **R**.

04.09.2017 7

Обзор и история R



Robert Gentleman Ross Ihaka

Decorative border with orange and grey human icons.

04.09.2017 8

Обзор и история R

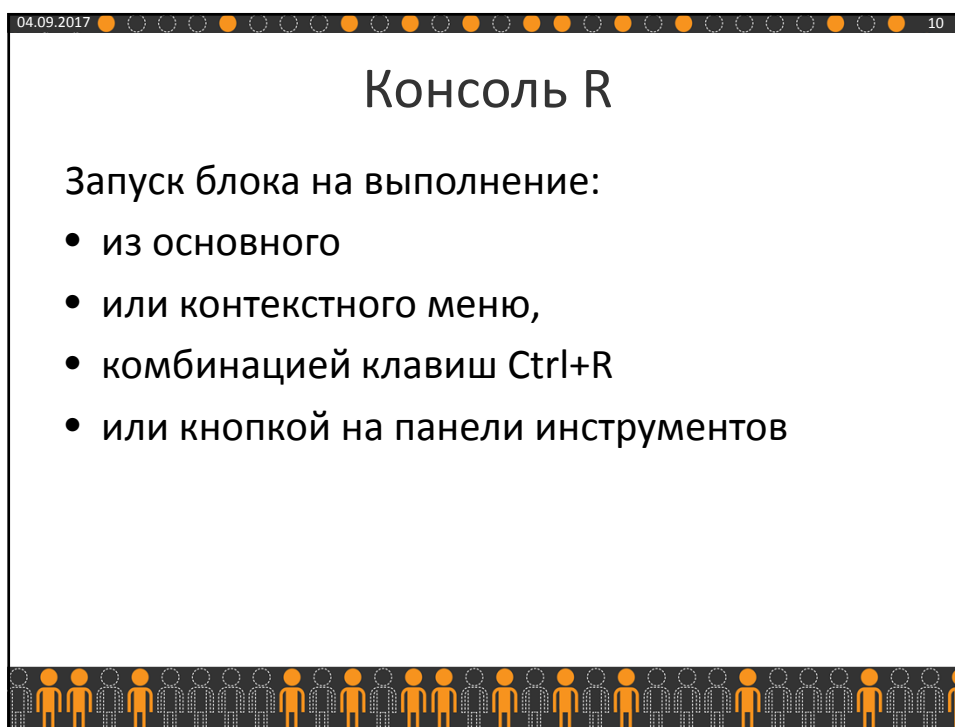
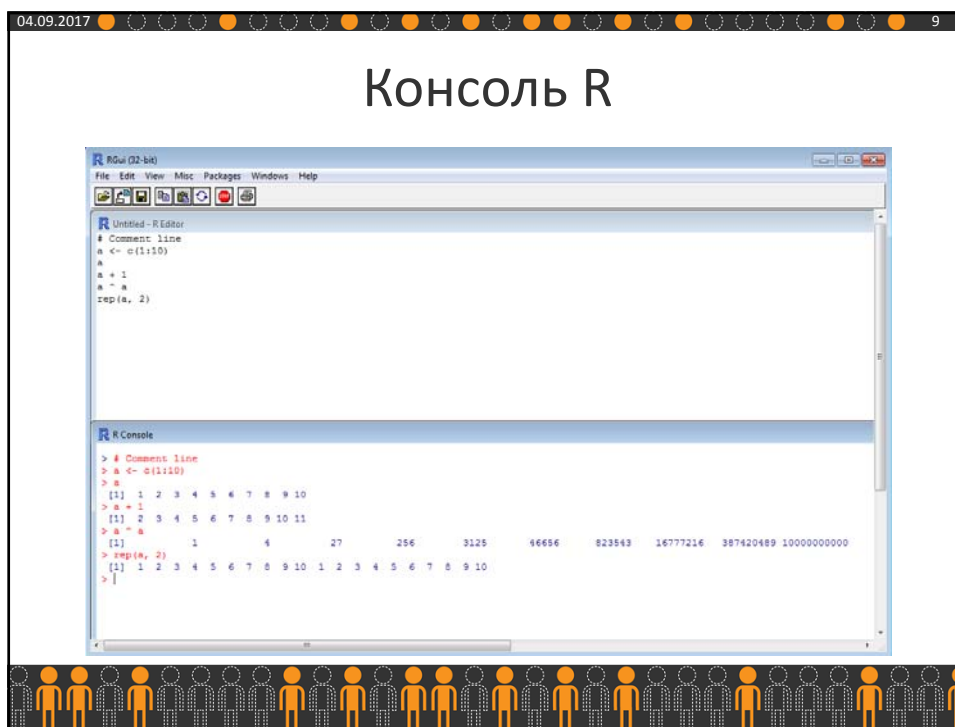
Главные преимущества:

- гибкость
- свободный код

Недостатки:

- трудность обучения языку
- относительная медлительность

Decorative border with orange and grey human icons.



04.09.2017 11

Консоль R: help.start()

Statistical Data Analysis

Manuals

- [An Introduction to R](#)
- [The R Language Definition](#)
- [Writing R Extensions](#)
- [R Installation and Administration](#)
- [R Data Import/Export](#)
- [R Internals](#)

Reference

- [Packages](#)
- [Search Engine & Keywords](#)

Miscellaneous Material

- [About R](#)
- [Authors](#)
- [Resources](#)
- [License](#)
- [Frequently Asked Questions](#)
- [Thanks](#)
- [NEWS](#)
- [User Manuals](#)
- [Technical papers](#)

Material specific to the Windows port

- [CHANGES up to R 2.15.0](#)
- [Windows FAQ](#)

04.09.2017 12

Консоль R

Справка по функциям:

- `help("foo")` или `?foo` – справка по функции foo (кавычки необязательны);
- `help.search("foo")` или `??foo` – поиск всех справочных файлов, содержащих foo;
- `example("foo")` – примеры использования функции foo;
- `RSiteSearch("foo")` – поиск ссылок в онлайн-руководствах и архивах рассылок;
- `apropos("foo", mode = "function")` – список всех функций с комбинацией foo;
- `vignette("foo")` – список руководств по теме foo.

04.09.2017 13

Интегрированная среда разработки RStudio

An integrated development environment (IDE)
RStudio позволяет дополнительно выполнять:

- подсветку синтаксиса кода,
- его автоматическое завершение,
- "упаковку" последовательности команды в функции для их последующего использования,
- работу с документами Sweave или TeX
- и другие операции.

04.09.2017 13

04.09.2017 14

RStudio

RStudio IDE features:

- runs on most desktops or on a server and accessed over the web;
- integrates the tools you use with R into a single environment;
- includes powerful coding tools designed to enhance your productivity;
- enables rapid navigation to files and functions;
- makes it easy to start new or find existing projects;
- has integrated support for Git and Subversion;
- supports authoring HTML, PDF, Word Documents, and slide shows;
- supports interactive graphics with Shiny and ggvis.

<https://www.rstudio.com/products/rstudio/>


04.09.2017 14

04.09.2017 17

Объекты и атрибуты. Пакеты и библиотеки

Выделяют два основных типа **объектов**:

1. Объекты, предназначенные для хранения данных ("data objects") – отдельные переменные, векторы, матрицы и массивы, списки, факторы, таблицы данных.
2. Функции ("function objects") – поименованные программы, предназначенные для создания новых объектов или выполнения определенных действий над ними.




04.09.2017 18

Объекты и атрибуты. Пакеты и библиотеки

Любой объект языка R имеет набор **атрибутов (attributes)**. Этот набор может быть разным для объектов разного вида, но каждый объект обязательно имеет пять встроенных атрибутов:

- длина (length);
- вид (mode);
- класс (class);
- тип (typeof);
- структура (str).




04.09.2017 19

Объекты и атрибуты. Пакеты и библиотеки

Можно написать функцию, которая будет возвращать значения атрибутов какого-либо объекта:

```
inspect <- function(a) {  
  list(.length = length(a), .mode = mode(a),  
       .type = typeof(a), .class = class(a),  
       .str = str(a))  
}
```




04.09.2017 20

Объекты и атрибуты. Пакеты и библиотеки

- **library (библиотека)** определяет директорию, которая может содержать один или несколько пакетов
- **package (пакет)** обозначает совокупность функций, HTML-страниц руководств и примеров объектов данных, предназначенных для тестирования или обучения

```
library(help=<имя_пакета>)
```




04.09.2017 21

Объекты и атрибуты. Пакеты и библиотеки

```
# Получение полного списка пакетов
packlist <- rownames(installed.packages())

# Вывод информации в буфер обмена в формате
для Excel
write.table(packlist, "clipboard", sep="\t",
col.names=NA)

# Установка пакетов
install.packages(c("vegan", "xlsReadWrite",
"car"))
```



04.09.2017 22


Объекты и атрибуты. Пакеты и библиотеки

При запуске консоли RGui загружаются только некоторые базовые пакеты. Для инициализации любого другого пакета перед непосредственным использованием его функций нужно ввести команду

```
library(<имя_пакета>)
```

Установить, какие пакеты загружены в каждый момент проводимой сессии:

```
sessionInfo()
```



04.09.2017 23


Объекты и атрибуты. Пакеты и библиотеки

Получить список аргументов входящих параметров любой функции:

```
> args(lm)
function (formula, data, subset, weights, na.action,
method = "qr", model = TRUE, x = FALSE, y = FALSE,
qr = TRUE, singular.ok = TRUE, contrasts = NULL,
offset,...)
```

Если ввести команду, состоящую только из аббревиатуры функции (например, IQR), можно получить исходный текст функции:

```
> IQR
function (x, na.rm = FALSE)
diff(quantile(as.numeric(x), c(0.25, 0.75), na.rm
= na.rm, names = FALSE))
```




04.09.2017 24

Типы данных: векторы, матрицы, факторы, списки, блоки данных (data frames)

Все объекты данных (а, следовательно, и переменные) в R можно разделить на следующие классы (типы объектов):


- **numeric** – объекты, к которым относятся целочисленные (**integer**) и действительные числа (**double**);
- **complex** – содержат комплексные числа, мнимая часть комплексного числа записывается с символом *i* на конце, например, `c(5.0i, -1.3+8.73i, 2.0)`;
- **logical** – логические объекты, которые принимают только два значения: `FALSE` и `TRUE`;
- **character** – символьные объекты (значения переменных задаются в двойных, либо одинарных кавычках).



04.09.2017 25

Типы данных: векторы, матрицы, факторы,
списки, блоки данных (data frames)

```
exists("<имя>")  
is.numeric(<имя>)  
is.integer(<имя>)  
is.logical(<имя>)  
is.complex(<имя>)  
is.character(<имя>)  
as.numeric(<имя>)  
as.integer(<имя>)
```




04.09.2017 26

Типы данных: векторы, матрицы, факторы,
списки, блоки данных (data frames)

- **Inf** – положительная или отрицательная бесконечность (обычно результат деления вещественного числа на 0)
- **NA** – "отсутствующее значение" (Not Available)
- **NaN** – "не число" (Not a Number)

```
is.finite(<имя>), is.infinite(<имя>)  
is.na(<имя>)  
is.nan(<имя>)
```



04.09.2017 27

Типы данных: векторы, матрицы, факторы,
списки, блоки данных (data frames)


Оператор присваивания:

=

<- (присваивание значения объекту слева)

-> (присваивание значения объекту справа).

Хорошим стилем программирования
считается использование <-.



04.09.2017 28

Типы данных: векторы, матрицы, факторы,
списки, блоки данных (data frames)

Арифметические операции:

+ (сложение)

– (вычитание)


* (умножение)

/ (деление)

^ (возведение в степень)

%% (целочисленное деление)

%% (остаток от деления)




04.09.2017 29

Типы данных: векторы, матрицы, факторы, списки, блоки данных (data frames)

Логические операторы:

- "Равно" ==
- "Не равно" !=
- "Меньше" <
- "Больше" >
- "Меньше либо равно" <=
- "Больше либо равно" >=
- "Логическое И" & (&& – сравнивает только первые элементы)
- "Логическое ИЛИ" |
- "Логическое НЕ" !



04.09.2017 30


Типы данных: векторы, матрицы, факторы, списки, блоки данных (data frames)

```
> fr <- c(TRUE, TRUE, FALSE)
> sc <- c(TRUE, TRUE, TRUE)
> th <- c(FALSE, TRUE, TRUE)

> fr & sc == TRUE
[1] TRUE TRUE FALSE

> fr && sc == TRUE
[1] TRUE

> fr && th == TRUE
[1] FALSE
```




04.09.2017 31

Векторы

Вектор представляет собой поименованный одномерный объект, содержащий набор однотипных элементов (сочетания не допускаются).

- Для создания векторов небольшой длины в R используется функция конкатенации **c()** (от "concatenate" – объединять, связывать).

```
> my.vector <- c(1, 2, 3, 4, 5)
> my.vector
[1] 1 2 3 4 5
```



04.09.2017 32


Векторы

```
scan()
fix()
seq()
rep()

V <- c(v1, v2)

v[index of element]

sort(v, decreasing = TRUE)
```



04.09.2017 33

Матрицы

Матрица представляет собой двумерный вектор.

- В R для создания матриц служит одноименная функция:

```
> my.mat <- matrix(seq(1, 16), nrow = 4, ncol = 4)
```

```
> my.mat
```

	[,1]	[,2]	[,3]	[,4]
[1,]	1	5	9	13
[2,]	2	6	10	14
[3,]	3	7	11	15
[4,]	4	8	12	16

04.09.2017 34

Матрицы

```
rownames(my.mat) <- c("A", "B", "C", "D")
```

```
dim()
```

```
cbind()
```

```
rbind()
```

```
t()
```


```
my.mat[row index, col index]
```

04.09.2017 35

Факторы

Факторы (factors) – специальный класс векторов, которые являются способом представления номинальных (категориальных) и шкальных типов данных в R.

```
> treatment <- c(1, 1, 1, 1, 1, 1, 0, 0, 0, 0)
> treatment <- factor(treatment, levels = c(0, 1))
> treatment
[1] 1 1 1 1 1 1 0 0 0 0
Levels: 0 1
```




04.09.2017 36

Факторы

`gl(n, k, length = n*k, labels = 1:n)`, где

- `n` – количество уровней фактора;
- `k` – число повторов для каждого уровня;
- `length` – размер итогового объекта;
- `labels` – необязательный аргумент, который можно использовать для указания названий каждого уровня фактора.




04.09.2017 37

Списки

В отличие от вектора или матрицы, которые могут содержать данные только одного типа, в **список (list)** можно включать сочетания любых типов данных.

```
vector1 <- c("A", "B", "C")
vector2 <- seq(1, 3, 0.5)
vector3 <- c(FALSE, TRUE)
my.list <- list(Text=vector1, Number=vector2,
Logic=vector3)
my.list
$Text
[1] "A" "B" "C"
$Number
[1] 1.0 1.5 2.0 2.5 3.0
$Logic
[1] FALSE TRUE
```



04.09.2017 38

Списки


```
my.list$Text

my.list$Number[3:5]

my.list[[1]]

my.list[[3]][1]


str(my.list)
```



04.09.2017 39

Таблица данных (data frame)

Каждый столбец **таблицы данных** является вектором, содержащим данные определенного типа. При этом все столбцы должны иметь одинаковую длину (собственно, с "точки зрения" R таблица данных является частным случаем списка, в котором все компоненты-векторы имеют одинаковый размер).




04.09.2017 40

Таблица данных (data frame)

```
> city <- c("City1", "City1", "City2", "City2", "City3",
"City3")
> sex <- c("Male", "Female", "Male", "Female", "Male",
"Female")
> number <- c(12450, 10345, 5670, 5800, 25129, 26000)

> CITY <- data.frame(City = city, Sex = sex,
Number = number)
> CITY
```

	City	Sex	Number
1	City1	Male	12450
2	City1	Female	10345
3	City2	Male	5670
4	City2	Female	5800
5	City3	Male	25129
6	City3	Female	26000



04.09.2017 41

Таблица данных (data frame)

```
> CITY$Sex
[1] Male Female Male Female Male Female
Levels: Female Male
```

Идентичные результаты можно получить при помощи команд:


```
> CITY[,2]
> CITY[[2]]
```

Тогда как предыдущие команды возвращали список, команда

```
> CITY["Sex"]
```

вернет таблицу данных:


	Sex
1	Male
2	Female
3	Male
4	Female
5	Male
6	Female



04.09.2017 42

Таблица данных (data frame)

```
head()
tail()
fix()
DF[is.na(DF)] <- 0
order()
cbind()
rbind()
merge()
```




04.09.2017 43

Tibble

Tibbles are a modern reimagining of the `data.frame`, keeping what time has proven to be effective, and throwing out what is not. The name comes from `dplyr`: originally you created these objects with `tbl_df()`, which was most easily pronounced as “**tibble diff**”.


```
install.packages("tibble")
data_frame()
View
```



04.09.2017 44

Tibble

```
library(nycflights13)
flights
#> Source: local data frame [336,776 x 16]
#>
#>   year month   day dep_time dep_delay arr_time arr_delay carrier tailnum
#>   (int) (int) (int)   (int)      (dbl)      (int)      (dbl)   (chr)   (chr)
#> 1  2013     1     1     517         2      830         11     UA   N14228
#> 2  2013     1     1     533         4      850         20     UA   N24211
#> 3  2013     1     1     542         2      923         33     AA   N619AA
#> 4  2013     1     1     544        -1     1004        -18     B6   N804JB
#> 5  2013     1     1     554        -6      812        -25     DL   N668DN
#> 6  2013     1     1     554        -4      740         12     UA   N39463
#> 7  2013     1     1     555        -5      913         19     B6   N516JB
#> 8  2013     1     1     557        -3      709        -14     EV   N829AS
#> 9  2013     1     1     557        -3      838         -8     B6   N593JB
#> 10 2013     1     1     558        -2      753          8     AA   N3ALAA
#> .. ... ..
#> Variables not shown: flight (int), origin (chr), dest (chr), air_time
#>   (dbl), distance (dbl), hour (dbl), minute (dbl).
```




04.09.2017 45

Tibble

It's possible to change the default printing appearance as follow:

- Change the maximum and the minimum rows to print: `options(tibble.print_max = 20, tibble.print_min = 6)`
- Always show all rows: `options(tibble.print_max = Inf)`
- Always show all columns: `options(tibble.width = Inf)`




04.09.2017 46

magrittr: Simplifying R code with pipes

R is a functional language, which means that your code often contains a lot of (parentheses). And complex code often means nesting those parentheses together, which make code hard to read and understand.

But there's a very handy R package — **magrittr**, by *Stefan Milton Bache* — which lets you transform nested function calls into a simple **pipeline** of operations that's easier to write and understand.




04.09.2017 47

magrittr: Simplifying R code with pipes

```
hourly_delay <- filter(
  summarise(
    group_by(
      filter(
        flights,
        !is.na(dep_delay)
      ),
      date, hour
    ),
    delay = mean(dep_delay),
    n = n()
  ),
  n > 10
)
```

```
hourly_delay <- flights %>%
  filter(!is.na(dep_delay)) %>%
  group_by(date, hour) %>%
  summarise(
    delay = mean(dep_delay),
    n = n() ) %>%
  filter(n > 10)
```

%>% (THEN)




04.09.2017 48

Чтение и запись данных. Форматы данных

```
edit()
options(editor="c:\\Program
Files\\CrazyPad\\crazypad.exe")

getwd()
setwd("e:/wrk/temp")
dir("data")
```



04.09.2017 49

Чтение и запись данных. Форматы данных

```
> read.table("data/mydata.txt",
sep=";", head=TRUE)
```

	a	b	c
1	1	2	3
2	4	5	6
3	7	8	9

```
file.show("data/mydata.txt")
```

04.09.2017 50

Чтение и запись данных. Форматы данных

(подробнее см. файл помощи, доступный по команде `?read.table`).

Аргумент	Назначение
<code>file</code>	Служит для указания пути к импортируемому файлу. Путь приводит либо в абсолютном виде (например, <code>file = "C:/Temp/MyData.dat"</code>), либо указывают только имя импортируемого файла (например, <code>file = "MyData.txt"</code>), но при условии, что последний хранится в рабочей папке программы (см. выше). В качестве имени можно также указывать полную URL-ссылку на файл, который предполагается загрузить из Сети (например: <code>file = "http://somesite.net/YourData.csv"</code>). Начиная с версии R 2.10, появилась возможность импортировать архивированные файлы в zip-формате.
<code>header</code>	Служит для сообщения программе о наличии в загружаемом файле строки с заголовками столбцов. По умолчанию принимает значение <code>FALSE</code> . Если строка с заголовками столбцов имеется, этому аргументу следует присвоить значение <code>TRUE</code> .
<code>row.names</code>	Служит для указания номера столбца, в котором содержится имена строк (например, в рассмотренном выше примере это был первый столбец, поэтому <code>row.names = 1</code>). Важно помнить, что все имена строк должны быть уникальными, т.е. одинаковые имена для двух или более строк не допускаются.
<code>sep</code>	Служит для указания разделителя значений переменных, используемого в файле (двоеточие – разделитель). По умолчанию предполагается, что значения переменных разделены "пустым пространством", например, в виде пробела или знака табуляции (<code>sep = ""</code>). В файлах формата <code>csv</code> значения переменных разделены запятыми, и поэтому для них <code>sep = ","</code> .
<code>dec</code>	Служит для указания знака, используемого в файле для отделения целой части числа от дроби. По умолчанию <code>dec = "."</code> . Однако во многих странах в качестве десятичного знака применяют запятую, о чем важно вспомнить перед загрузкой файла и, при необходимости, использовать <code>dec = ","</code> . Сделайте, чтобы <code>dec</code> и <code>sep</code> не были бы одинаковыми.
<code>nrows</code>	Выражается целым числом, указывающим количество строк, которое должно быть считано из загружаемой таблицы. Остаточные и иные значения игнорируются. Пример: <code>nrows = 100</code> .
<code>skip</code>	Выражается целым числом, указывающим количество строк в файле, которое должно быть пропущено перед началом импортирования. Пример: <code>skip = 5</code> .


04.09.2017 51

Чтение и запись данных. Форматы данных

Основные форматы данных:

- текстовые
- бинарные

```
пакет foreign  
read.table("clipboard")  
read.csv()  
ReadRDS()
```




04.09.2017 52

Представление даты и времени

Анализ данных, содержащих даты и время, может быть достаточно трудоемким. Причин этому несколько:

- разные годы начинаются в разные дни недели;
- високосные годы имеют дополнительный день в феврале;
- американцы и европейцы по-разному представляют даты (например, 8/9/2011 будет 9-м августа 2011 г. для первых и 8-м сентября 2011 г. для вторых);
- в некоторые годы добавляется так называемая "секунда координации";
- страны различаются по временным поясам и в ряде случаев применяют переход на "зимнее" и "летнее" время.



04.09.2017 53


Представление даты и времени

```
> Sys.time()
[1] "2017-01-31 13:47:23 MSK"

> substr(as.character(Sys.time()), 1, 10)
[1] "2017-01-31"

> substr(as.character(Sys.time()), 12, 19)
[1] "13:52:41"

> date()
[1] "Tue Jan 31 12:53:35 2017"
```



04.09.2017 54

Представление даты и времени


```
> unclass(Sys.time())
[1] 1485860107
(формат POSIXct – в секундах, прошедших с 1 января 1970 г. )

> (date <- as.POSIXlt(Sys.time()))
[1] "2017-02-01 14:37:24 MSK"

> unlist(unclass(date))
      sec      min      hour      mday
"18.8165259361267"  "38"    "14"    "1"
      mon      year      wday      yday
      "1"      "117"    "3"      "31"
      isdst      zone      gmtoff
      "0"      "MSK"    "10800"
```

```
> date$yday
[1] 3

> date$yday
[1] 31
```




04.09.2017 55

Представление даты и времени

В R можно выполнять следующие типы вычислительных операций с датами и временем:

- число + время;
- время – число;
- время1 – время2;
- время1 "логический оператор" время2
(в качестве логического оператора могут использоваться ==, !=, <=, <, > или >=).

В R нельзя складывать две даты.



04.09.2017 56


Представление даты и времени

```
> t3<-as.POSIXlt("2010-09-22 08:30:30")
> t4<-as.POSIXlt("2010-09-22 22:25:30")
> t4-t3
Time difference of 13.91667 hours

> difftime("2011-09-22", "2010-06-22")
Time difference of 457 days

> as.numeric(difftime("2011-09-22", "2010-06-22"))
[1] 457

proc.time()
```



04.09.2017 57

Представление даты и времени

`strptime()`

- `%a` – сокращенное название для недели (англ. яз.)
- `%A` – полное название для недели (англ. яз.)
- `%b` – сокращенное название месяца (англ. яз.)
- `%B` – полное название месяца (англ. яз.)
- `%d` – день месяца (01–31)
- `%H` – часы от 00 до 23
- `%I` – часы от 01 до 12
- `%j` – порядковый номер дня года (001–366)
- `%m` – порядковый номер месяца (01–12)
- `%M` – минуты (00–59)
- `%S` – секунды (00–61, с возможностью добавить "високосную секунду")
- `%U` – неделя года (00–53), первое воскресенье считается первым днем первой недели
- `%w` – порядковый номер дня недели (0–6, воскресенье – 0)
- `%W` – неделя года (00–53), первый понедельник считается первым днем первой недели
- `%Y` – год с указанием века
- `%y` – год без указания века

04.09.2017 58

Временные ряды

Во многих областях деятельности людей замеры показателей проводятся не один раз, а повторяются через некоторые интервалы времени.

Такой интервал называют *интервалом выборки (sampling interval)*. А образующийся в результате выборки ряд данных называют *временным рядом (time series)*.

`ts` – специальный класс объектов для работы с данными, представляющими собой временные ряды (от *time series*). Интервалы выборки в этом случае одинаковые.

`ts()`

04.09.2017 59

Функции

Функции представляют собой поименованный программный код, состоящий из некоторого набора переменных, констант, операторов и других функций, и предназначенный для выполнения конкретных операций и задач.

Как правило (но не всегда), функции возвращают результат своего выполнения в виде объекта языка R – переменной определенного класса: вектора, списка, таблицы и т.д.

04.09.2017 60

Функции

Вызов функции и описание	Пример и результат
Арифметические функции	
abs(x) – модуль величины x	abs(-1) ⇒ 1
ceiling(x) – округление до целого в большую сторону	ceiling(9.435) ⇒ 10
floor(x) – округление до целого в меньшую сторону	floor(2.975) ⇒ 2
round(x, digits=n) – округление до указанного числа digits знаков после десятичной точки	round(5.475, 2) ⇒ 5.48
signif(x, digits=n) # округление до указанного числа digits значащих цифр	signif(3.475, 2) ⇒ 3.5
trunc(x) – округление до целого числа	trunc(4.99) ⇒ 4
exp(x) – e^x	exp(2.87) ⇒ 17.637
log(x) – логарифм натуральный x	log(3.12) ⇒ 1.137
log10(x) – логарифм десятичный x	log(3.12) ⇒ 0.494
sqrt(x) # корень квадратный x	sqrt(2.12) ⇒ 1.456
cos(x) sin(x) tan(x) acos(x) cosh(x) acosh(x) – тригонометрические функции от x	cos(1.27*pi) ⇒ -0.661
Функции для работы с символьными типами данных	
grep(pattern, x, ignore.case=FALSE, fixed=FALSE) – возврат индекса первого найденного элемента pattern в x	grep("А", c("x", "y", "А", "z"), fixed=TRUE) ⇒ 3
substr(x, start=n1, stop=n2) – выбор или замена символов в строках символьного вектора x	substr("язык R", 2, 4) ⇒ "зык"
paste(..., sep="") – объединение символов или строк через значение разделителя sep	paste("x", 1:3, sep="") ⇒ "x1" "x2" "x3"
strsplit(x, split) – разделяет элементы вектора по разделителю split	strsplit("абв", "") ⇒ "a" "б" "в"
toupper(x) и tolower(x) – преобразуют буквы текстового вектора x в прописные и обратно	toupper("Мал") ⇒ "МАЛ" toupper("БАЛ") ⇒ "БАЛ"

04.09.2017 61

Функции

Выражения `expr`, состоящие из объектов данных, вызовов функций и других операторов языка могут группироваться в фигурных скобках: `{expr_1; ...; expr_m}`.

Значение, которое возвращает эта группа, представляет собой результат выполнения последнего выражения.

```
имя_функции <- function(arg1, arg2,...) {
  группа_выражений
  return(object)
}
```

04.09.2017 61

04.09.2017 62

Функции

```
source()
```

```
my_examp1 <- function(n, func_trans)
{ x <- 1:n ; abs(func_trans(x)) }
```

Тогда сгенерировать 5 прологарифмированных значений можно, если записать:

```
my_examp1(5, log)
```

```
[1] 0.0000000 0.6931472 1.0986123
1.3862944 1.6094379
```


04.09.2017 62

04.09.2017 63

Условия и циклы

```
if( логическое_выражение )  
{ группа_выражений_1 если  
логическое_выражение равно TRUE }  
else { группа_выражений_2 в противном  
случае }
```


```
ifelse(логическое_выражение,  
группа_выражений_1,  
группа_выражений_2)
```



04.09.2017 64

Условия и циклы

- `for (index in for_object) {
группа_выражений }`
- `while(логическое_выражение) {
группа_выражений }`
- `repeat { группа_выражений ; break }`




04.09.2017 65

Векторизованные вычисления в R

Правила переписывания (recycling rules):

- Длина результата совпадает с длиной операнда наибольшей длины.
- Если длина операнда меньшей длины делит длину второго операнда, то такой операнд повторяется (переписывается) столько раз, сколько нужно до достижения длины второго операнда. После этого операция производится покомпонентно над операндами одинаковой длины.
- Если длина операнда меньшей длины не является делителем длины второго операнда (то есть она не укладывается целое число раз в длину большего операнда), то такой операнд повторяется столько раз, сколько нужно для перекрытия длины второго операнда. Лишние элементы отбрасываются, производится операция и выводится предупреждение.




04.09.2017 66

Векторизованные вычисления в R

```
> 2 + c(3, 5, 7, 11)
[1] 5 7 9 13

> c(1, 2) + c(3, 5, 7, 11)
[1] 4 7 8 13


> c(1, 2, 3) + c(3, 5, 7, 11)
[1] 4 7 10 12
Warning message:
In c(1, 2, 3) + c(3, 5, 7, 11) :
longer object length is not a multiple of
shorter object length
```



04.09.2017 67

Векторизованные вычисления в R

```
p <- 1:20  
lik <- 0  
for (i in 1:length(p)) {  
  lik <- lik + log(p[i])  
}  
  
lik <- sum(log(p))
```




04.09.2017 68

Векторизованные вычисления в R

В базовой комплектации R имеется целое семейство функций, предназначенных для организации векторизованных вычислений.

В названии всех этих функций имеется слово **apply** (англ. применить), которому предшествует буква, указывающая на принцип работы той или иной функции.



04.09.2017 69

Векторизованные вычисления в R

```

apply(x, MARGIN, FUN, ...)

lapply(X, FUN, ...)

sapply(X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)


vapply(X, FUN, FUN.VALUE, ..., USE.NAMES = TRUE)

mapply(FUN, ..., MoreArgs = NULL, SIMPLIFY = TRUE, USE.NAMES = TRUE)

rapply(object, f, classes = "ANY", deflt = NULL, how = c("unlist", "replace", "list"), ...)

tapply(X, INDEX, FUN = NULL, ..., simplify = TRUE)

```



04.09.2017 70

Векторизованные вычисления в R


```

replicate(n, expr, simplify = "array")

outer(X, Y, FUN = "*", ...)

do.call(what, args, quote = FALSE, envir = parent.frame())

```



04.09.2017 71

Основная литература

- Мاستицкий, С. Э. Статистический анализ и визуализация данных с помощью R [Электронный ресурс] / С. Э. Маститский, В. К. Шитиков. – 2014. – Режим доступа: <http://www.ievbras.ru/ecostat/Kiril/R/Mastitsky%20and%20Shitikov%202014.pdf>. – Дата доступа: 01.09.2016.
- Шипунов, А. Б. Наглядная статистика. Используем R! [Электронный ресурс] / А. Б. Шипунов, Е. М. Балдин, П. А. Волкова, А. И. Коробейников, С. А. Назарова, С. В. Петров, В. Г. Суфиянов. – 2014. – Режим доступа: <https://cran.r-project.org/doc/contrib/Shipunov-rbook.pdf>. – Дата доступа: 01.09.2016.
- Grolemond, G. R for Data Science [Electronic resource] / Garrett Grolemond, Hadley Wickham. – 2016. – Mode of access: <http://r4ds.had.co.nz/index.html>. – Date of access: 01.09.2016.

