

Лукьянов Павел Борисович
профессор департамента Анализа данных, принятия решений
и финансовых технологий

Программирование в среде R

Лекция 1

Назначение и особенности языка R. Решаемые задачи

Финансовый университет, 2020

<https://www.youtube.com/watch?v=Og847HxwRSI>

Популярность языков программирования

2

Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C++		99.7
3. Java		97.5
4. C		96.7
5. C#		89.4
6. PHP		84.9
7. R		82.9
8. JavaScript		82.6
9. Go		76.4
10. Assembly		74.1

- Развитие общества → Развитие ИТ-технологий → Рост эффективности любой деятельности при ее автоматизации → Рост количества задач для автоматизации → Спрос на специалистов по автоматизации решения задач
- Разновидности задач:
 - Простые, повторяющиеся
 - Сложные, ресурсоемкие
 - Локальные
 - Распределенные
 - Оптимизационные
 - Решаемые в реальном времени
 - ...
- В зависимости от постановки, от особенностей задачи для ее решения требуются **различные инструменты** - Средства автоматизации (компьютерные программы)
- Если готового средства автоматизации нет, используют языки программирования для создания необходимых программ
- Т.о., любой **язык программирования** – всего лишь **инструмент** для решения определенных задач с помощью компьютера



Visual Basic – изначально простой язык для не-программистов

C – сложный язык для разработки операционных систем, больших программ, замена Assembler

C++ – развитие языка С с добавлением объектно-ориентированного подхода

Java – упрощение C++, независимость от архитектуры компьютеров

C# – конкурент Java, скопирован с Java фирмой Microsoft, работает на OS Windows

Pascal – язык для обучения программированию

Python – современный быстроразвивающийся язык: web-разработка, системное администрирование, прикладное ПО, игры и т.д.

Решения всегда принимаются на основе анализа данных

3 уровень

ЛПР – Лицо, Принимающее Решение (начальник, директор, руководитель, ...)

Управленческое решение

2 уровень

Анализ данных, обзоры, выводы

Аналитик, эксперт, консультант

Анализ данных, обзоры, выводы

Анализ данных, обзоры, выводы

Аналитик, эксперт, консультант

1 уровень

Данные

Исполнитель

Исполнитель

Исполнитель

Исполнитель

Исполнитель

В некоторой предметной области накоплено много «сырых» данных

- Данные копятся по каждой операции, каждому действию миллионов людей
- Стоимость получения и хранения данных постоянно дешевеет
- Данные содержат ошибки, пропуски, выбросы, дубликаты и т.д.
- В массивах данных скрыты некоторые важные зависимости, взаимосвязи

Ставится задача по очистке данных и нахождению зависимостей и взаимосвязей

- Понимание предметной области, главных и второстепенных факторов
- Знание методов очистки данных
- Владение методами обработки данных, их анализа
- Знание алгоритмов поиска решений, оптимизации управленческих решений
- Модификация алгоритмов и типовых решений под задачу

Нужны инструментальные средства для автоматизации решения задачи

- Стандартные решения: готовые программы, пакеты, сервисы
- Доработка стандартных решений под поставленную задачу
- Разработка инструментального средства для решения задачи

Практическая реализация алгоритмов, обработка данных

Анализ полученных результатов

Тиражирование решения

Решения формируются ЛПР на основе данных различной природы

Что учитывает ЛПР? Например:

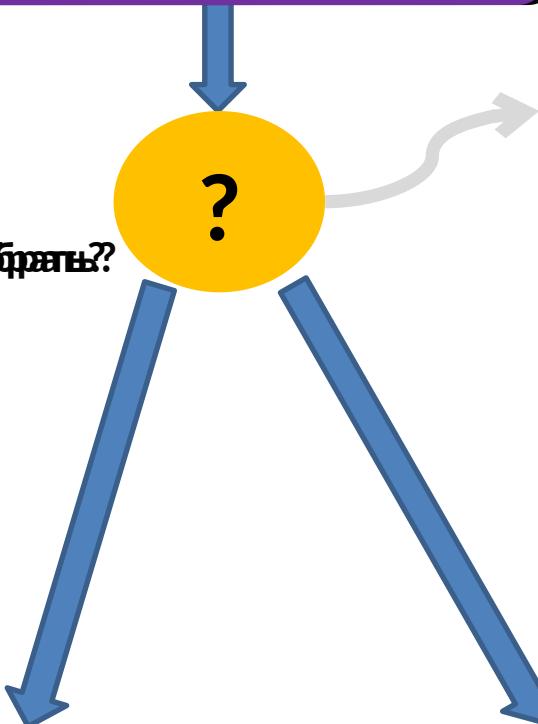
- Объемы выпуска и продаж
- Цены и предложения конкурентов
- Логистика и сроки доставки
- Надежность Поставщиков и Клиентов
- Качество и стоимость сырья
- Налоги, кредиты
- Сезонность потребления
- Текущий и прогнозируемый курсы валют
- Обязательства по ранее заключенным договорам и т.д.

Требуется оптимальное решение

- Среди десятков и сотен вариантов **управленческих решений** существуют **оптимальные решения**
- Решение будет тем более оптимальным, чем большее количество зависимостей и связей между всеми данными оно учитывает
- Необходимо **программное обеспечение (ПО)** для выявления и анализа зависимостей и связей между данными

Программное обеспечение –
инструмент для решения задач
с помощью компьютера

Какое ПО выбрать?



Стандартное ПО

- Калькуляторы, электронные таблицы

Специализированное ПО

- Современные языки программирования
- Сложные программные комплексы и системы
- Среда (язык) R

Решение фундаментальных задач

Получение новых знаний об
окружающем мире

Решение прикладных задач

Вычисления, выявление и представление
 зависимостей, анализ, оптимизация деятельности

Стандартное программное обеспечение



Калькулятор: четыре арифметических действия, квадратные корни и степени, логарифмы. Достаточно для простейшей обработки данных.



Недостаток: сложность работы с сериями чисел; обычно данные идут сериями (колонками, векторами).



Электронные таблицы: наглядность, развитый инструментарий для ввода и преобразования данных — автодополнение, автокопирование, сортировка, наличие математических функций, графики



Недостатки:

- Ориентация на типовые задачи офисного применения, недостаточно инструментов для всестороннего анализа
- Отсутствуют методы для обработки многомерных данных
- Скрытая реализация алгоритмов и ограничения
- Отсутствие гибкости в формировании сложных графиков и отчетов
- Трудности при обработке данных разной природы

Примеры: [MS Excel](#), [Gnumeric](#), [OpenOffice.org Calc](#)

Специализированное программное обеспечение

Оконно-кнопочные системы

Среды и языки программирования

ПО для решения конкретных специализированных задач

Специализированное программное обеспечение

Оконно-кнопочные системы

- **STATISTICA** платная, алгоритмы вычислений закрыты, необходимость писать макросы
- **STADIA** платная, закрытая
- **PAST** бесплатная
- **SPSS** платная
- **MiniTab** платная
- **StatGraphics** платная



Достоинства

- Готовые сценарии и алгоритмы обработки данных
- Возможность конструирования своих сценариев обработки данных



Недостатки

- Отсутствие гибкости при обработке данных
- Закрытость алгоритмов

Специализированное программное обеспечение

Среды и языки программирования



Особенности

- Использование **интерфейса командной строки**: Пользователь вводит команду, система отвечает. Четкость работы, минимализм
- Поддержка **графического интерфейса**: система меню, работа с несколькими окнами
- Возможность реализации сложных технологий обработки данных
- Полный контроль над системой:
 - объединение любых видов анализа
 - работа со скриптами
 - интерактивные графики
 - вывод в любые графические форматы и т.д.
- Возможность расширения системы
- Модификация самой системы в случае ее **открытого кода**
- Прозрачность работы алгоритмов
- Требуются навыки написания алгоритмов
- Необходимость знания языка программирования
- Знание особенностей среды разработки
- Высокий порог входления



Специализированные статистические программы

Среды и языки программирования

SAS – коммерческая система, лидер, существует с 70-х годов 20 века

Stata, SYSTAT - конкуренты

S-Plus – язык **S** и среда для выполнения расчетов, платная

R – развитие языка **S**, появился в 90-х годах. Два разработчика: **Robert Gentleman** и **Ross Ihaka**.

Применяется во всех областях, где нужна обработка данных:

- первичный анализ
- статистический анализ
- математическое моделирование

Преимущества **R**: гибкость и открытый код, бурное развитие, поддержка независимых разработчиков во всем мире (создают пакеты)

Среда **R** на сегодня – фактический стандарт в сообществе ученых и аналитиков во всех областях человеческой деятельности, в том числе и в экономике

R для решения фундаментальных задач

Получение новых знаний

- Выявление и исследование зависимостей
- Изучение закономерностей
- Построение моделей поведения процессов и систем
- Проверка гипотез
- Прогнозирование
- Оценка и анализ состояния объектов, систем, процессов

R для решения прикладных задач

Вычисления, анализ, представление

- Сложные статистические расчеты
- Проверка, структурирование информации
- Анализ данных различной природы
- Выявление и представление зависимостей
- Поддержка принятия решений

Повышение эффективности деятельности в различных областях

Определение

R – язык программирования для статистической обработки данных и работы с графикой, а также свободная программная среда вычислений с открытым исходным кодом

Особенности

- **Бесплатность**

Официальное сообщество – The Comprehensive R Archive Network (CRAN)

Официальный сайт – <https://cran.r-project.org/> + сотни зеркал по всему миру

- **Кроссплатформенность:** возможность исходному коду быть запущенным и отработать с одинаковым предсказуемым результатом на компьютерах с разными ОС

- **R доступен для всех основных операционных систем:** FreeBSD, Solaris, Unix, Linux, Microsoft Windows, Mac OS X, см. <https://cran.r-project.org/>

- **Открытый исходный код. R – свободное программное обеспечение**

Доступен исходный код как самого языка R, так и всех алгоритмов, функций на R
<https://cran.r-project.org/>

- **Постоянное развитие языка**

Периодически выходят новые версии, исправляются ошибки, появляются новые функции <https://cran.r-project.org/>

- **Постоянное расширение возможностей языка**

Сотни тысяч разработчиков и ученых создают пакеты на R на все случаи жизни и выкладывают их в общий доступ <https://cran.r-project.org/>

Особенности

- **Интерфейс командной строки**

Изначально все команды и операторы набираются вручную, вводятся в консоль. Пользователь перед вводом каждой команды понимает, зачем он это делает.

- **Графический интерфейс Пользователя**

RCommander, RKWard, RStudio, Weka, Rapid Miner, KNIME и др.

- **Режим интерпретатора**

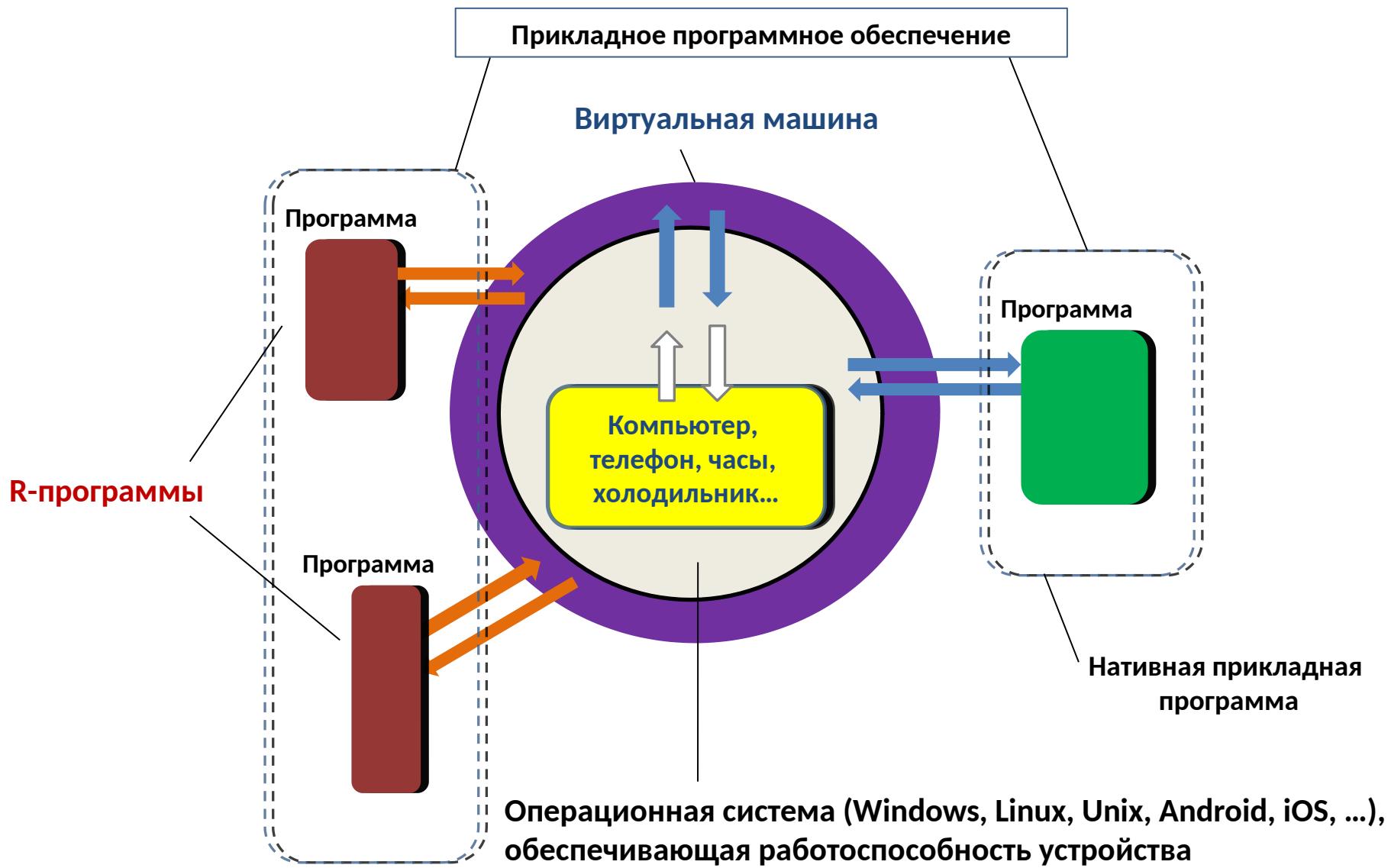
Код R выполняется последовательно, строчка за строчкой – так, как написал программист

- **Компиляция в байт-код**

В последних версиях R поставляется компилятор байт-кода и JIT-компилятор

- **Объектно-ориентированный подход**

Любая переменная является объектом. Объект может сам манипулировать со своими данными



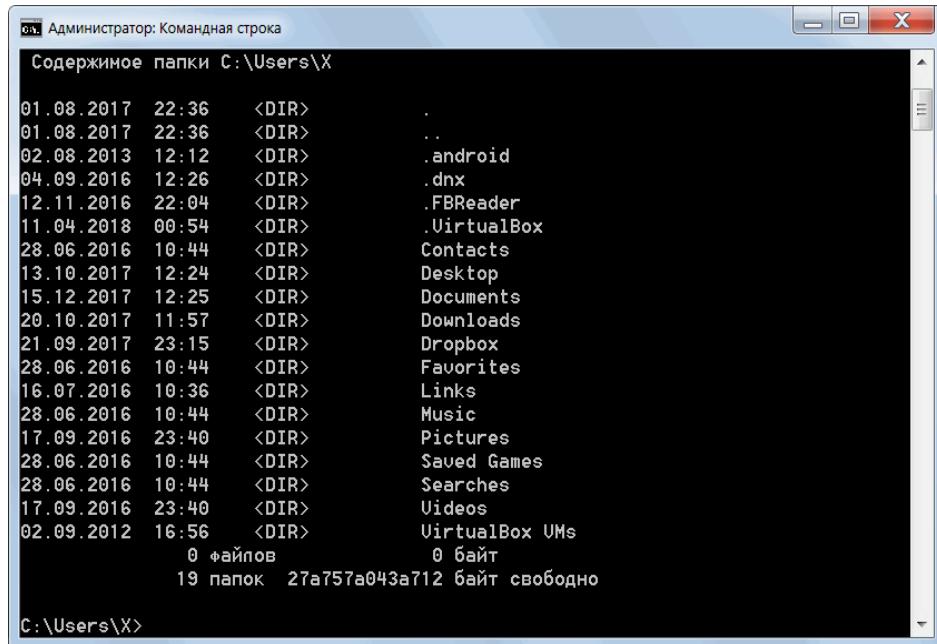
Режим интерпретатора

- все команды программы на языке R выполняются **последовательно**, строчка за строчкой, именно в том порядке, как написал эти строчки программист.
- Основной недостаток – **медленная работа**, отсутствие компиляции по сравнению с работой нативных программ.
- Попытки оптимизации кода – **реализация динамической компиляции**: байт-код перед исполнением обрабатывается JIT-компилятором (just-in-time compiler)

Интерфейс командной строки

Интерфейс командной строки. (англ. Command line interface, CLI) — способ общения пользователя с программой, при котором используется только текстовый интерфейс для передачи команд программе и для просмотра результатов работы программы.

Также используется синоним для обозначения этого режима – «работа с консолью».



Администратор: Командная строка
Содержимое папки C:\Users\X

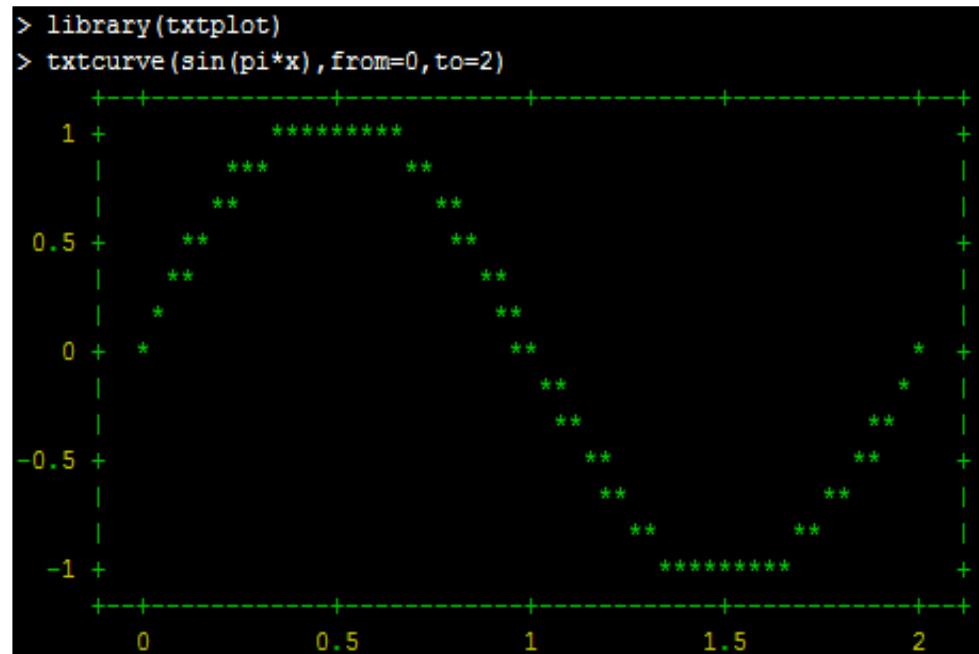
Дата	Время	Тип	Название
01.08.2017	22:36	<DIR>	.
01.08.2017	22:36	<DIR>	..
02.08.2013	12:12	<DIR>	.android
04.09.2016	12:26	<DIR>	.dnx
12.11.2016	22:04	<DIR>	.FBReader
11.04.2018	00:54	<DIR>	.VirtualBox
28.06.2016	10:44	<DIR>	Contacts
13.10.2017	12:24	<DIR>	Desktop
15.12.2017	12:25	<DIR>	Documents
20.10.2017	11:57	<DIR>	Downloads
21.09.2017	23:15	<DIR>	Dropbox
28.06.2016	10:44	<DIR>	Favorites
16.07.2016	10:36	<DIR>	Links
28.06.2016	10:44	<DIR>	Music
17.09.2016	23:40	<DIR>	Pictures
28.06.2016	10:44	<DIR>	Saved Games
28.06.2016	10:44	<DIR>	Searches
17.09.2016	23:40	<DIR>	Videos
02.09.2012	16:56	<DIR>	VirtualBox VMs
	0 файлов		0 байт
	19 папок		27a757a043a712 байт свободно

C:\Users\X>

Для ОС Windows переход в режим CLI:

меню «Пуск» → в окошке «Найти программы и файлы» набрать текст «командная строка» → из выпадающего списка выберите эту строчку

```
> # The colors are customizable:  
> setOutputColors()  
normal, number, negnum, date, string, const, stderr, warn, error.  
> setOutputColors256(normal = 39, number = 51, negnum = 183, date = 43,  
+                      string = 79, const = 75, verbose = FALSE)  
> x  
  logic factor string  real    cien.not      date  
1 TRUE     abc     ABC  1.23  1.234e-23 2012-02-21  
2 TRUE     def     DEF -4.56 -4.560e+45 2013-02-12  
3 FALSE    ghi     GHI  7.89  7.890e+78 2014-03-04  
> setOutputColors256(202, 214, 209, 184, 172, 179, verbose = FALSE)  
> x  
  logic factor string  real    cien.not      date  
1 TRUE     abc     ABC  1.23  1.234e-23 2012-02-21  
2 TRUE     def     DEF -4.56 -4.560e+45 2013-02-12  
3 FALSE    ghi     GHI  7.89  7.890e+78 2014-03-04  
>
```



RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

+ - Addins Go to file/function Addins

Untitled1*

```

12 v2 <- FALSE
13 r2 <- v == q
14 r2
15 w <- 'a'
16 r<- q < w
17 s<- v==q
18 s
19 r
20 q
21 w
22 q1 <-(2:4)
23 q2 <-(0:2)
24 q3 <- (q1 == 2) || (q2 ==2)
25 q3
26
26:1 (Top Level) R Script

```

Environment History Connections

Import Dataset package:graphics

Values

abline	<Promise>
arrows	<Promise>
assocplot	<Promise>
axis	<Promise>
axis.Date	<Promise>
axis.POSIXct	<Promise>
axTicks	<Promise>
barplot.def...	<Promise>
box	<Promise>
boxplot.def...	<Promise>
boxplot.mat...	<Promise>
bxp	<Promise>

Console Terminal

Type 'demo()' for some demos, 'help()' for on-line help, or 'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

>

Files Plots Packages Help Viewer

Install Update

Name	Description	Version
boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-20
class	Functions for Classification	7.3-14
cluster	"Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.	2.0.7-1
codetools	Code Analysis Tools for R	0.2-15
compiler	The R Compiler Package	3.5.0
datasets	The R Datasets Package	3.5.0
foreign	Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ...	0.8-70
graphics	The R Graphics Package	3.5.0
grDevices	The R Graphics Devices and Support for Colours and Fonts	3.5.0
grid	The Grid Graphics Package	3.5.0
KernSmooth	Functions for Kernel Smoothing Supporting Wand & Jones (1995)	2.23-15
lattice	Trellis Graphics for R	0.20-35
MASS	Support Functions and Datasets for Venables and Ripley's MASS	7.3-49
Matrix	Sparse and Dense Matrix Classes and Methods	1.2-14
methods	Formal Methods and Classes	3.5.0



Определение

Свободное программное обеспечение (free software, software libre, libre software) – это ПО, Пользователи которого имеют следующие права (четыре свободы):

1. Программу можно использовать для любых целей
2. Можно изучать работу программы и адаптировать её для своих целей
3. Можно распространять копии программы
4. Программу можно улучшать и публиковать свою улучшенную версию

Для реализации 2 и 4 свобод необходима доступность исходного текста программы и возможность внесения в него модификаций и исправлений

Если на ПО есть исключительные права, то свободы объявляются при помощи свободных лицензий (GNU General Public License)

Определение

Открытое программное обеспечение (open-source software) – программное обеспечение с открытым исходным кодом. Код доступен для просмотра, изучения и изменения.

Открытое ПО – ключевой признак Свободного ПО.

Определение

**Проприетарное программное обеспечение,
несвободное программное обеспечение**

(proprietary software) — программное обеспечение, являющееся частной собственностью авторов или правообладателей и не удовлетворяющее критериям свободного ПО.

Правообладатель проприетарного ПО сохраняет за собой монополию на

- его использование
- тиражирование
- модификацию

Проприетарное ПО

Собственник ПО может предоставлять некоторые гарантии и сервисы

- Служба контроля качества ПО
- Подробная документация
- Служба поддержки
 - Консультации по телефону, почте, ответы на форумах
 - Обновления, исправление ошибок
 - Развитие программы, выход новых версий

Свободное ПО

Программа поставляется «как есть»

- Автор не несет никаких обязательств перед Пользователем:
не обязан объяснять, как работает программа,
не обязан отвечать на вопросы
- Пользователь работает с программой на свой страх и риск

Что делать Пользователю?

- Разбираться в чужом исходном коде
- Находить и исправлять ошибки
- Улучшать программу под себя и свои задачи

Лукьянов Павел Борисович
профессор департамента Анализа данных, принятия решений
и финансовых технологий

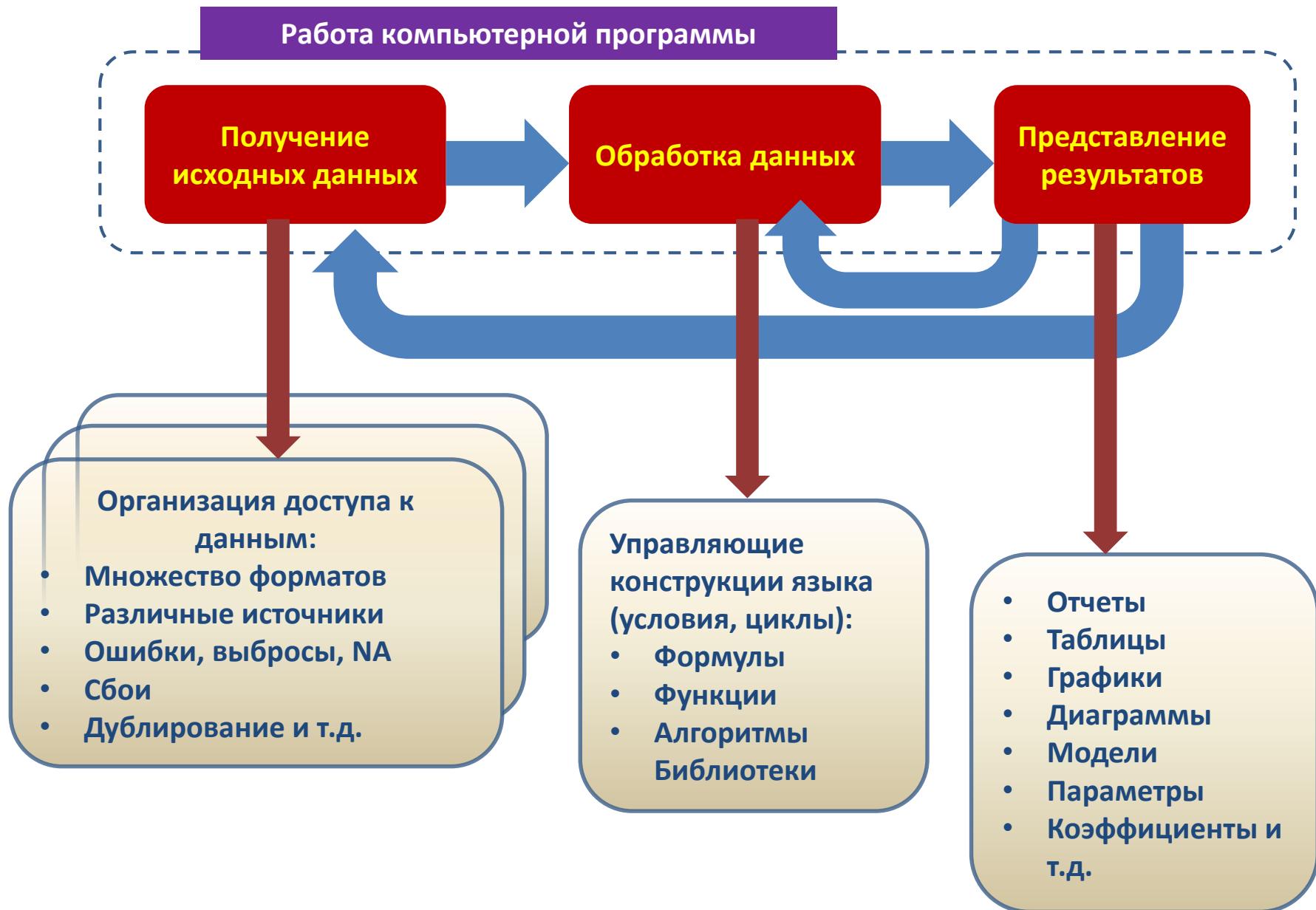
Программирование в среде R

Лекция 2

Векторы. Контейнеры.
Управляющие конструкции языка.

Финансовый университет, 2020

Работа компьютерной программы – цикл типовых действий



Использование данных в программе

Данные разных форматов из различных источников. Как они хранятся?

- Хранение данных в программе: контейнеры, специальные структуры
- Обработка данных: специальные функции

Функции анализа и представления данных

Таблицы

Информация регистрируется и сохраняется как последовательность однотипных данных

Деревья

Информация задается как множество узлов и связей между ними

Какие задачи нужно решить

- Нужно передать в программу массивы однотипных данных
- Данные требуется обрабатывать быстро и просто, единым образом
- В идеале: одним действием или вызовом одной функции обработать весь массив данных
- Базовых типов языка (`integer`, `double`, `character`, `logical`) недостаточно
- Для хранения массивов данных одного типа в R реализована структура (контейнер) данных – **vector**



Объект Вектор (vector)

Вектор (vector)



Вектор – объект, объединяющий элементы одного типа

Вектор – именованный одномерный объект, содержащий набор однотипных элементов (или числовые, или логические, или текстовые значения)

Вектор – одномерный массив, характеризуется

- Именем
- Длиной
- Типом хранимых данных

Тип вектора x проверяется функциями `typeof(x)`, `str(x)`, `mode(x)`
длина вектора проверяется `length(x)`

Элементы вектора проиндексированы, к каждому элементу вектора x можно обратиться по его индексу, используя квадратные скобки []: $x[i]$

В R индексация векторов, массивов и т.д. всегда начинается с 1:

$x[1]$ – первый элемент

$x[2]$ – второй элемент

$x[length(x)]$ – последний элемент

В R вообще нет скалярных величин

Следовательно, в R все изначально создаваемые объекты – уже **векторы**

Все данные передаются через параметры функций

Данные разных форматов из различных источников

Сырые данные

Очищенные данные

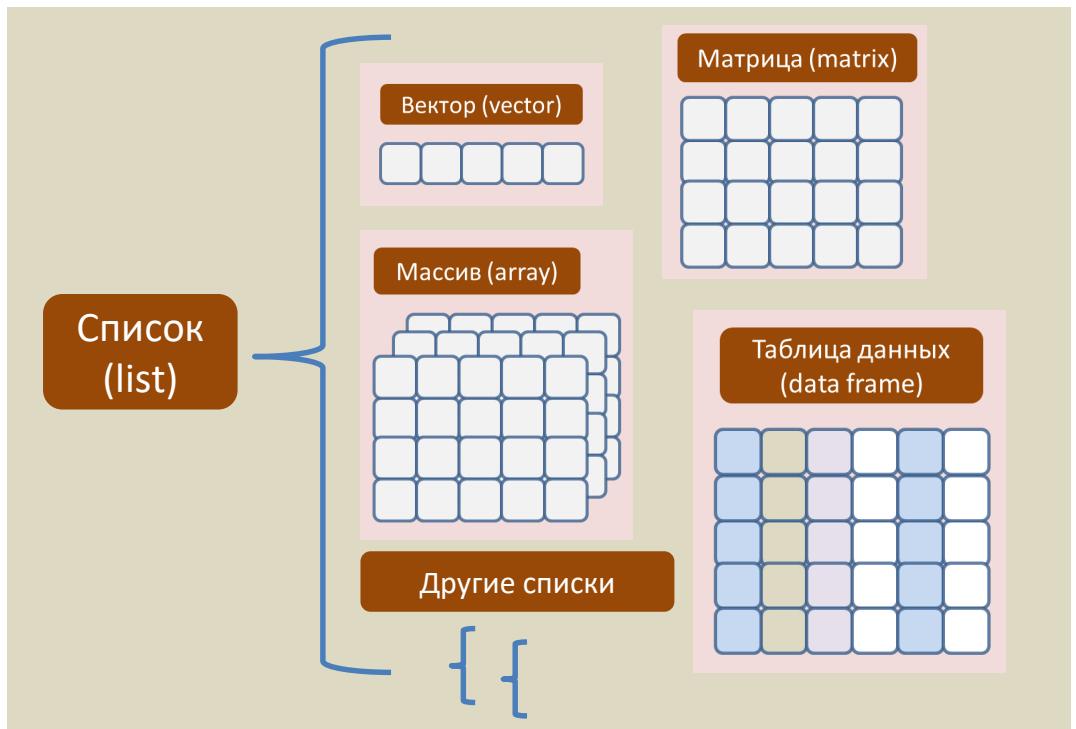
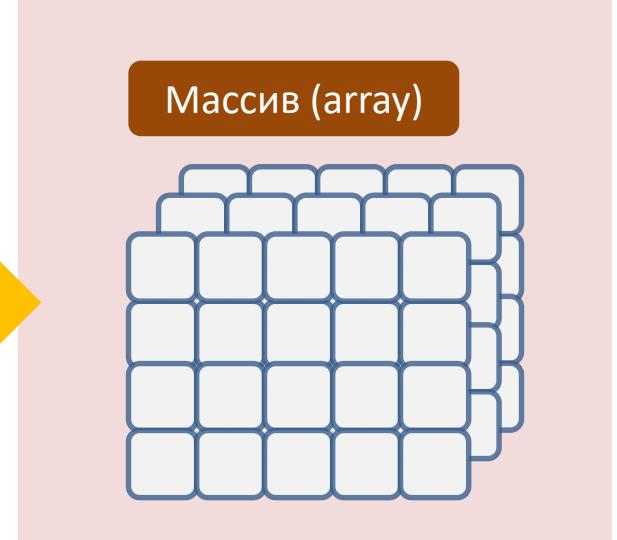
По каким признакам
определить качество данных?

Контейнеры разной структуры

Данные разных форматов из различных источников

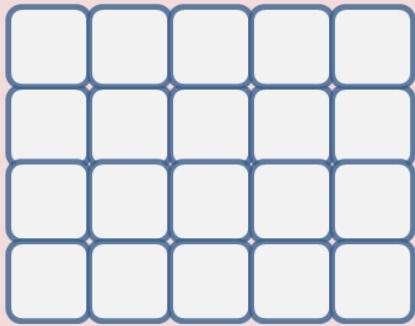


Контейнеры R для хранения различных структур данных



Объект Матрица (matrix)

Матрица (matrix)



Работа с матрицей

`m1 <- cbind(a,b,c)` (column bind – связывать столбцы)

`m2 <- rbind(a,b,c)` (row bind – связывать строки)

`t(m)` – транспонирование матрицы (transpose)

`m3 <- matrix(вектор, число_строк, число_столбцов)`

`rownames(), colnames()` – задаем заголовки для строк и столбцов

`m4 <- 1:12`

Создаем матрицу из различных векторов

Создаем матрицу из одного вектора

Создаем матрицу через функцию `dim()`

`m4 <- dim(число_строк, число_столбцов)`

`class(m4)` – возвращает тип объекта (вектор, матрица)

`m[3,]` – выделяем все столбцы
`m[, 4:5]` – выделяем все строчки
`m[,]` – что будет выведено?
`m[-3:-5, 'c']` – что будет выведено?
`m[-3:-5]` – что будет выведено?

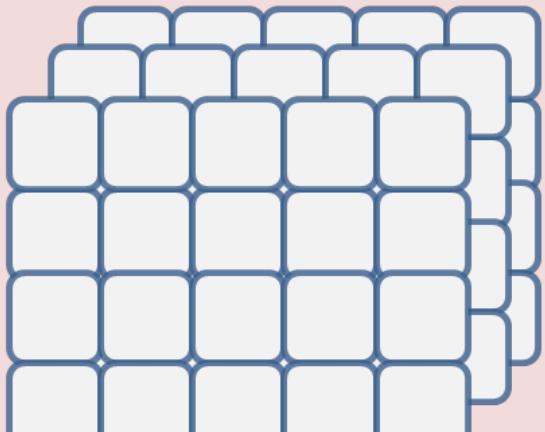
Объект Массив (array)

Массив – именованный объект размерности N ($N = 1, 2, 3, \dots$), содержащий набор однотипных элементов (числовые, логические, или текстовые значения)

Массив – объединение векторов, следовательно, все функции, работающие с векторами, применимы к матрице

Элементы массива проиндексированы, к каждому элементу можно обратиться по его индексу

Массив (array)



Массив создается **функцией array()** с указанием
размерности массива

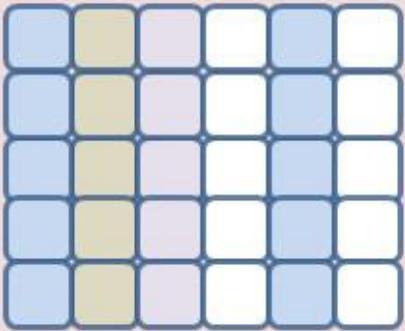
Представление данных различной природы в таблице

Пример. Поиск дачи: анализ показателей, влияющих на решение о покупке

№ объекта	Удаленность от города, км	Качество дороги	Наличие жд станции	Площадь участка, сот	Забор	Площадь дома, кв. м	Цена вопроса, млн. руб	Кооператив / Деревня	Охрана
1	90	отличное	Нет	12	Профнастил	70	2.2	Кооператив	Нет
2	78	отличное	Нет	10	Рабица	78	2.5	Кооператив	Нет
3	102	хорошее	Есть	16	Нет	58	1.7	Деревня	Нет
4	88	хорошее	Нет	15	Евроштакетник	90	2.8	Кооператив	Есть
5	110	плохое	Есть	22	Старый	95	1.4	Деревня	Нет
6	87	среднее	Нет	15	Рабица	80	1.9	Деревня	Нет
7	97	среднее	Нет	15	Профнастил	68	2.0	Кооператив	Есть
8	71	отличное	Нет	12	Рабица	110	2.5	Кооператив	Есть
9	100	отличное	Есть	20	Деревянный	80	2.2	Деревня	Нет
10	84	хорошее	Нет	14	Деревянный	90	2.0	Кооператив	Нет

Таблица данных (data frame)

Таблица данных
(data frame)



- **Таблица данных (data frame)** – самая используемая структура данных в R
- Разные столбцы могут содержать разные типы данных (числовой, текстовый и т. д.)
- Столбцы таблицы формируются из заранее подготовленных векторов разного типа
- Каждому столбцу может быть присвоено имя
- Столбцы в R также называют *переменными*

Создание контейнера типа data frame

```
создание таблицы.R — Блокнот
Файл Правка Формат Вид Справка
# загоняем данные столбцов из таблицы в вектора
ID <- c(1:10)
distance <- c(90,78, 102, 88, 110, 87, 97, 71, 100, 84)
road <- c("отличное", "отличное", "хорошее", "хорошее", "плохое", "среднее",
"среднее", "отличное", "отличное", "хорошее")
railway <- c(FALSE, FALSE, TRUE, FALSE, TRUE, FALSE, FALSE, TRUE, FALSE, FALSE)
piece <- c(12, 10, 16, 15, 22, 15, 15, 12, 20, 14)

# объединяем вектора в таблицу
campData <- data.frame(ID, distance, road, railway, piece)

# с нетерпением смотрим, что получилось
campData
```

Результат					
	ID	distance	road	railway	piece
1	1	90	отличное	FALSE	12
2	2	78	отличное	FALSE	10
3	3	102	хорошее	TRUE	16
4	4	88	хорошее	FALSE	15
5	5	110	плохое	TRUE	22
6	6	87	среднее	FALSE	15
7	7	97	среднее	FALSE	15
8	8	71	отличное	FALSE	12
9	9	100	отличное	TRUE	20
10	10	84	хорошее	FALSE	14

Объект Список (list)

Список

Векторы

Матрицы

Массивы

Таблицы данных

Другие списки

Список (list) – вектор с элементами произвольного типа.
Элементами списка могут быть данные **любых типов**.

Список (list) – именованный объект, содержащий набор элементов разного типа (числовые, логические, или текстовые значения).

Список позволяет хранить разнородную информацию в одном объекте.

Как правило, результаты наблюдений, экспериментов хранятся в объектах-списках, которые содержат текстовые значения, числа, факторы и т.д.

Работа со списком

Списки создаются одноименной функцией list()

```
# создадим три вектора разного типа данных  
# текст, числа, логические значения:
```

```
a <- c("A", "B", "C", "D")  
b <- c(1, 3, 0.5, -12, 77, 5.1)  
c <- c(FALSE, TRUE)
```

```
# объединим три вектора в один объект-список,  
# частям (компонентам) которого присвоим имена:
```

```
l1 <- list(TextName=a, NumberName=b, LogicName=c)
```

```
# Просмотрим содержимое созданного списка:
```

```
l1
```

```
> l1  
$TextName  
[1] "A" "B" "C" "D"  
  
$NumberName  
[1] 1.0 3.0 0.5 -12.0 77.0 5.1  
  
$LogicName  
[1] FALSE TRUE
```

Одномерный анализ

Одномерный анализ – анализ объекта или явления по одному, главному показателю

Анализируемый объект рассматривается, анализируется и оценивается только **по одному признаку**, по одной шкале, соответствующей выбранному аналитическому показателю:

- Товары – **по прибыли**
- Складские остатки – **по средней стоимости**
- Страны – **по уровню ВВП**
- Студенты – **по успеваемости**
- Семьи – **по количеству детей**
- Сотрудники – **по зарплате**
- Олигархи – **по длине яхты**
- Новорожденные – **по весу**
- Долгожители – **по возрасту**
- Паращитисты – **по весу**

Переходим Контрольной работе № 1

Плюсы

- Исследуется наиболее значимый показатель
- Анализ выполняется просто, быстро
- Имеется развитый инструментарий
- Самый популярный вид анализа

Минусы

- Нередко достаточно грубая, качественная оценка
- Не учитывается влияние сопутствующих и связанных показателей
- Нет ответа на вопрос «**Почему?**»

Контейнер данных для анализа в R: вектор

```
skyDiver <- c(72, 81, 64, 49, 59, 77, 84)
family <- c(0, 0, 2, 1, 1, 0, 3, 2, 2, 1, 4, 0, 1, 2)
gender <-c("муж", "жен", "жен", "муж", "жен", "муж" , "муж", "муж")
```

Структура Вектор

Вектор



Вектор – объект, объединяющий элементы одного типа

Вектор – именованный одномерный объект, содержащий набор однотипных элементов (или числовые, или логические, или текстовые значения)

Вектор – одномерный массив, характеризуется

- Именем
- Длиной
- Типом хранимых данных

Тип вектора x проверяется функциями `typeof(x)`, `str(x)`, `mode(x)`
длина вектора проверяется `length(x)`

Элементы вектора проиндексированы, к каждому элементу вектора можно обратиться по его индексу, используя квадратные скобки []

В R индексация векторов, массивов и т.д. всегда начинается с 1:

$x[1]$ – первый элемент

$x[2]$ – второй элемент

$x[length(x)]$ – последний элемент

В R вообще нет скалярных величин

Следовательно, изначально в R все создаваемые объекты – уже **векторы**

Переходим Контрольной работе № 1

Алгоритмы работы компьютерной программы

Как работает компьютерная программа?

Работа компьютерной программы – реализация определенных алгоритмов: очистка данных, расчеты, оптимизация параметров, поиск области решений и т.д.

Что такое алгоритм?

Алгоритм — набор инструкций, описывающих порядок действий для достижения некоторого результата

Как описывается нужный порядок действий?

1. Проверяется выполнение условий, задается ветвление
2. Выполняются многократные повторения некоторой части алгоритма (циклы)

`if, else`

`repeat, while, for,
in, next, break`

Управляющие конструкции языка

Управляющие конструкции всех языков одинаковы

Используются одни и те же ключевые слова (операторы)
if, else, repeat, while, for, in, next, loop, break

Есть отличия в правилах оформления управляющих конструкций
(наличие круглых и фигурных скобок, точек с запятыми и т.д.)

В некоторых языках есть специальные функции, расширяющие
возможности управляющих конструкций (в R – функция **ifelse()**)

Алгоритм с проверкой условия и ветвлением

Определение

Разветвляющийся алгоритм – алгоритм, в котором в зависимости от выполнения или не выполнения некоторого условия совершается выбор:

- выполняется последовательность действий № 1
- выполняется последовательность действий № 2



Синтаксис ветвления на языке R

Test Condition



TRUE or FALSE

```
if (test_expression) {  
  statement  
}
```

TRUE

TRUE or FALSE

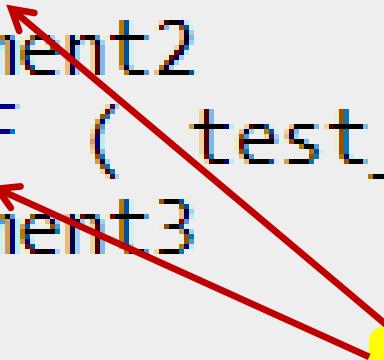
```
if (test_expression) {  
  statement1  
} else {  
  statement2  
}
```

TRUE

FALSE

Вложенные конструкции if ... else

```
if ( test_expression1) {  
    statement1  
} else if ( test_expression2) {  
    statement2  
} else if ( test_expression3) {  
    statement3  
} else  
    statement4
```



Вложенные конструкции if - else

Фигурные скобки { } можно не использовать, если выполняется только одна команда.

Если по условию нужно выполнить несколько команд, код помещаем в фигурные скобки

Вложенные конструкции if ... else

```
if ( test_expression1) {  
    statement1  
} else if ( test_expression2) {  
    statement2  
} else if ( test_expression3) {  
    statement3  
} else ←  
    statement4
```

Где фигурная закрывающая скобка?

Фигурные скобки { } можно не использовать, если выполняется только одна команда.

Если по условию нужно выполнить несколько команд, код помещаем в фигурные скобки

Вложенные конструкции if ... else. Пример

```
x <- 0
if (x < 0) {
    print("Negative number")
} else if (x > 0) {
    print("Positive number")
} else
    print("Zero")
```

Фигурные скобки { } можно не использовать, если выполняется только одна команда.

Если по условию нужно выполнить несколько команд, код помещаем в фигурные скобки

Функция ifelse() языка R для работы с векторами

ifelse(test_expression, x, y)

test_expression – логический вектор или выражение, приводимое к логическому вектору

Возвращаемые значения: **x, y** – вектора той же длины, что и **test_expression**

Если (**test_expression[i] == TRUE**), возвращается **x[i]**, иначе **y[i]**.

Пример ifelse()

```
> a = c(5,7,2,9)
> ifelse(a %% 2 == 0,"even","odd")
[1] "odd"  "odd"  "even" "odd"
```

Циклы

Цикл - управляющая конструкция, предназначенная для выполнения многократного исполнения набора инструкций

Тело цикла – набор инструкций, предназначенный для многократного исполнения

Итерация – единичное выполнение тела цикла

Условие окончания цикла – выражение, определяющее, будет ли выполняться итерация или цикл завершится

Счетчик цикла – переменная, хранящая текущий номер итерации

Как работает цикл

- первоначальная инициализация переменных цикла – делается один раз
- проверка условия выхода (1)
- исполнение тела цикла (2)
- обновление счетчика цикла после каждой итерации (3)
- (1) – (2) – (3)
-

Циклы в R. Цикл for()

Цикл for

Цикл **for () {...}** используется для итерации по вектору:

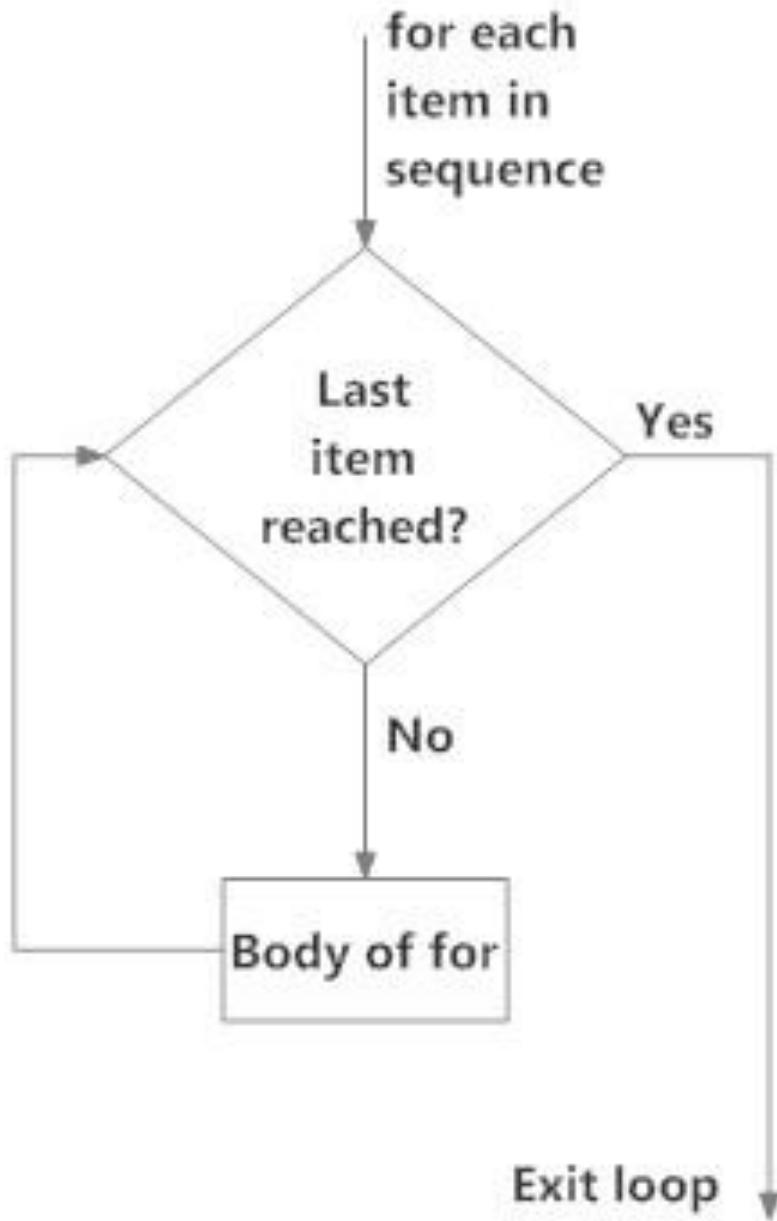
```
for (val in sequence)
{
    statement
}
```

sequence - представляет собой вектор

val последовательно принимает значения всех элементов вектора **sequence** во время выполнения цикла

На каждой итерации выполняется **statement**.

Циклы в R. Алгоритм работы цикла for()



Циклы в R. Цикл while()

while() – цикл с предусловием

Цикл с предусловием — цикл, который выполняется, пока истинно некоторое условие, указанное перед его началом

Это условие проверяется до выполнения тела цикла, поэтому тело может быть не выполнено ни разу, если предусловие ложно

В большинстве языков программирования реализуется оператором while, отсюда его второе название — while-цикл

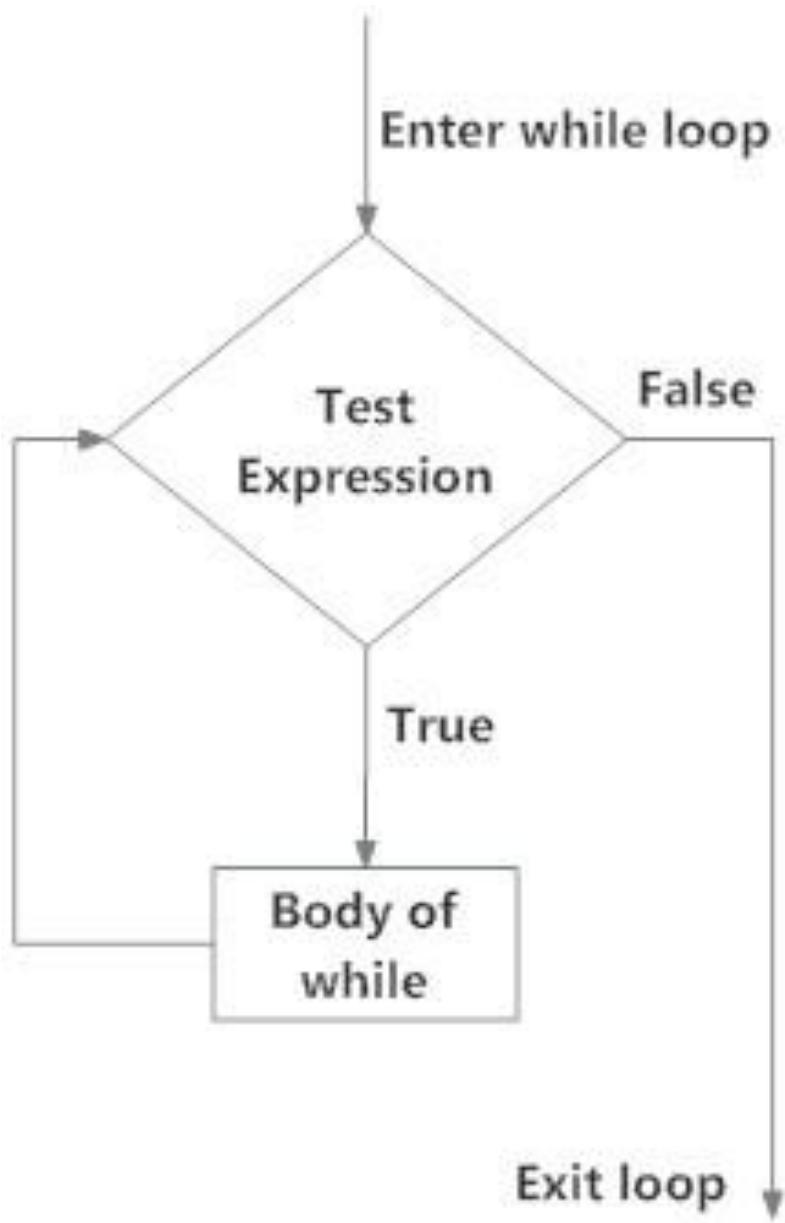
```
while (test_expression)
{
    statement
}
```

test_expression – логическое выражение, оценивается перед началом цикла
если (**test_expression == TRUE**) выполняется первая итерация, выполняется **statement**

снова оценивается **test_expression**

если (**test_expression == TRUE**), **statement** выполняется еще раз
и т.д.

Циклы в R. Алгоритм работы while()



Циклы в R. Использование оператора break

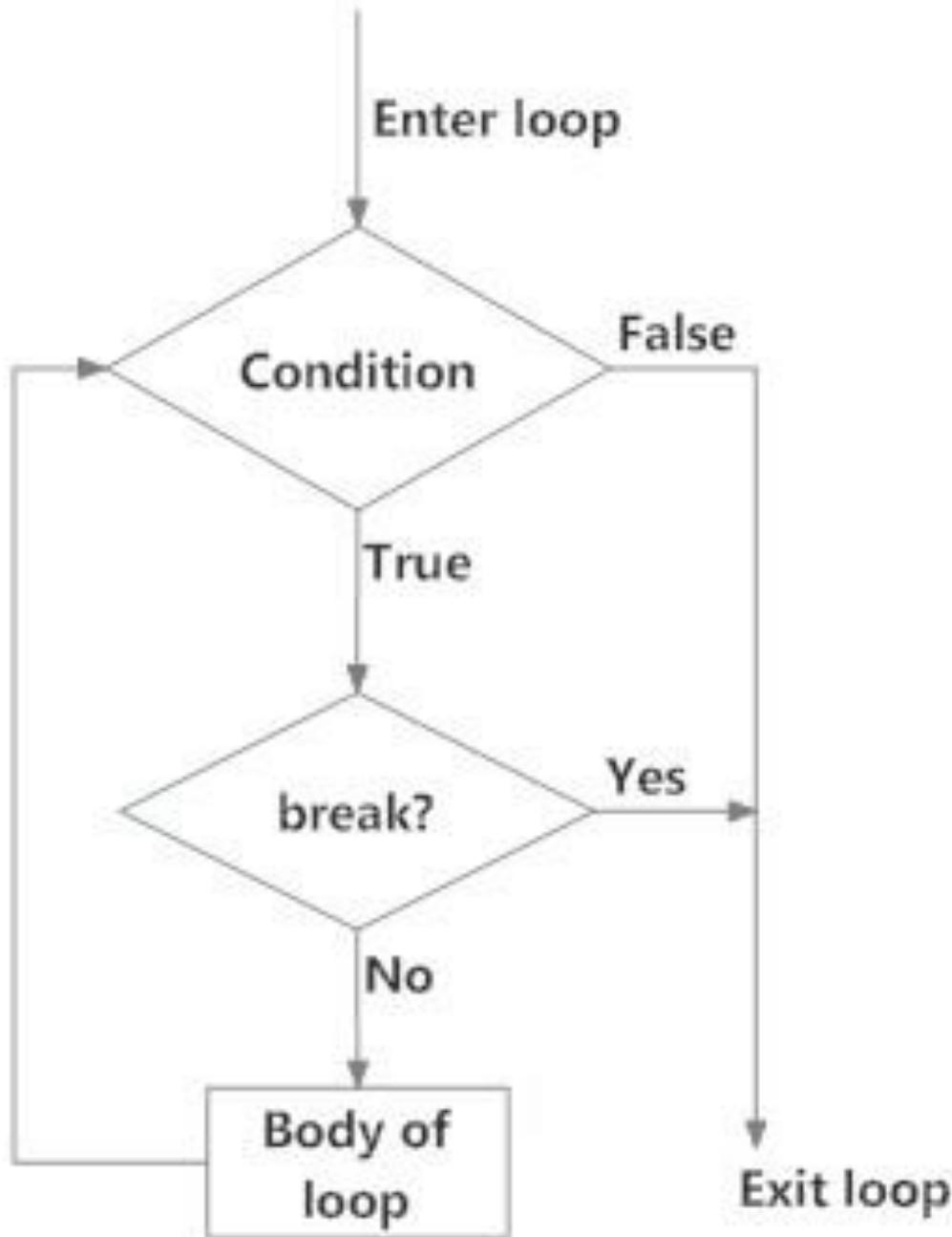
Из цикла можно выйти по условию

Оператор **break** может использоваться внутри цикла (**repeat**, **for**, **while**), чтобы остановить итерации и передать управление операторам, выполняющимся после цикла

В ситуации с вложенными циклами, где есть цикл внутри другого цикла, по оператору **break** выполняется выход из внутреннего цикла во внешний

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

Циклы в R. Алгоритм работы break



Циклы в R. Использование оператора next

Из итерации можно выйти по условию

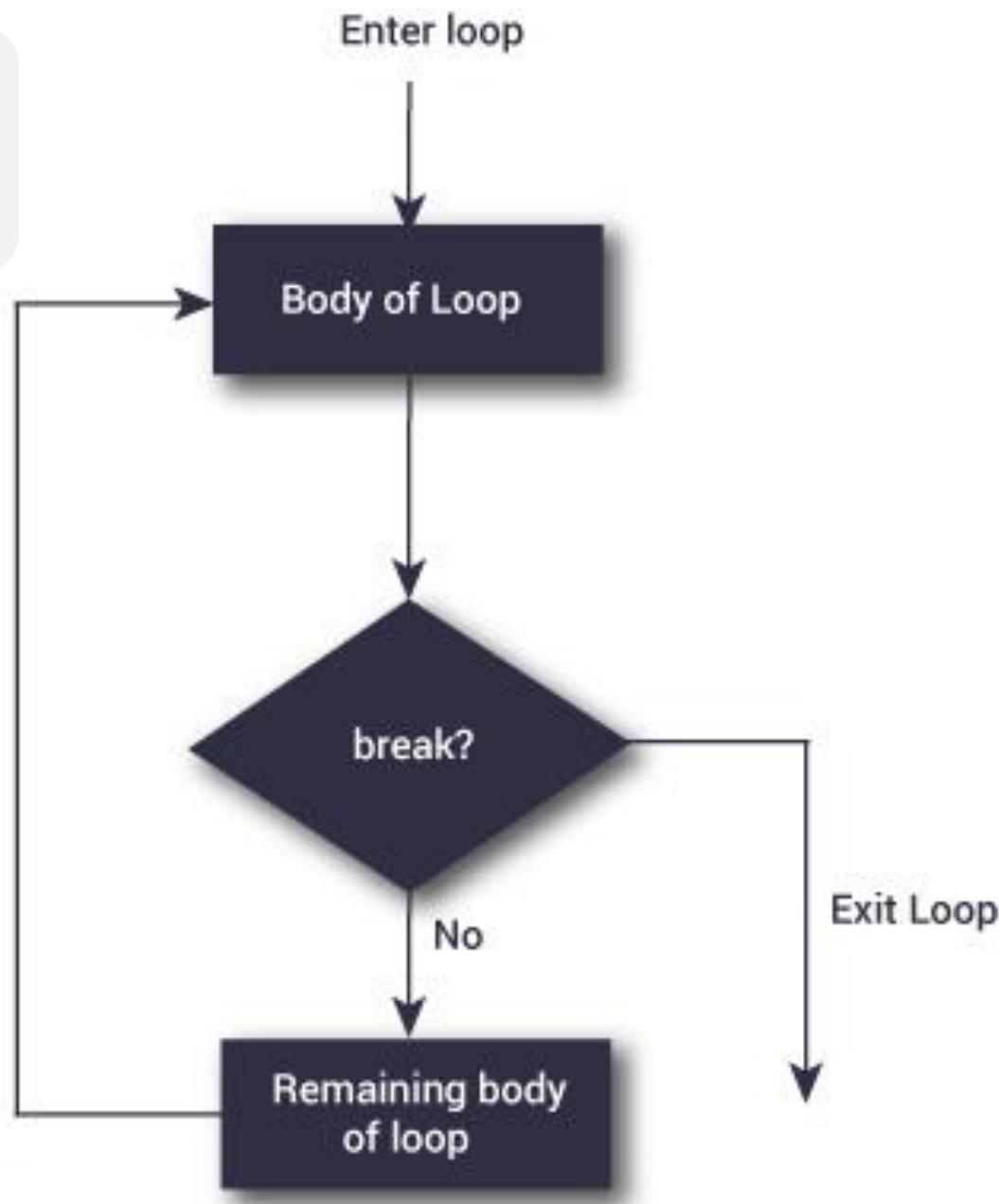
Оператор **next** используется, когда нужно пропустить текущую итерацию цикла, не прерывая сам цикл.

При достижении оператора **next** R пропускает код, идущий после **next** и запускает следующую итерацию цикла.

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

Циклы в R. Цикл repeat

```
repeat {  
  statement  
}
```



Дополнительная гибкость алгоритма – выбор из нескольких значений

- Во многих языках программирования предусмотрены конструкции **выбора из нескольких значений**
- Выбор осуществляется в зависимости от значения целочисленного выражения
- Такие конструкции в языках реализуются с помощью ключевых слов **case, switch**

```
int month = 3;  
string monthString;  
switch (month) {  
    case 1: monthString = "январь";  
    break;  
    case 2: monthString = "февраль";  
    break;  
    case 3: monthString = "Март";  
    break;  
    case 4: monthString = "Апрель";  
    break;  
    case 5: monthString = "Май";  
    break;  
    case 6: monthString = "Июнь";  
    break;  
    case 7: monthString = "Июль";  
    break;  
    case 8: monthString = "Август";  
    break;  
    case 9: monthString = "Сентябрь";  
    break;  
    case 10: monthString = "Октябрь";  
    break;  
    case 11: monthString = "Ноябрь";  
    break;  
    case 12: monthString = "декабрь";  
    break;  
    default: monthString = "Не знаем такого";  
}
```

```
case glc_app_selector  
    of glc_appDix  
        st_prog = 'd'  
    of glc_appMix  
        st_prog = 'm'  
    of glc_appPix  
        st_prog = 'p'  
    of glc_apptix  
        st_prog = 't'  
    of glc_appVix  
        st_prog = 'v'  
    of glc_appRix  
        st_prog = 'r'  
end
```

В языке R встроенной конструкции выбора из нескольких значений нет

Функция switch() – аналог выбора из нескольких значений

w <- switch(Q, Выбор 1, Выбор 2, , Выбор N),
где Q – вектор единичной длины, содержит оцениваемое выражение,
Q – целое число или строка

Если Q – целое число

Если Q == 1, то w<- Выбор 1

Q == 2, то w<- Выбор 2

.....

Q == N, то w<- Выбор N

Если Q вне диапазона 1..N, чему равно w ?

e <- sample(x=6, size=1)

w<- switch(e, '55qw', 2:13, -7.7, F, 3)

w

если Q = не целое, чему равно w?

Варианты Q :

- Целое
- Действительное
- Логическое
- Строковое

Q приводится:

- К целому
- К целому

Как анализируется строка?

Функция switch() – аналог выбора из нескольких значений

Если Q – строка

В общем случае каждому варианту выбора присваивается имя (метка)

w <- switch(Q, имя1 = Выбор 1, имя2 = Выбор 2, , имяN = Выбор N)

При точном совпадении Q и имени выбора функция switch() вернет значение, соответствующее совпавшему имени выбора, иначе вернет NULL

```
e<- '6qTR'
```

```
w<- switch(e, '6qTR' = '55-65-12.4', '6qTR' = 2:13, 't' = -7.7, 'T'=T)
```

У **одного** варианта выбора **имя можно не указывать**. В этом случае при несовпадении Q и имен выборов функция switch() вернет значение выбора без имени.

Значение может быть любого типа

```
e<- 'T'
```

```
w<- switch(e, '6qTR' = '55-65-12.4', '6qTR' = 2:13, '45-fgx0')
```

Лукьянов Павел Борисович
профессор департамента Анализа данных, принятия решений
и финансовых технологий

Программирование в среде R

Лекция 3

Разработка сложных программ.
Возможности языка R

Финансовый университет, 2020

Большие программы и проблемы разработки

Реальные задачи Заказчика



Сложные задачи :

- Многообразие взаимосвязанных параметров
- Сложность обработки данных
- Алгоритмическая сложность
- Вычислительная сложность
- Сложность поиска и оптимизации решений
- Многообразие видов анализа и т.д.



Для решения сложных задач Разработчик пишет
большие программы (усилия, проблемы, время)



Современные
средства разработки



Использование
возможностей языка



Готовые программные
решения

Стандартный набор: консоль, ядро R

R version 3.4.1 (2017-06-30) -- "Mojave"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R -- это проект, в котором сотрудничает множество разработчиков.
Ведите 'contributors()' для получения дополнительной информации и
'citation()' для ознакомления с правилами упоминания R и его пакетов
в публикациях.

Ведите 'demo()' для запуска демонстрационных программ, 'help()' -- для
получения справки, 'help.start()' -- для доступа к справке через браузер.
Ведите 'q()', чтобы выйти из R.

[Загружено ранее сохраненное рабочее пространство]

> |

The screenshot shows the R console interface with two windows open. The main window displays the R startup message and help information. A yellow callout box highlights the text 'Минимализм, работа на всех устройствах' (Minimalism, work on all devices). A blue arrow points from this callout to a red button labeled 'Требуется' (Required). Another blue arrow points from this button to a list of requirements. A large yellow callout box contains a bulleted list of requirements.

Требуется

- Повышение удобства работы
- Развитие функционала
- Рост эффективности разработок

- Пакеты и библиотеки для решения прикладных задач в различных областях
- Средства командной работы
- Распределенные сложные вычисления
- Интеллектуальные среды разработки, свои среды под конкретные задачи
- Клиент-серверные технологии (web-интерфейс, работа в браузере без установки R)
- Поддержка Пользователей и Разработчиков

Варианты запуска R (папка c:\...R\R-номерВерсии\bin\x64)

Rlapack	dll
open	exe
R	exe
Rcmd	exe
Rfe	exe
Rgui	exe
Rscript	exe
RSetReg	exe
Rterm	exe

R.exe запускает Rterm.exe (терминал, консоль)
Результат запуска R и Rterm одинаков

Rcmd.exe – запускает R с поддержкой
системных команд операционной системы

Rscript.exe – запускает R для выполнения
скриптов, указываемых как параметры

Работа с консолью

RSgui.exe – простейшая реализация графического
интерфейса R для работы с консолью и скриптами

- Интерактивность
- Анализ каждой команды

Работа с GUI (Graphic User Interface)

- Работа со скриптами
- Пакетная обработка команд

Современные средства разработки

Состав типового дистрибутива R



Развитие функционала

Рост удобств, эффективности и скорости работы, развитие сервисов

Развитие GUI

Клиент-серверная технология

Коллективная работа

GUI – Graphics User Interface

Graphical User Interface для R

R Commander или Rcmdr

Платформонезависимый графический интерфейс

<http://socserv.mcmaster.ca/jfox/Misc/Rcmdr/>

Установка > install.packages("Rcmdr", dependencies=TRUE)

Запуск > library(Rcmdr)

RKward

удобный графический интерфейс к R

<https://rkward.kde.org/>

SciViews

Простой графический интерфейс

<http://www.sciviews.org/>

Repl.it

On-line среда: создание и
запуск приложений в
браузере <http://repl.it/>

RPMG (Really Poor Man's GUI)

Простейший GUI – создание
интерактивных графических окон
install.packages("RPMG", dep=TRUE)
library(RPMG); demo(RPMG)

RStudio

Современный GUI с реализацией клиент-серверной технологии
<https://www.rstudio.com/>

Ядро R

Консоль

JGR (Java GUI for R)

Графический интерфейс на Java

<https://www.rforge.net/JGR/>

RATTLE (R Analytical Tool To Learn Easily)

Интеллектуальный анализ данных (data
mining) <https://rattle.togaware.com/>
install.packages("rattle", dependencies=TRUE)
library(rattle); rattle()

Emacs

Семейство редакторов с расширенными
возможностями настроек

<https://vigou3.gitlab.io/emacs-modified-windows/>

[/https://rstudio.com/products/rstudio/download](https://rstudio.com/products/rstudio/download)

<https://resources.rstudio.com/>

- **Решение проблем с кодировкой**
(File – Reopen with Encoding – выбор кодировки)
- **Автоматический перенос фокуса в консоль**
(Tools – Global Options – Code – Focus console after executing from source)
- **Автоматическое форматирование кода**
(Code – Reformat Code или **Ctrl–Shift–A**)
- **Задать / отключить комментарии для выделенного текста**
(Code – Comment/Uncomment Lines или **Ctrl–Shift–C**)
- **В компьютерных классах работает подсказка**
например, ?q

Большие программы и проблемы разработки

Реальные задачи Заказчика



Сложные задачи :

- Многообразие взаимосвязанных параметров
- Сложность обработки данных
- Алгоритмическая сложность
- Вычислительная сложность
- Сложность поиска и оптимизации решений
- Многообразие видов анализа и т.д.



Для решения сложных задач Разработчик пишет
Большие программы (усилия, проблемы, время)



Современные
средства разработки

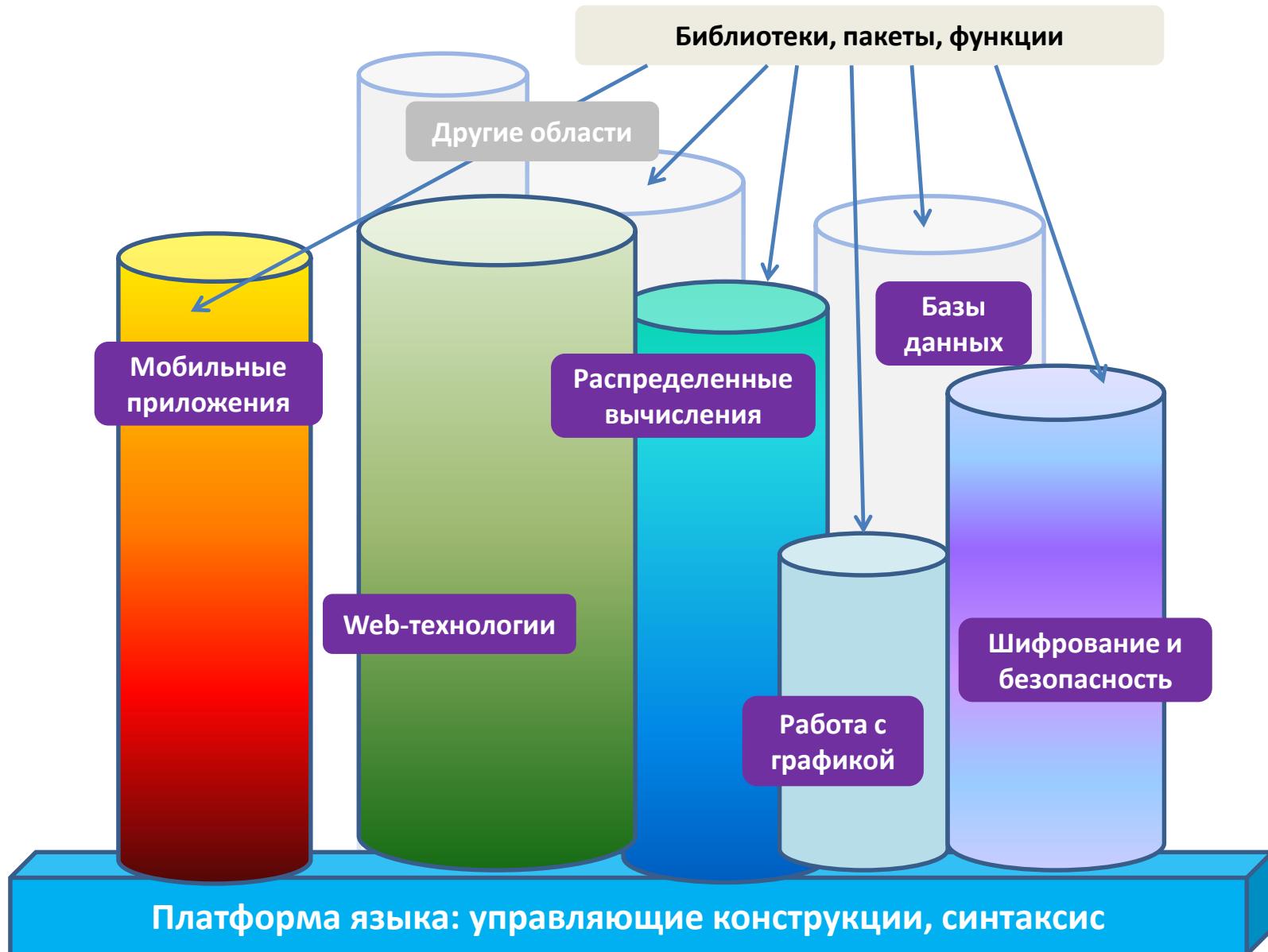


Использование
возможностей языка



Готовые программные
решения

Портрет современного языка программирования



Что общего и в чем различие в современных языках программирования?

Пример: язык Z0



Пример: язык Z1

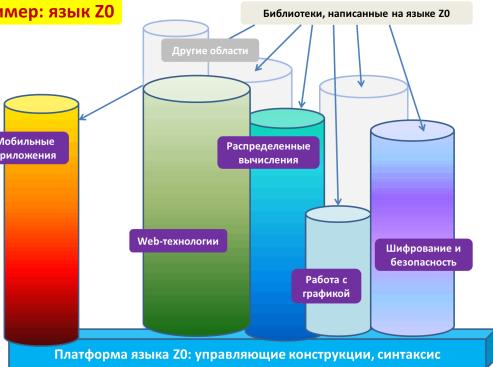


Пример: язык Z++



Что общего и в чем различие в современных языках программирования?

Пример: язык Z0



Основы языков очень похожи:

- Управляющие конструкции
- Синтаксис

В чем проявляются различия:

- Библиотеки, написанные на языке
- Технологии, реализованные в библиотеках

Пример: язык Z++



Условно старые языки (C, C#, Java, Fortran, ...):

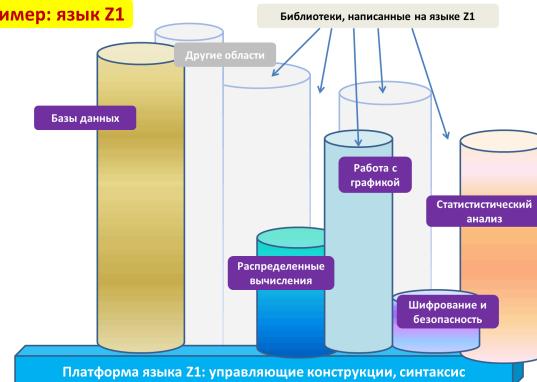
- Сложность
- Наличие старых технологий из-за проблем наследования
- Огромные библиотеки на все случаи жизни

R

Условно новые языки (Python, JavaScript, F#, ...):

- Более гибкая платформа
- Легкость изучения
- Низкий порог входа
- Относительно малое количество библиотек

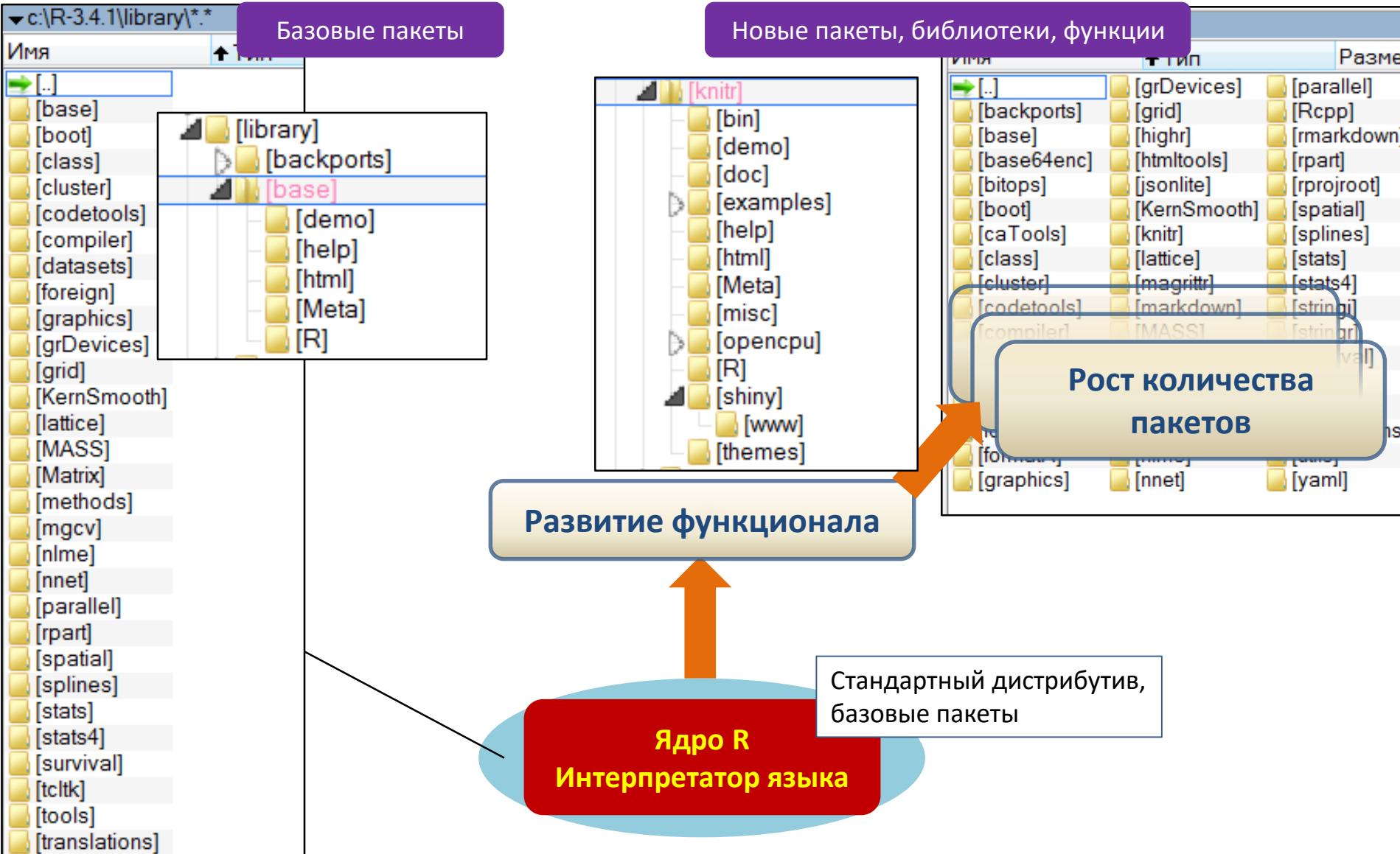
Пример: язык Z1



Расширение возможностей R: все функции размещаются в пакетах

Пакет

- Пакет R - набор связанных между собой данных, функций и документации
- Файлы пакета размещаются в нескольких вложенных папках внутри папки с названием пакета
- Все пакеты размещаются в папке **library** установленной среды R



Как вызвать функцию из базового пакета?

Базовые пакеты,
стандартный дистрибутив

[..]
[base]
[boot]
[class]
[cluster]
[codetools]
[compiler]
[datasets]
[foreign]
[graphics]
[grDevices]
[grid]
[KernSmooth]
[lattice]
[MASS]
[Matrix]
[methods]
[mgcv]
[nlme]
[nnet]
[parallel]
[rpart]
[spatial]
[splines]
[stats]
[stats4]
[survival]
[tcltk]
[tools]
[translations]
[utils]

В скрипте Программист выполняет
прямой вызов функции из базового пакета

- Все пакеты из стандартного дистрибутива автоматически загружаются в оперативную память при запуске R
- Вызов функции происходит через указание ее имени и передачу управляющих параметров

```
i = readline("Введите число ")
i = as.integer(i)

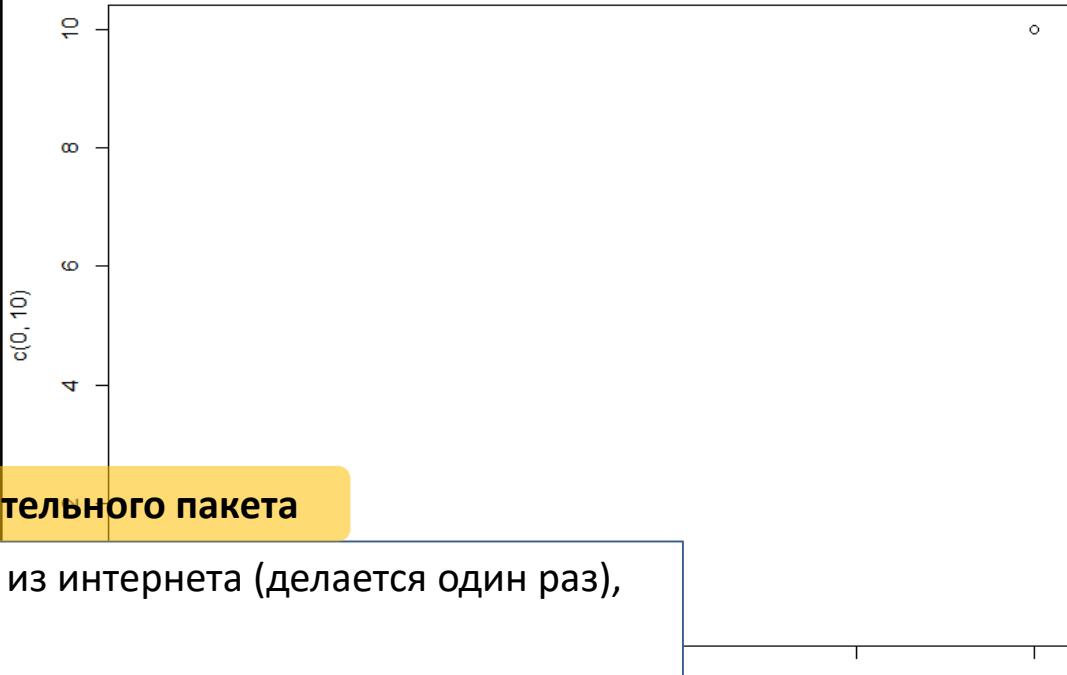
# или в одну строчку:

i = as.integer(readline("Введите число "))
```

Вызов функции из дополнительно установленного пакета

Дополнительно установленные пакеты – в дистрибутив не входят!

[backports]	[grid]	[Rcpp]
[base]	[highr]	[rmarkdown]
[base64enc]	[htmltools]	[rpart]
[bitops]	[jsonlite]	[rprojroot]
[boot]	[KernSmooth]	[spatial]
[caTools]	[knitr]	[splines]
[class]	[lattice]	[stats]
[cluster]	[magrittr]	[stats4]
[codetools]	[markdown]	[stringi]
[compiler]	[MASS]	[stringr]
[datasets]	[Matrix]	[survival]
[digest]	[methods]	[tcltk]
[evaluate]	[mgcv]	[tools]
[foreign]	[mime]	[translations]
[formatR]	[nlme]	[utils]
[graphics]	[nnet]	[yaml]



Порядок вызова функции из дополнительного пакета

- Сначала выполнить установку пакета из интернета (делается один раз), функция `install.packages()`
- После запуска R пакет загрузить в оперативную память, функция `library()`
- Функцию из пакета вызывать обычным образом

```
install.packages("RPMG", dep=TRUE)  
library(RPMG)  
plot(c(0,100), c(0,10))  
rowBUTTONS(c("Сохранить", "Отмена"))
```

Большие программы и проблемы разработки

Реальные задачи Заказчика



Сложные задачи :

- Многообразие взаимосвязанных параметров
- Сложность обработки данных
- Алгоритмическая сложность
- Вычислительная сложность
- Сложность поиска и оптимизации решений
- Многообразие видов анализа и т.д.



Для решения сложных задач Разработчик пишет
Большие программы (усилия, проблемы, время)

Современные
средства разработки

Использование
возможностей языка

Готовые программные
решения

Расширение возможностей языка: функции, созданные Пользователем

Когда нужна своя функция?

- Если часть кода в программе выполняется **более одного раза**
- Если код решает некоторую **частную задачу**
- Если в дальнейшем предполагается использовать код **повторно**

Как написать
свою функцию?

- Как правило, функция объявляется в **отдельном скрипте**
- Программист задает функцию, пишет ее код:
 - придумывает **имя**, перечисляет **параметры** функции
 - задает **правила обработки параметров** функции
 - определяет **результат выполнения** функции
- Сначала идет **объявление** функции, затем ее **использование** (вызов функции)
- Программист может делать вызовы одной функции в разных программах

Функции, написанные Пользователем
(Программистом)

Вычислительные
пакеты и библиотеки

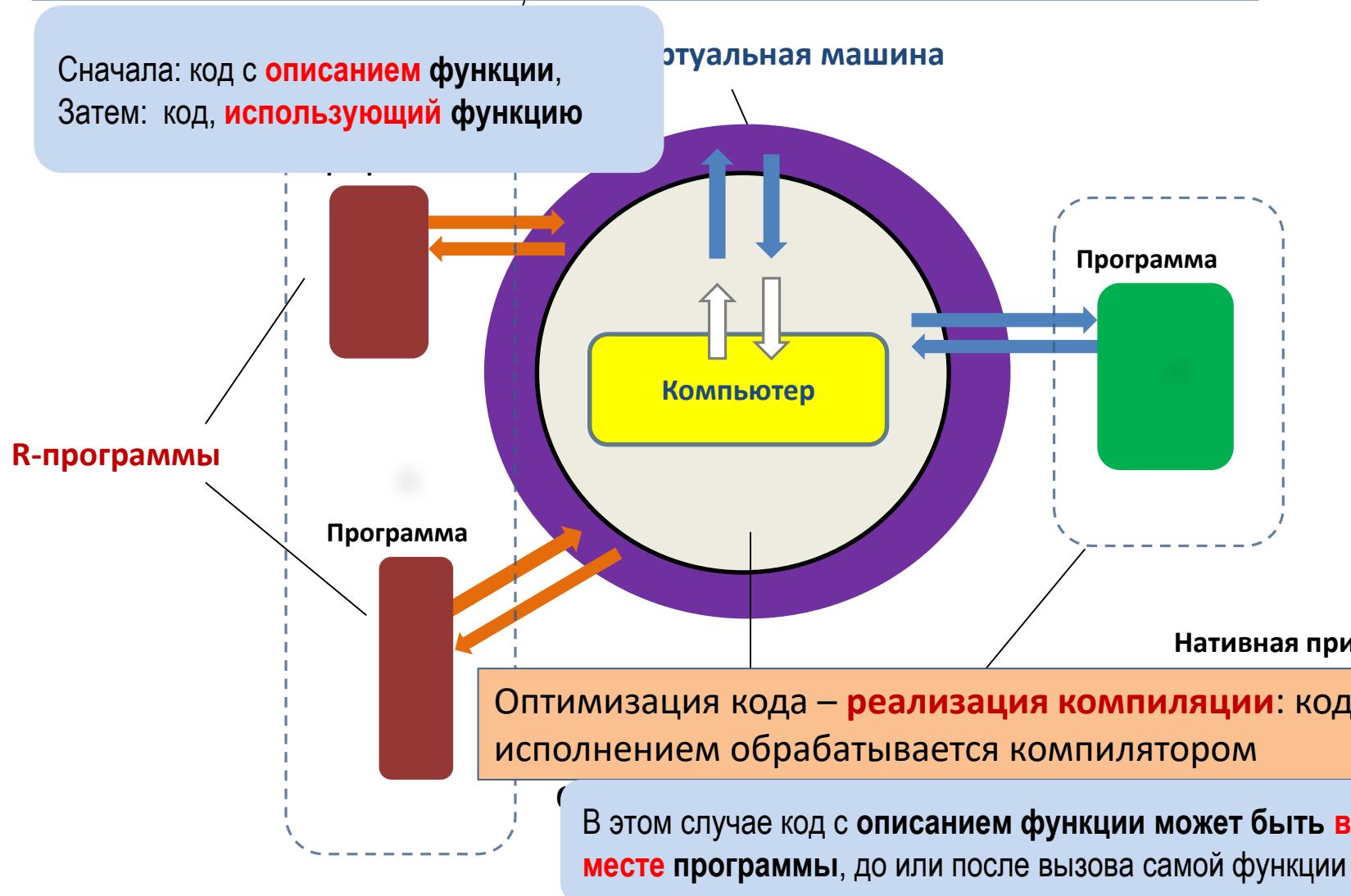
Развитие функционала

Ядро R
Интерпретатор языка

Стандартный дистрибутив,
базовые пакеты

Порядок выполнения программ с использованием функций

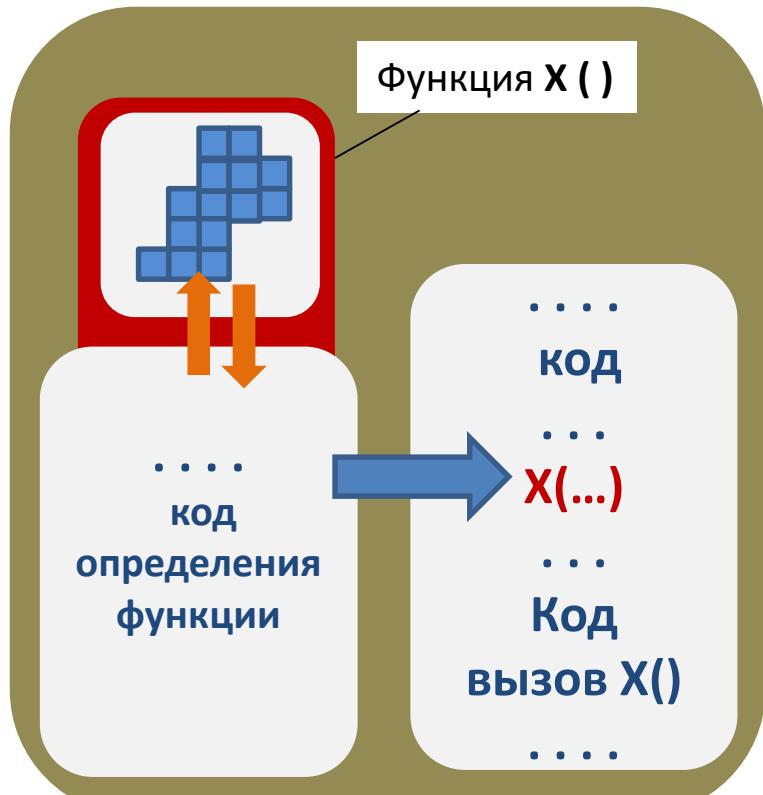
все команды программы на языке R выполняются **последовательно**,
строчка за строчкой, в том порядке, как написал строки программист



Использование функции Пользователя в программе

Определение и вызов функции в одном скрипте

Используется для простых задач



Скрипт один. Функция сначала описывается, затем вызывается

Важно

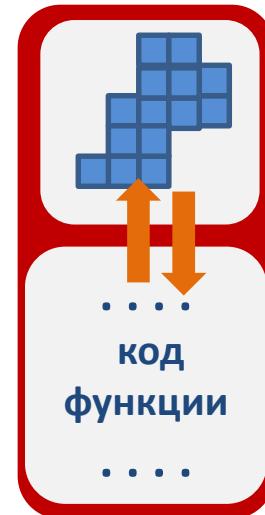
Описание X() должно идти раньше вызова X()

Почему?

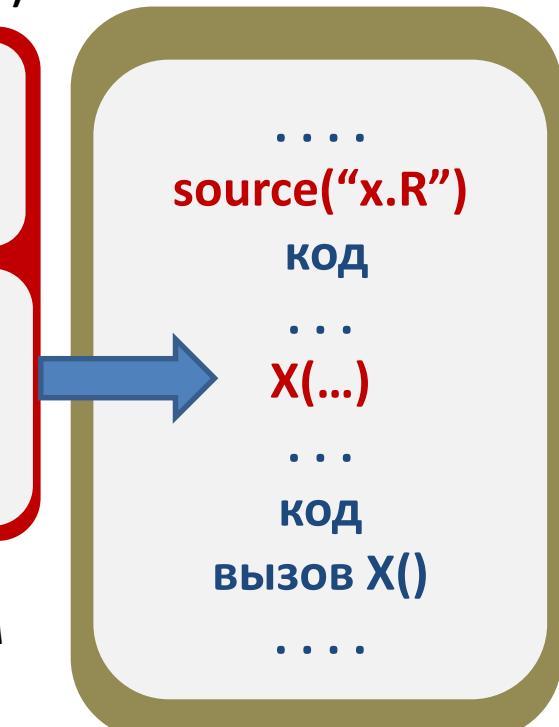
Определение функции – в одном скрипте, вызов – из других скриптов

Используется для сложных задач, когда скрипт, вызывающий функцию, не один, а несколько

Функция X ()



Скрипт x.R с определением функции



Скрипт с вызовом функции

Важно

Сначала вызывается source("x.R"), затем идет обращение к функции X()

Почему?

Зачем в программировании нужны свои функции?

Функции используются для логического разбиения кода на более простые части, что позволяет легко их понимать и развивать

Правила в программировании

- код, который используется более одного раза, надо помещать в отдельную функцию
- код, который не виден полностью на одном экране, надо разбивать на функции

```
funcName <- function(argument) {  
    statement  
}
```

```
pow <- function(x, y){  
    result <- x^y  
    print(paste(x, "в степени", y, 'равно', result))  
}
```

Именованные аргументы функции

При вызове функции фактические аргументы (2, 8) сопоставляются с формальными аргументами (x, y) в порядке следования аргументов

x = 2; y = 8

Можно вызвать функцию, используя явное задание значений формальным аргументам:

pow(8, 2)

pow(x = 8, y = 2)

pow(y = 2, x = 8) # изменили порядок следования аргументов

pow(x = 8, 2)

pow(2, x = 8) # изменили порядок следования аргументов

При вызове функции таким образом порядок фактических аргументов не имеет значения.

ПРАВИЛО. Сначала сопоставляются все именованные аргументы, а затем оставшиеся неименованные аргументы сопоставляются в порядке их следования

Значения по умолчанию для аргументов функции

Часто аргументам функции присваивают значения по умолчанию

Это делается путем инициализации формального аргумента при объявлении функции.

Инициализация – задание начального значения чему-либо

приведенная выше функция со значением по умолчанию для y:

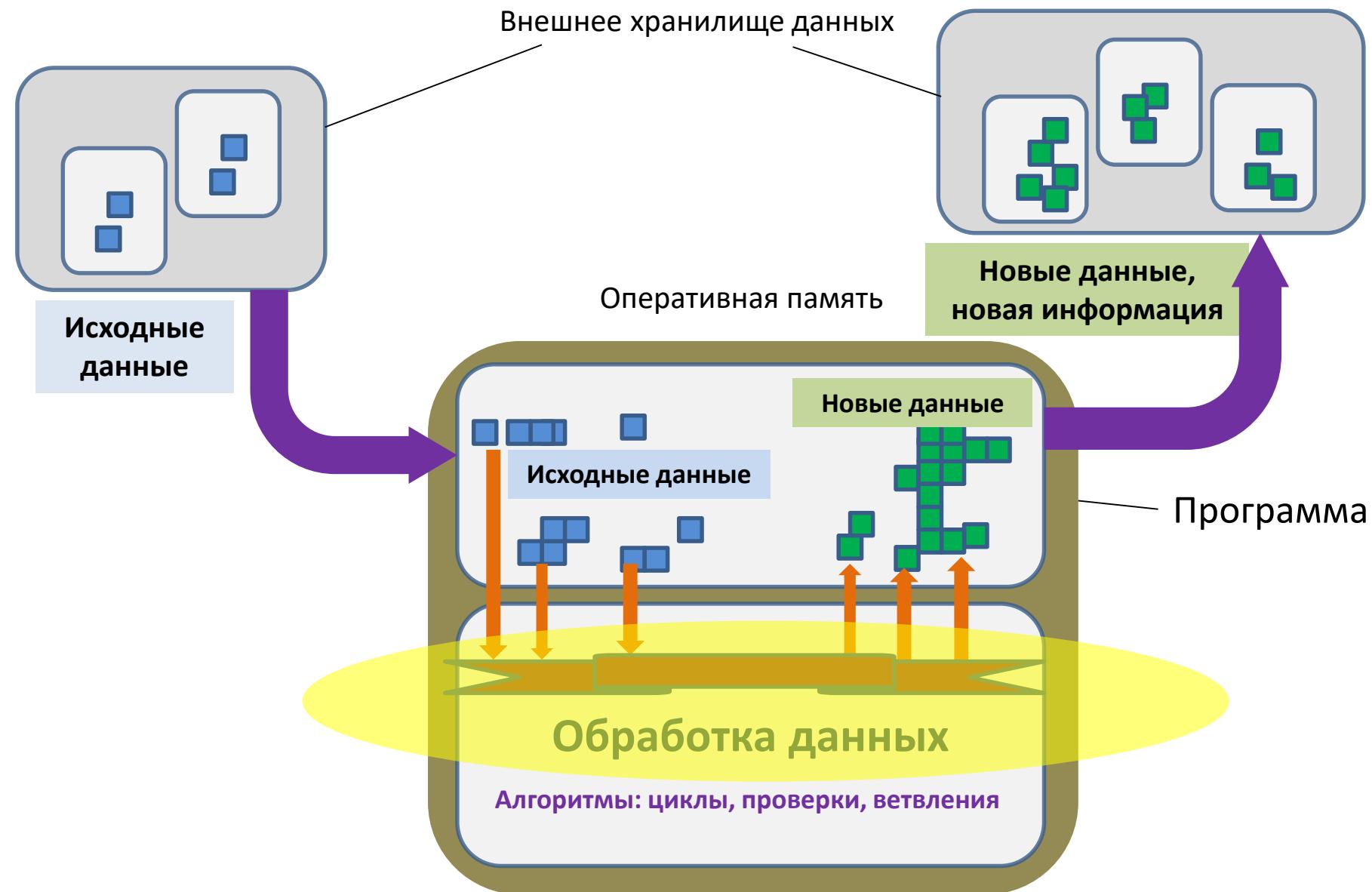
```
pow <- function(x, y = 2) {  
  result <- x^y  
  print(paste(x, "в степени", y, "равно", result))  
}
```

```
pow(3) # 9
```

```
pow(3, 1) # 3
```

```
pow(3, 0.4) # 1.551846
```

Обработка данных как цель работы программы



Исходные данные -> Обработка данных -> Итоговые данные -> Новая информация -> Новое знание

Лукьянов Павел Борисович
профессор департамента Анализа данных, принятия решений
и финансовых технологий

Программирование в среде R

Лекция 4

Графика в R

Финансовый университет, 2020

Графические функции и их возможности

Графические функции на языке R

base graphics: Базовые графические функции

Определены в пакете `graphics`.
Пакет включен в стандартный дистрибутив R

Альтернативные функции из внешних пакетов

Сложный в настройке визуальный анализ,
решение специальных задач

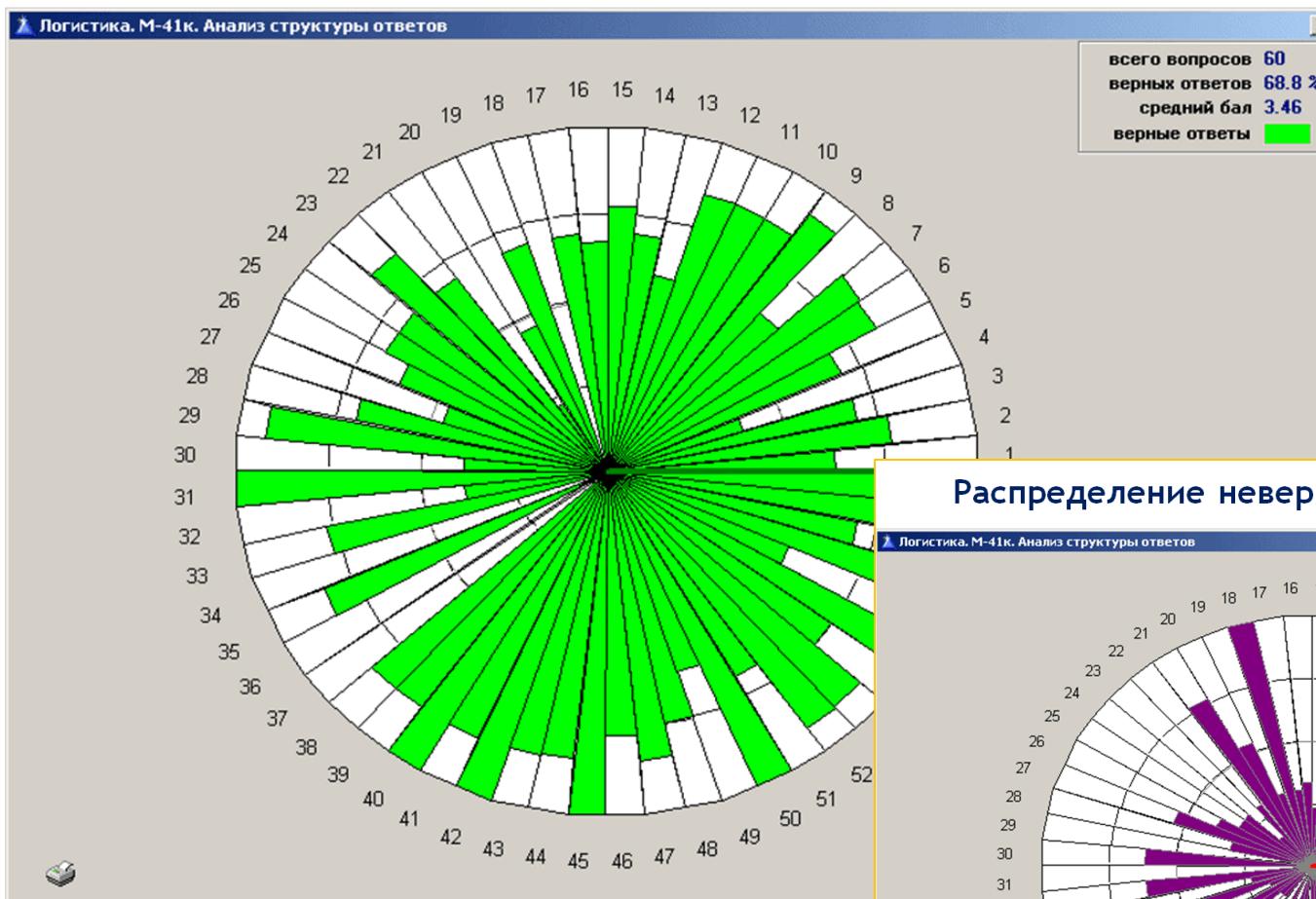
Графические функции Пользователя

Создаются Программистом для визуального
представления данных в своих программах

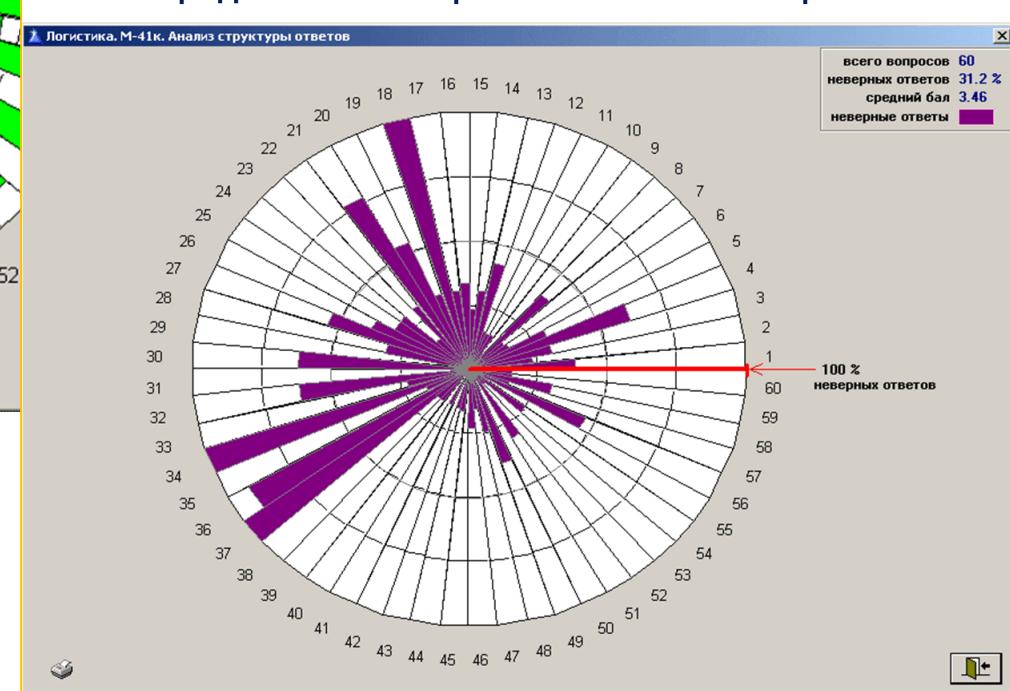
В нашем курсе мы будем изучать базовые графические функции пакета `graphics`.
Но сначала увидим примеры функций Пользователя и функций из внешних пакетов

Примеры графических функций, написанных Пользователем, 1

Распределение правильных ответов по вопросам

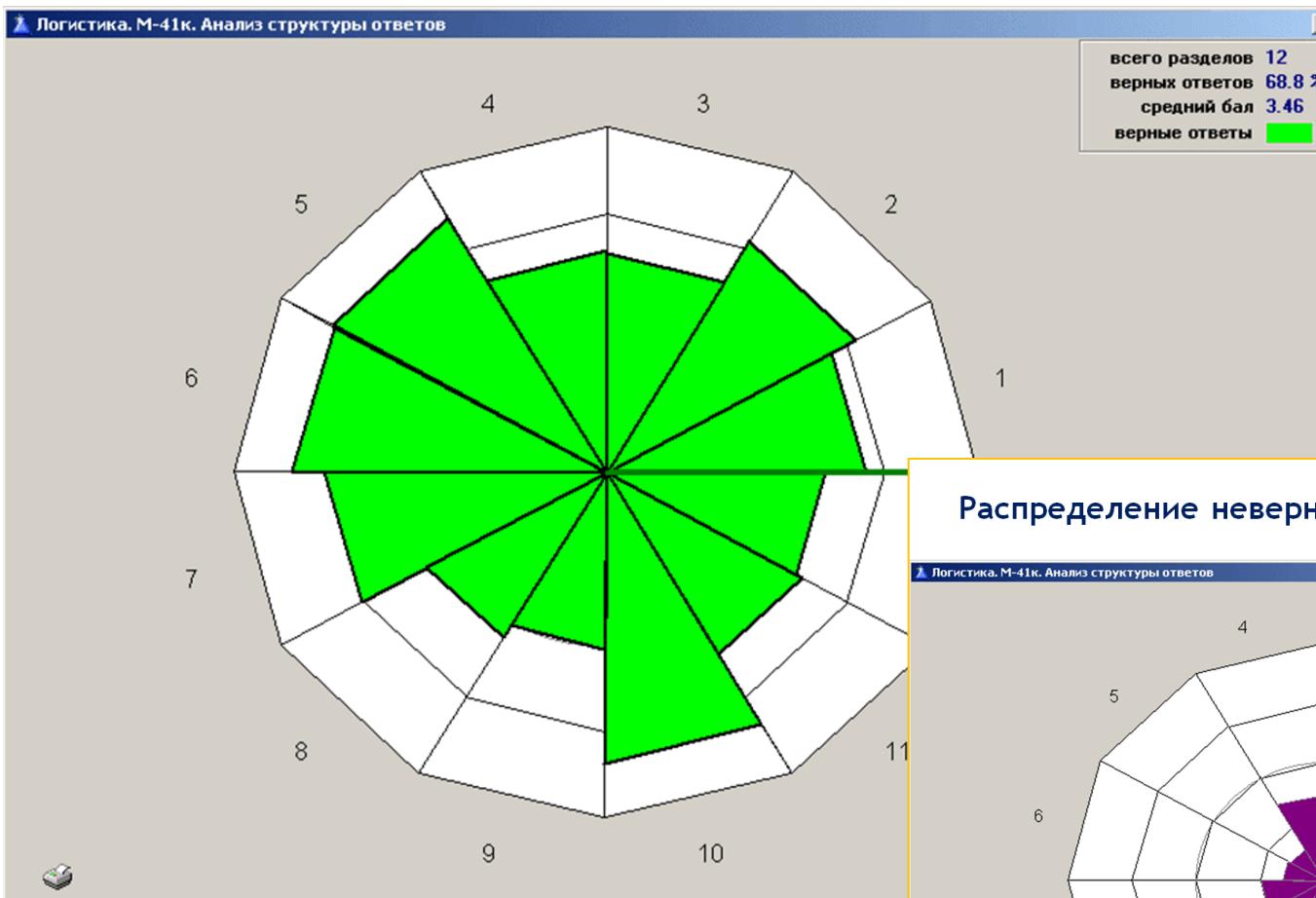


Распределение неверных ответов по вопросам

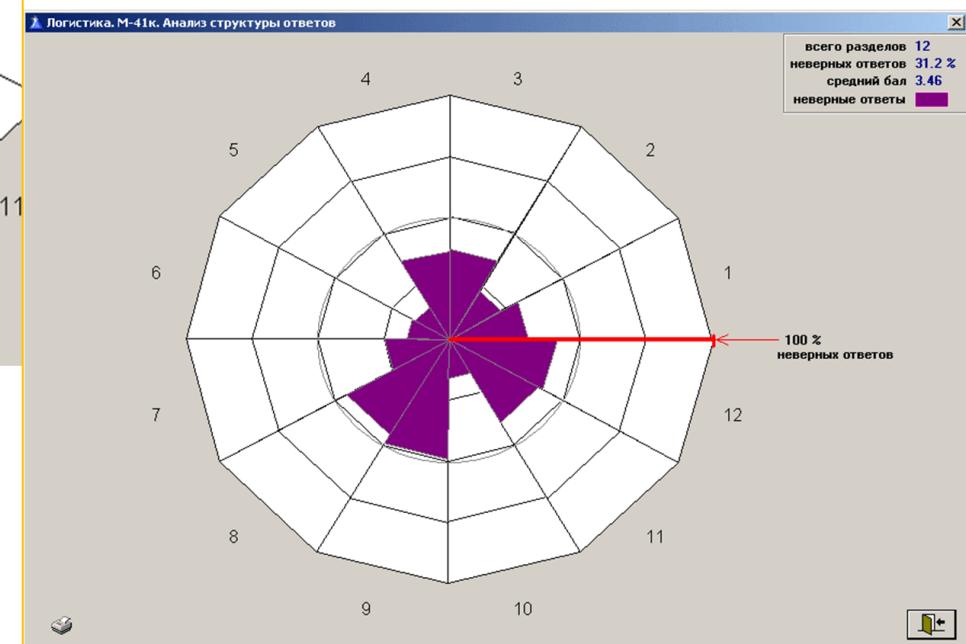


Примеры графических функций, написанных Пользователем, 2

Распределение правильных ответов по разделам



Распределение неверных ответов по разделам



Графические функции для языка R



Функции из внешних пакетов

Графические библиотеки из внешних пакетов

- **plotly** – интерактивная графика, написана на JS
<https://plotly.com/r/getting-started/>
- **ggplot2** – универсальный пакет альтернативных графических функций
- **lattice** – трехмерная графика, анализ многомерных данных
- **ggvis** – интерактивная графика

визуальный анализ,
аналитических задач

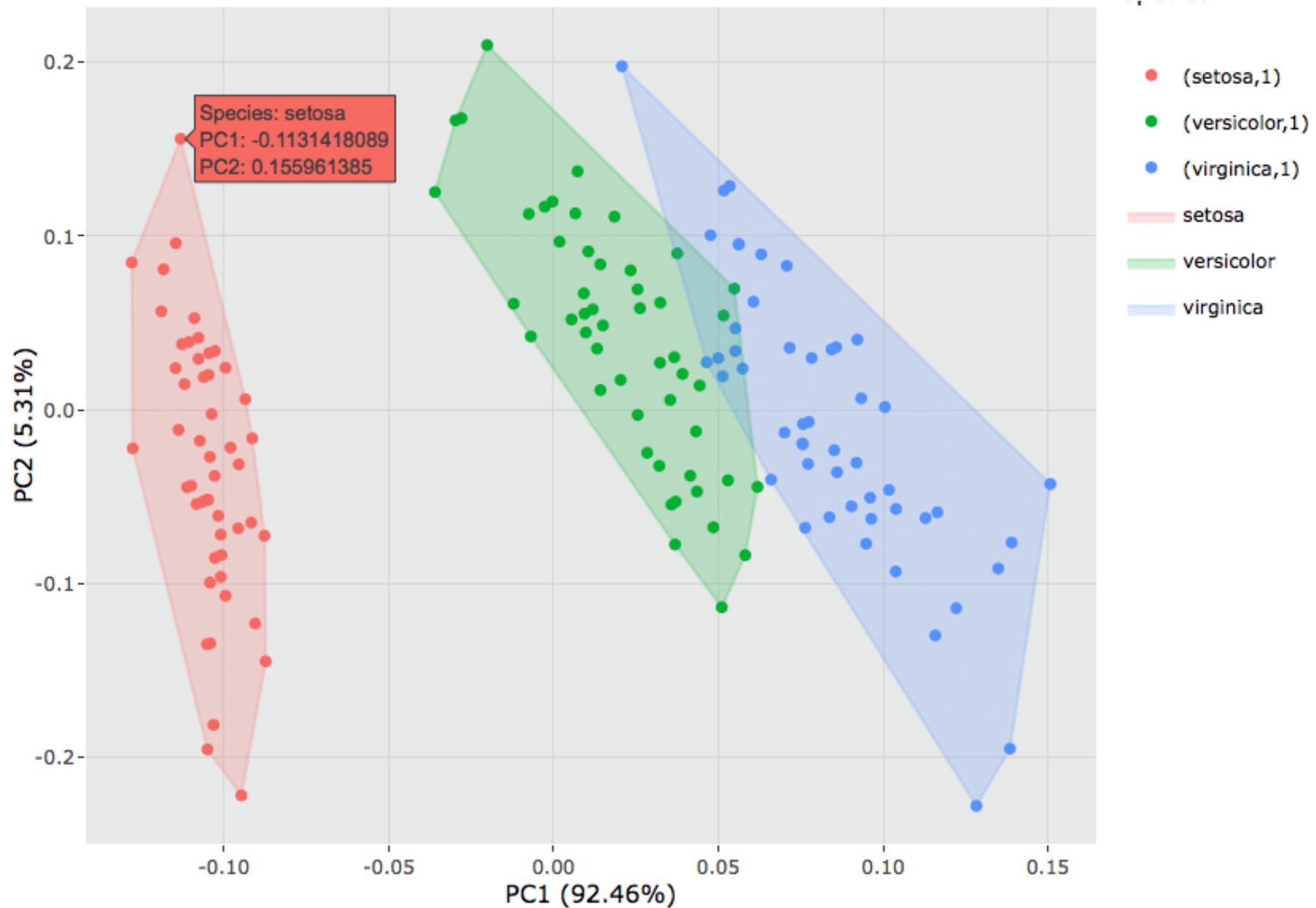
Еще раз про использование функций из внешних пакетов

Установка:

```
# выполняется один раз  
install.packages("ggplot2")
```

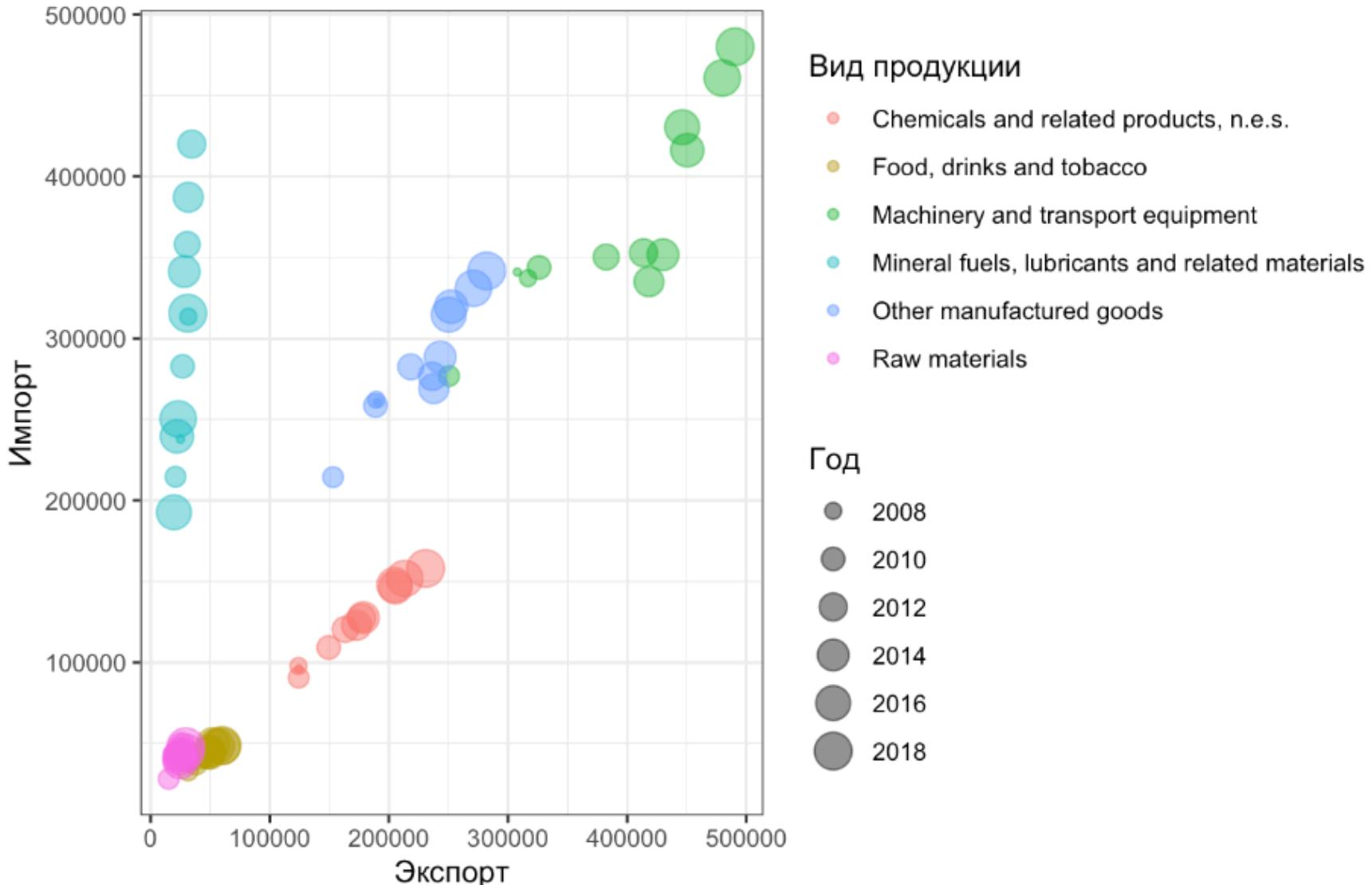
```
# выполняется каждый раз перед вызовом функций пакета  
library(ggplot)
```

Примеры использования графических функций пакета plotly

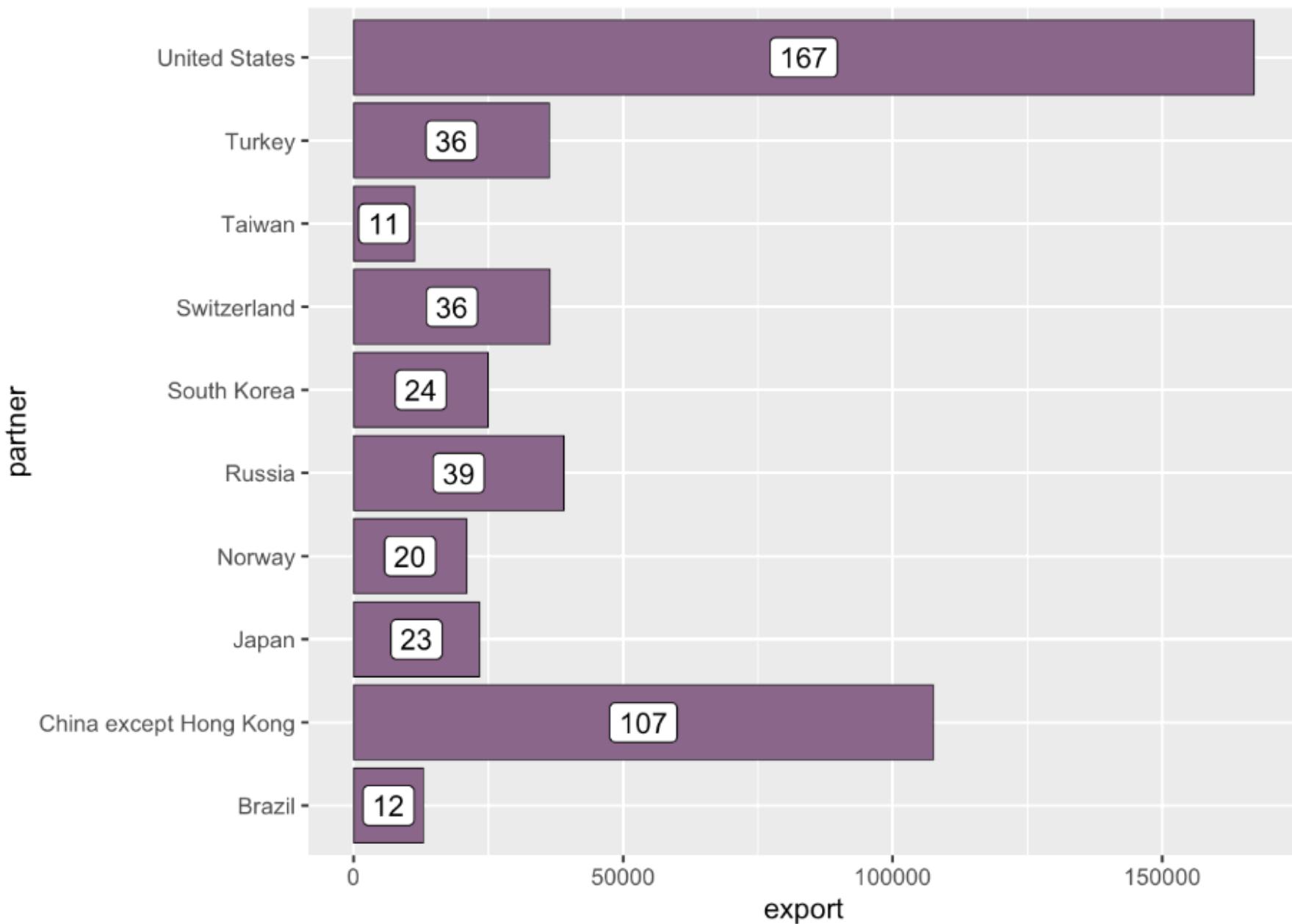


Примеры использования графических функций пакета ggplot2

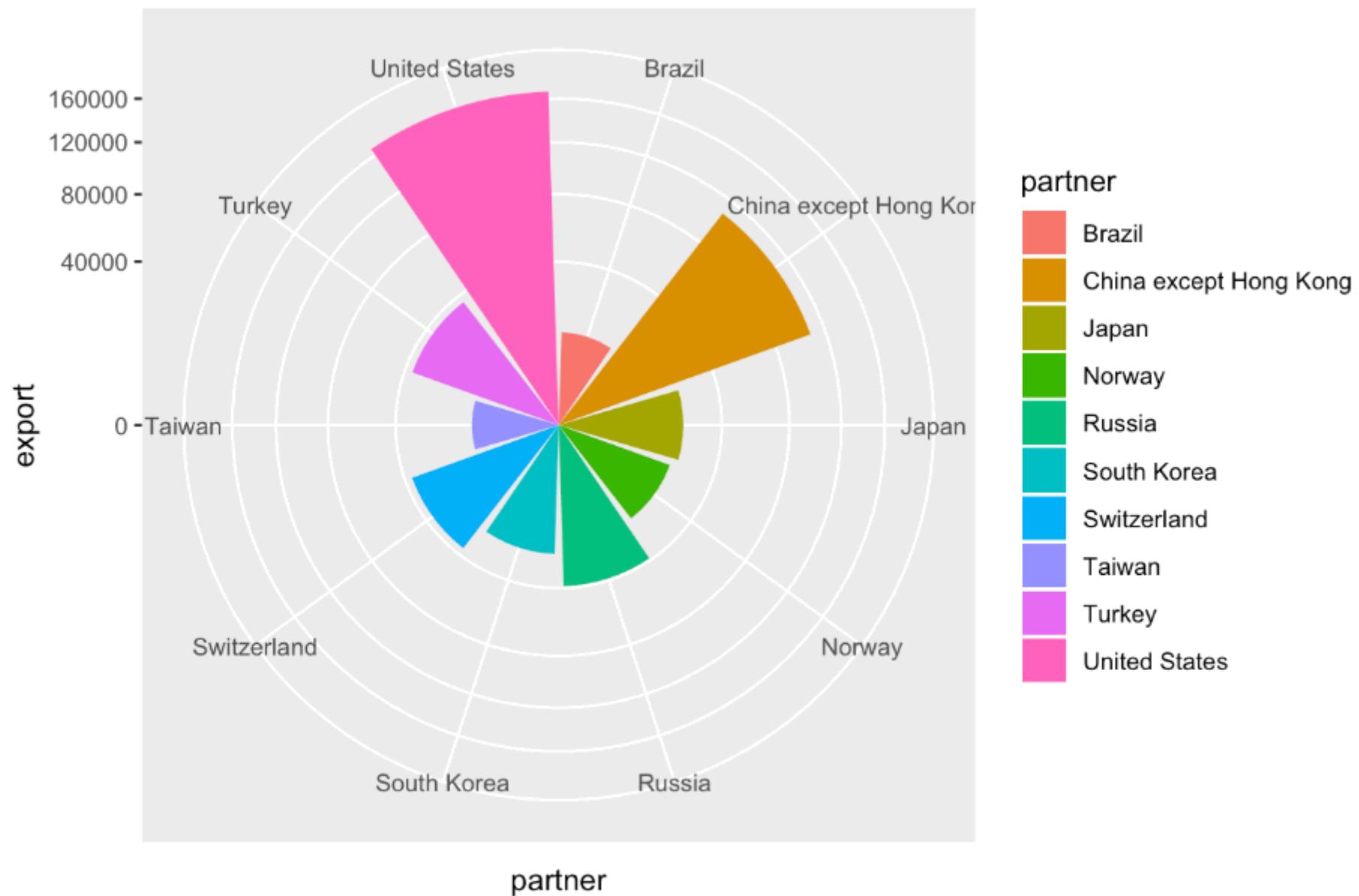
Соотношение импорта и экспорта в странах Евросоюза (млн долл. США) Данные по ключевым партнерам



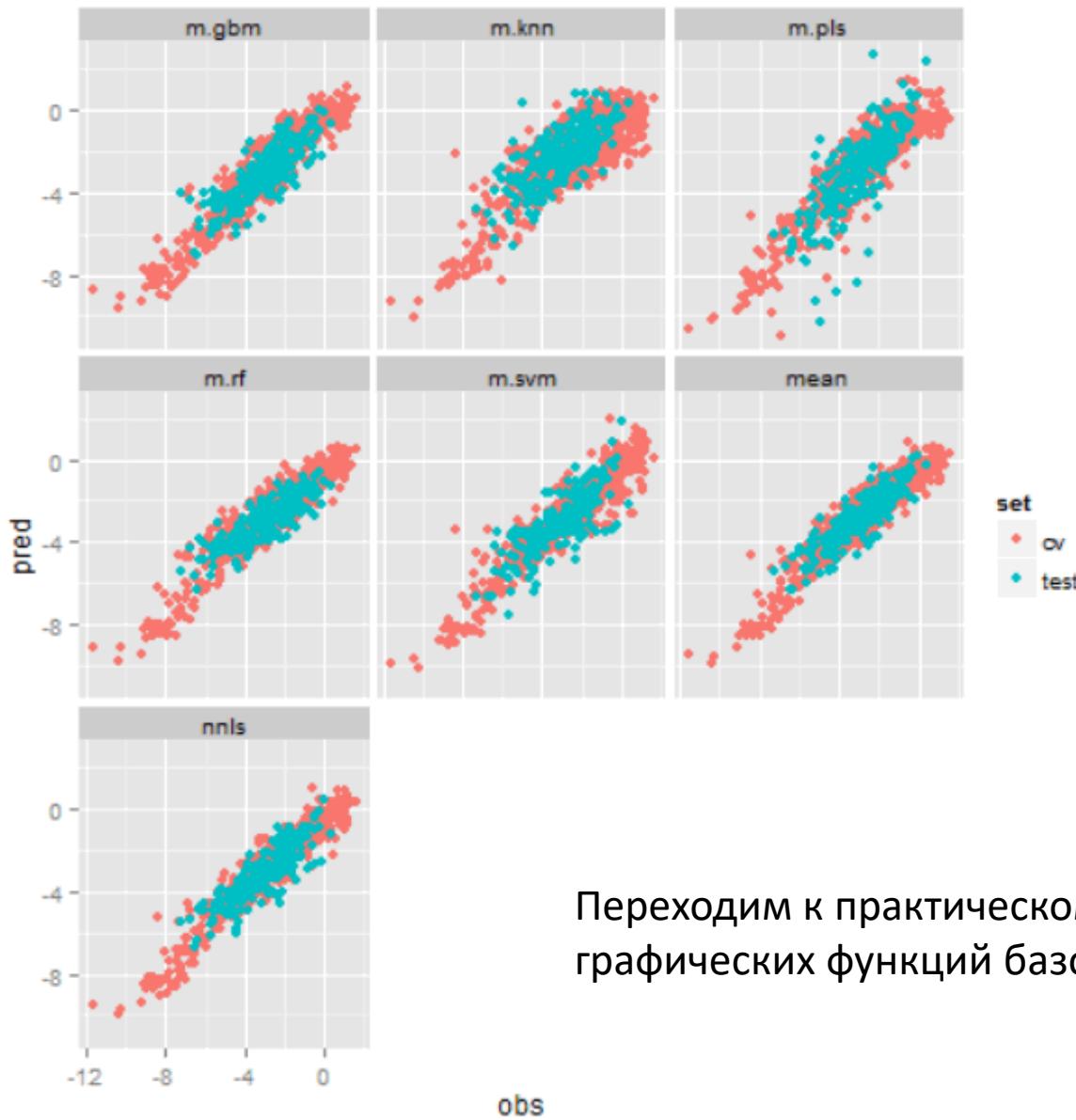
Примеры использования графических функций пакета ggplot2



Примеры использования графических функций пакета ggplot2



Примеры использования графических функций пакета ggplot2



Переходим к практическому изучению
графических функций базового пакета `graphics`

Графические функции R, пакет `graphics`

Включены в стандартный дистрибутив R

Основные

Вспомогательные

Не работают без основных

Связь основных и вспомогательных графических функций

Основные графические функции

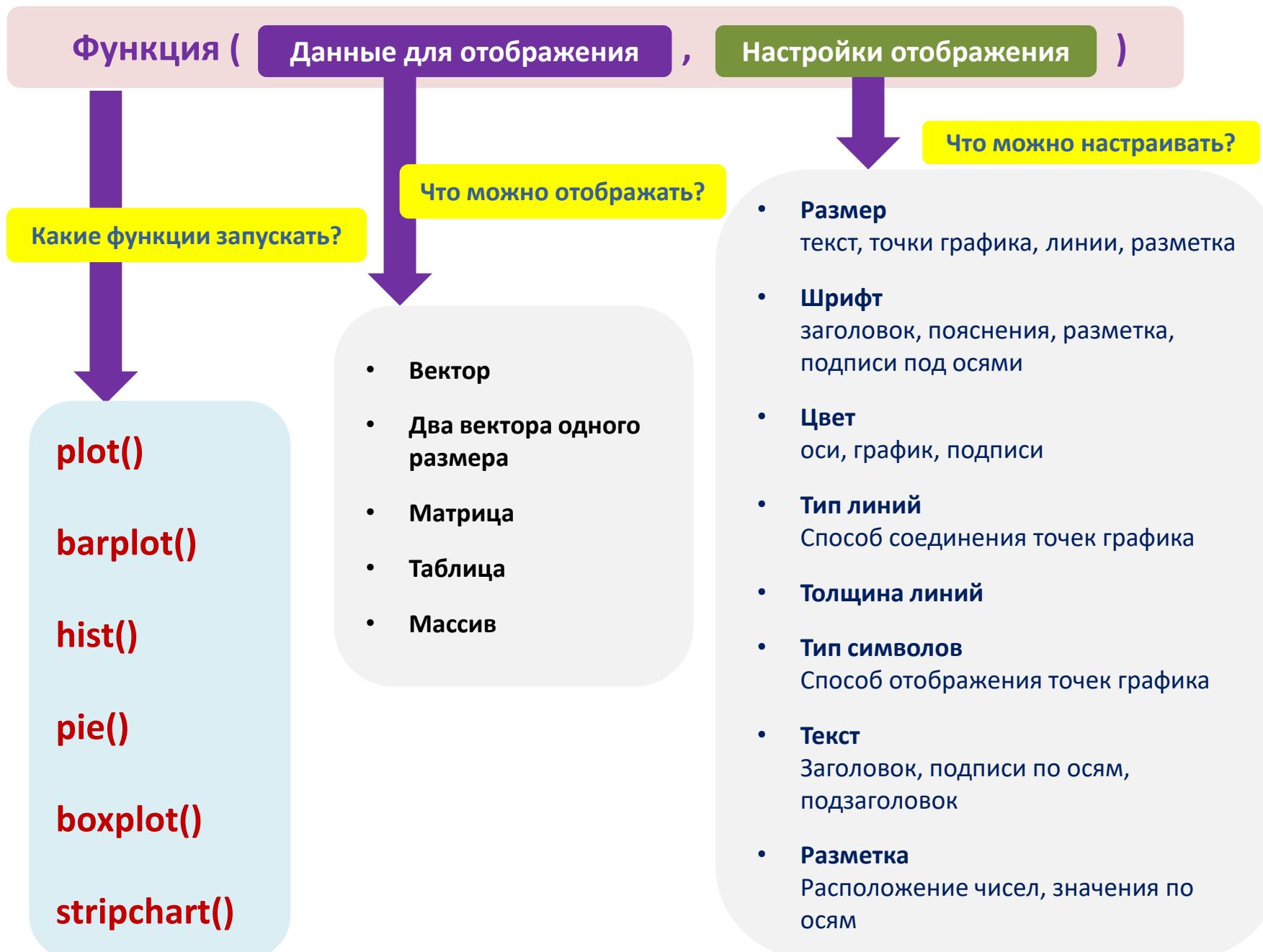
`plot(), barplot(), hist(), pie(), boxplot(), stripchart(), ...`

Вспомогательные графические функции

- всегда вызываются ПОСЛЕ основной функции
- размещают что-либо на готовом графике
- дополняют или меняют готовый график
- основная функция может ничего не отображать

- `legend()` – вывод поясняющего сообщения
- `abline()` – рисование опорных линий
- `lines()` – рисование новых зависимостей линиями
- `points()` – рисование новых зависимостей точками
- `axis()` – рисование осей
- `title()` – вывод заголовка
- ...

Основные графические функции R и их параметры



Функция `plot()` – основная графическая функция в R

`plot(Данные для отображения , Настройки отображения)`

Что можно отображать?

- **Вектор
`plot(x)`**
Значения по y – элементы вектора, значения по x – номера элементов вектора
- **Два вектора одного размера
`plot(x, y)`**
Значения по y – элементы вектора y, значения по x – элементы вектора x

Что можно настраивать?

- **Текст (`main, sub, xlab, ylab`)**
Заголовок, подписи по осям, подзаголовок
- **Размер (`cex, cex.axis, cex.lab, cex.main, cex.sub`)**
текст, точки графика, линии, разметка
- **Цвет (`col.axis,, fg, bg`)**
оси, график, подписи
- **Тип соединений точек графика (`type`)**
- **Тип линий (`lty`)**
Варианты рисования линий между точками графика
- **Толщина линий (`lwd`)**
- **Тип символов (`pch`)**
Способ отображения точек графика
- **Выравнивание (`adj`)**
Расположение чисел, значения по осям
- **Шрифт (`family, font, font.axis, font.lab, font.main, ...`)**
заголовок, пояснения, разметка, подписи под осями

Настройки отображения. Задание размера элементов

cex =

Параметр	Результат применения
cex	Коэффициент изменения размера элемента. По умолчанию cex = 1 cex = 1.5 означает, что элемент на 50% больше cex = 0.5 – на 50% меньше
cex.axis	Размер цифр на осях по отношению к cex
cex.lab	Размер названий осей по отношению к cex
cex.main	Размер заголовков по отношению к cex
cex.sub	Размер подзаголовков по отношению к cex

type =

"**p**" (**points**) отображаем точки

"**l**" (**lines**) рисуем линии, проходящие через точки, точки не отображаются

"**b**" (**both**) отображаем точки, рисуем линии между ними

"**c**" рисуем линии, вместо точек - пробелы

"**o**" аналог "**b**"

"**h**" (**high density**) рисуем вертикальные линии до точек

"**s**" (**steps**) рисуем ступеньки между точками, вариант 1

"**S**" (**steps**) рисуем ступеньки между точками, вариант 2

"**n**" (**no**) вообще ничего не рисуем

Настройки отображения. Символы для отображения точек на графике

pch =

```
plot(x,y, pch=15)
```

```
plot(20:-5, (20:-5)^2, pch = 17)
```

□ 0 ◇ 5 ⊕ 10 ■ 15 • 20 ▽ 25

○ 1 △ 6 ★ 11 • 16 ○ 21

△ 2 ✕ 7田 12▲ 17 □ 22

+ 3 * 8 ⊗ 13 ♦ 18 ◇ 23

✗ 4 ◆ 9 □ 14 ● 19 △ 24

Для символов с 21 по 25 можно отдельно указывать цвет контура
(border=) и заполнения (bg=)

lty = , lwd =

Параметр **lwd=** задает **толщину линии**

lwd= 2.5 – линия в два с половиной раза толще, чем по умолчанию

Параметр **lty=** задает **тип** рисуемой линии

1 – сплошная линия

2 – пунктирная линия

3 – линия из точек

4, 5, 6 – другие варианты



`plot(x,y, pch=15)`

`plot(20:-5, (20:-5)^2, pch = 17, lty=5, lwd= 2, type = 'b')`

Настройки отображения. Параметры задания цвета элементов

col.* = , fg = , bg =

Параметр	Результат применения
col.axis	Цвет значений на осях
col.lab	Цвет подписей на осях
col.main	Цвет заголовков
col.sub	Цвет подзаголовков
fg	Цвет графика
bg	Цвет фона

Перечень всех цветов – функция colors()

Для некоторых функций (**lines()**, **pie()**) можно указывать вектор из значений, которые используются по очереди. Если **col=c("red","blue")** и изображены три линии, первая будет красной, вторая – синей и третья – красной

Настройки отображения. Задание заголовков и подписей

main = , sub = , xlab = , ylab =

Параметр	Результат применения
main	Задает заголовок графика
sub	Поясняющая надпись под графиком
xlab	Подпись оси X
ylab	Подпись оси Y

Если на графике нужен поясняющий текст

ПОСЛЕ вызова **plot()**

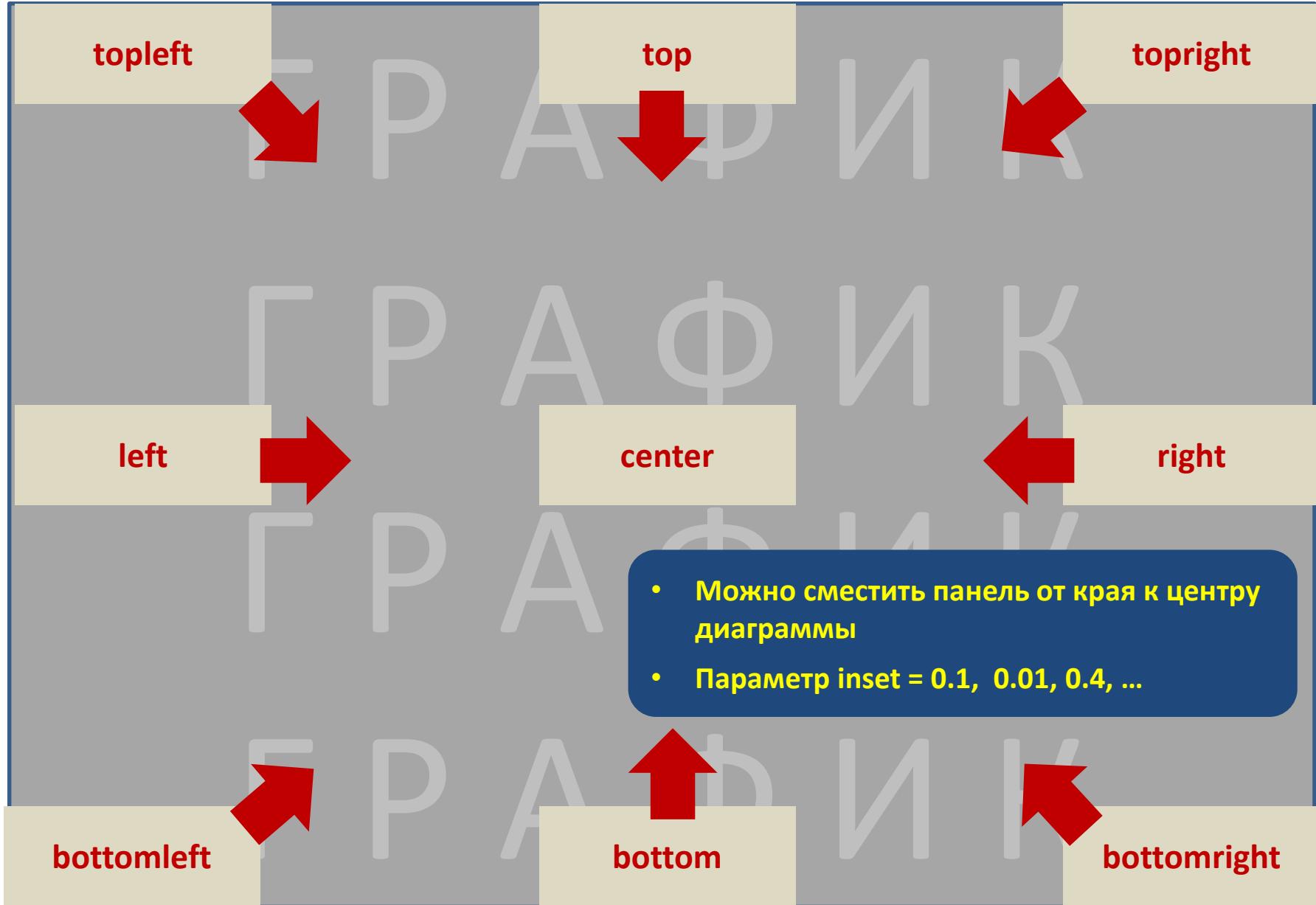
вызывается дополнительная функция **legend()**

Дополнительная функция legend() (информационная панель)

- Информационная панель (legend) показывает, что отображает каждый столбик, сектор круговой диаграммы или линия на графике
- Информационная панель добавляется на график вызовом функции **legend()**
- **legend(место, заголовок, расшифровка, ...)**

Параметр	Описание
место	Использование ключевых слов bottom, bottomleft, left, topleft, top, topright, right, bottomright, center
заголовок	Заголовок панели (необязательный параметр)
расшифровка:	Текстовый вектор с расшифровкой условных обозначений
fill =	Вектор с названиями цветов для раскраски
pch =	Вектор с номерами символов
lwd =	Вектор значений ширины линий
lty =	Вектор стиля линий
horiz = TRUE	Горизонтальное размещение панели

legend(location,)



dev

- В процессе работы программы может быть создано большое количество графиков и диаграмм
- По умолчанию каждый новый график затирает ранее созданный
- Необходимо иметь доступ ко всем построенным графикам
- Решение:

Создать и открыть новое устройство (device) для вывода графики, для этого

перед вызовом каждого нового графика использовать
вызов функции **dev.new()**

```
dev.new()
```

команды для построения диаграммы 1

```
dev.new()
```

команды для построения диаграммы 2

и т. д.

Функции для работы с графическими устройствами

Действие	Функция
Создать новое графического устройства	<code>dev.new()</code>
Получить список графических устройств (Имя – Номер)	<code>dev.list()</code>
Закрыть текущее устройство	<code>dev.off()</code>
Закрыть устройство N	<code>dev.off(N)</code>
Перейти к предыдущему устройству из списка	<code>dev.prev()</code>
Перейти к следующему устройству из списка	<code>dev.next()</code>
Перейти к устройству N	<code>dev.set(N)</code>
Узнать имя и номер текущего графического устройства	<code>dev.cur()</code>

Варианты настройки параметров отображения для графических функций

Многое можно задавать, менять, переопределять, настраивать

- Шрифты для вывода заголовка, пояснений, разметки, подписей под осями
- Цвета осей, графика, подписей
- Типы линий
- Представление точек на графике
- Размеры текста, точек, линий, разметки

Первый способ



Эффект действует для одного графика

- Все параметры задаются при вызове функции **plot()**
- При повторном вызове **plot()** эти же параметры нужно снова задавать
- Если все графики готовятся в одном стиле, это неудобно
- Нужно поменять некие общие настройки, чтобы эффект отразился на всех графиках

Второй способ



Эффект действует для всех графиков

- R запускается в определенном окружении (**Environment**)
- Environment** можно настраивать, меняя его параметры
- Параметры **Environment** действуют на работу всех функций, использующих эти параметры
- Задание параметров **Environment** реализуется через функцию **par()**
- Некоторые параметры менять нельзя (**readonly = TRUE**)
- Если (**no.readonly==TRUE**), можем задать свои параметры
- saveParam <- par(no.readonly=TRUE)**
....
par(saveParam)

Лукьянов Павел Борисович
профессор департамента Анализа данных, принятия решений
и финансовых технологий

Программирование в среде R

Лекция 5

R и интервальные данные.
Интервальная арифметика

Финансовый университет, 2020

Петля управления бизнесом. Использование R

Новые бизнес-идеи

Точные прогнозы и планирование невозможны

Почему ?

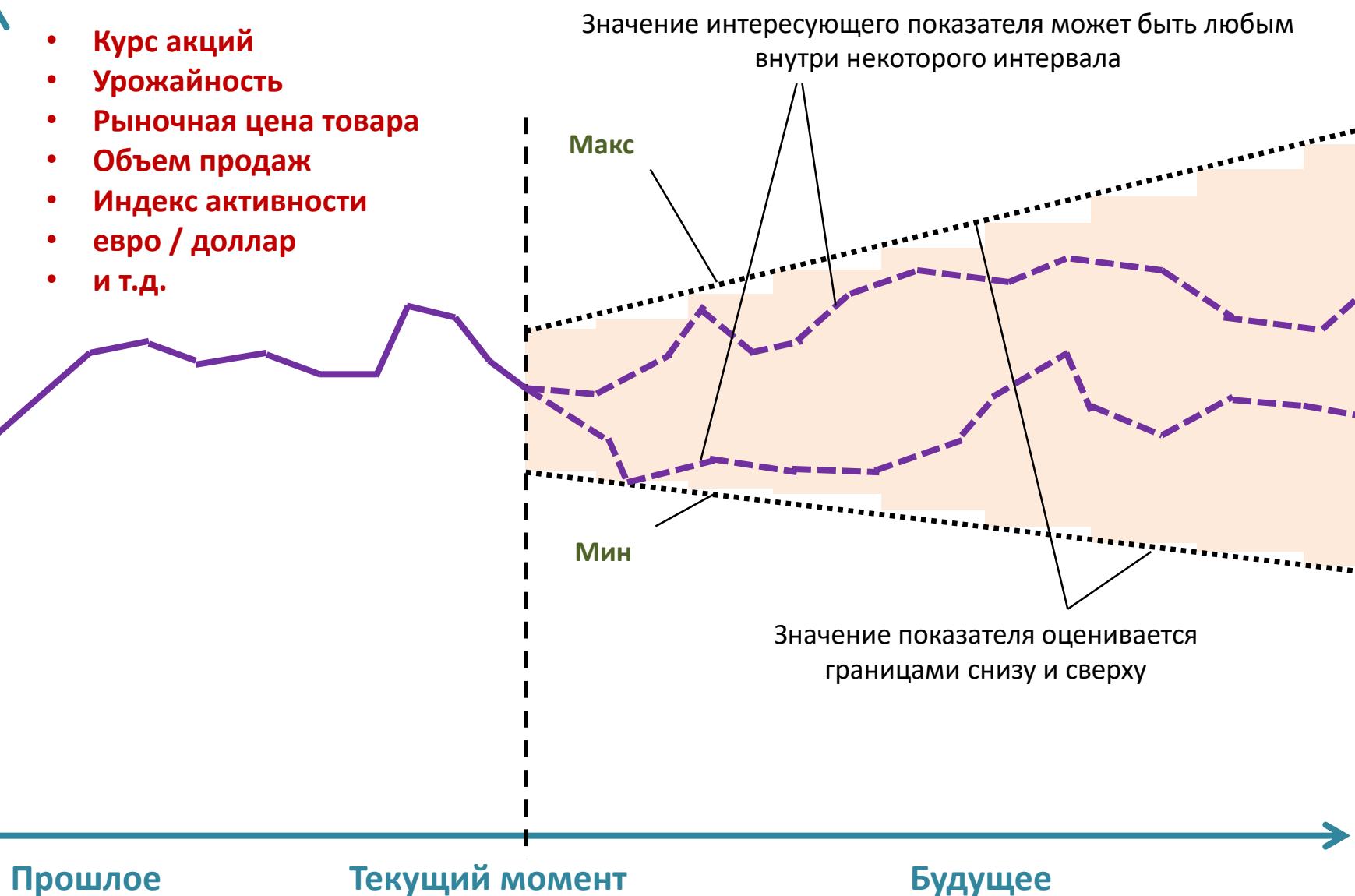


Точные расчеты и анализ возможны, если все исходные данные достоверны

Неопределенность будущего как свойство реального мира

Показатель

- Курс акций
- Урожайность
- Рыночная цена товара
- Объем продаж
- Индекс активности
- евро / доллар
- и т.д.



Пример экономической задачи, данные заданы точно

Дано

Компьютерная фирма разрабатывает и реализует сложные комплексы защиты корпоративных данных. Есть несколько заказов, есть оценка затрат, известна примерная цена реализации.

- Себестоимость одного комплекса $C = 5$ млн. руб
- Планируемый выпуск $Q_{prod} = 10$ шт.
- Цена реализации комплекса $P = 7$ млн. руб / шт.
- Объем реализации $Q_{sale} = 9$ шт.

Определить прибыль (Pr) и рентабельность (R) производства

Прибыль Pr и рентабельность R зависят от

- объема выпуска Q_{prod}
- суммы постоянных и переменных затрат C
- цены реализации товара P
- объема реализации Q_{sale}

$$Pr = f(C, Q_{prod}, P, Q_{sale})$$

$$R = g(C, Q_{prod}, P, Q_{sale})$$

Решение экономической задачи, данные заданы точно

Дано

Компьютерная фирма разрабатывает и реализует сложные комплексы защиты корпоративных данных. Есть несколько заказов, есть оценка затрат, известна примерная цена реализации.

- Себестоимость одного комплекса $C = 5$ млн. руб
- Планируемый выпуск $Q_{prod} = 10$ шт.
- Цена реализации комплекса $P = 7$ млн. руб / шт.
- Объем реализации $Q_{sale} = 9$ шт.

Определить прибыль (Pr) и рентабельность (R) производства

$$Pr = \text{Выручка} - \text{Затраты}$$

$$\text{Выручка} = \text{Цена} * \text{Объем реализации}$$

$$\text{Затраты} = \text{Себестоимость} * \text{Объем выпуска}$$

$$\text{Выручка} =$$

$$\text{Затраты} =$$

$$Pr =$$

$$R = \text{Прибыль} / \text{Затраты} * 100\%$$

$$\text{или } R = (\text{Выручка} / \text{Затраты} - 1) * 100\%$$

Пример задачи с интервальными данными. Постановка

Дано

Компьютерная фирма разрабатывает и реализует сложные комплексы защиты корпоративных данных. Есть несколько заказов, есть оценка затрат, известна примерная цена реализации.

- Себестоимость 1 комплекса $C = [5, 6]$ млн. руб
- Планируемый выпуск $Q_{prod} = [8, 12]$ шт.
- Цена реализации комплекса $P = [7, 10]$ млн. руб / шт.
- Объем реализации $Q_{sale} = [8, 12]$ шт.

Определить прибыль [Пр] и рентабельность [R] производства

Расчет выполнить, используя мин. и макс. оценки [Pr], [R]
через выбор соответствующих граничных значений исходных
данных

Учет неопределенности исходных данных

Традиционный подход к расчету экономических показателей

Прибыль Pr и рентабельность R зависят от

- объема выпуска Q_{prod}
- суммы постоянных и переменных затрат C
- цены реализации товара P
- объема реализации Q_{sale}

$$Pr = f(C, Q_{prod}, P, Q_{sale})$$

$$R = g(C, Q_{prod}, P, Q_{sale})$$

Если данные известны не точно, переход к интервалам

$$P \rightarrow [P] = [P_{min}, P_{max}]$$

$$Q_{prod} \rightarrow [Q_{prod}] = [Q_{prod_min}, Q_{prod_max}]$$

$$C \rightarrow [C] = [C_{min}, C_{max}]$$

$$Q_{sale} \rightarrow [Q_{sale}] = [Q_{sale_min}, Q_{sale_max}]$$

В результате расчета получаем интервалы возможных значений

$$[Pr] = f([C], [Q_{prod}], [P], [Q_{sale}])$$

$$[R] = g([C], [Q_{prod}], [P], [Q_{sale}])$$

Оценка предельных значений прибыли и рентабельности

Минимальная прибыль, минимальная рентабельность

$$Pr_{\min} = f(C_{\max}, Q_{\text{prod_max}}, P_{\min}, Q_{\text{sale_min}})$$

$$R_{\min} = g(C_{\max}, Q_{\text{prod_max}}, P_{\min}, Q_{\text{sale_min}})$$

Максимальная прибыль, максимальная рентабельность

$$Pr_{\max} = f(C_{\min}, Q_{\text{prod_max}}, P_{\max}, Q_{\text{sale_max}})$$

$$R_{\max} = g(C_{\min}, Q_{\text{prod_max}}, P_{\max}, Q_{\text{sale_max}})$$

Пример задачи с интервальными данными. Решение

Расчет через оценку границ исходных данных

$Pr = \text{Выручка} - \text{Затраты}$

Выручка = Цена * Объем реализации

Затраты = Себестоимость * Объем выпуска

$Pr_{\min} = f(P_{\min}, Q_{prod_max}, \text{Затр_макс}, V_{реал_мин})$

$Pr_{\max} = f(P_{\max}, Q_{вып_макс}, \text{Затр_мин}, V_{реал_макс})$

$Pr_{\min} = P_{\min} * Q_{sale_min} - C_{max} * Q_{prod_max}$

$Pr_{\min} =$

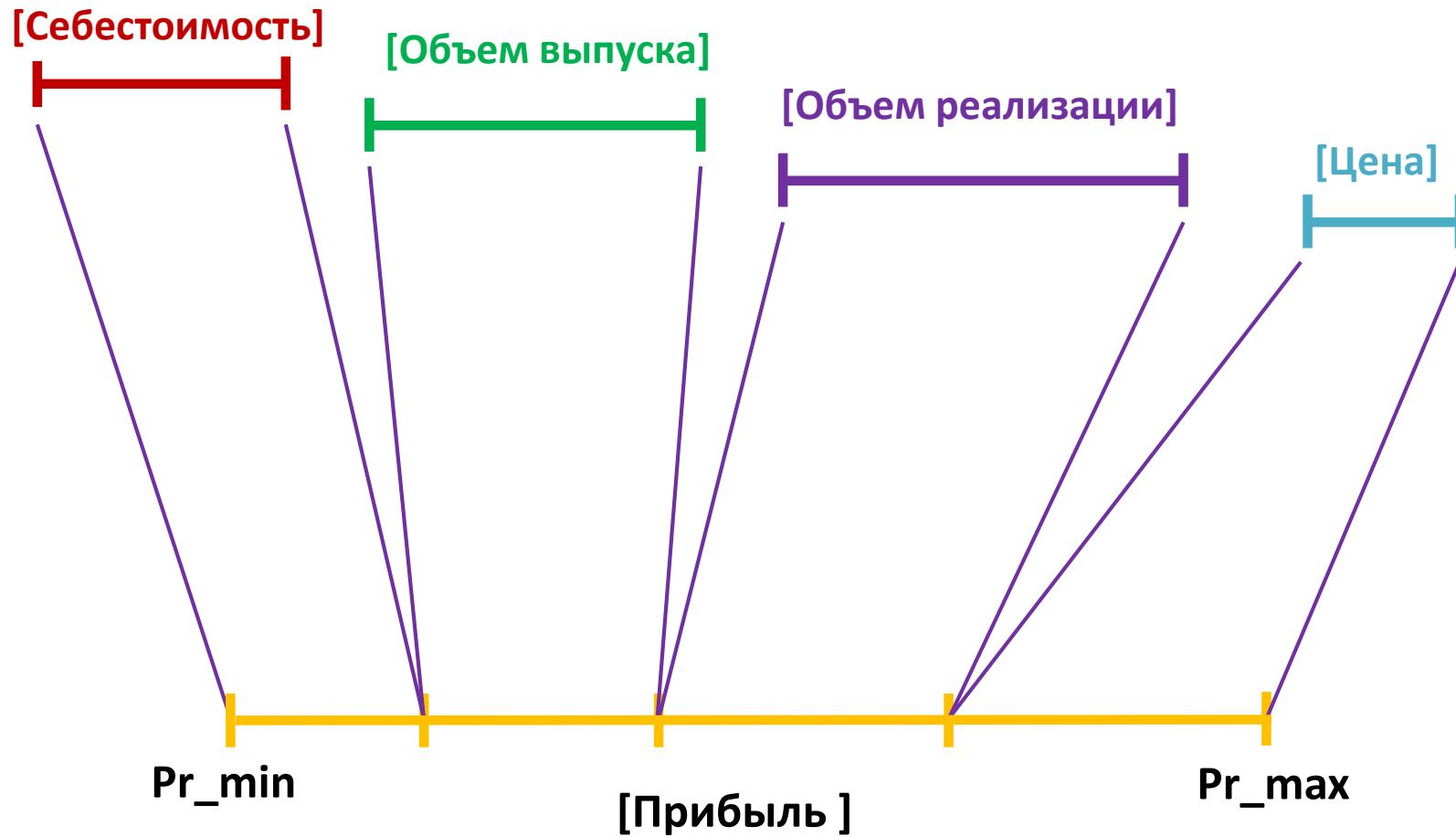
$Pr_{\max} = P_{\max} * Q_{sale_max} - C_{min} * Q_{prod_max}$

$Pr_{\max} =$

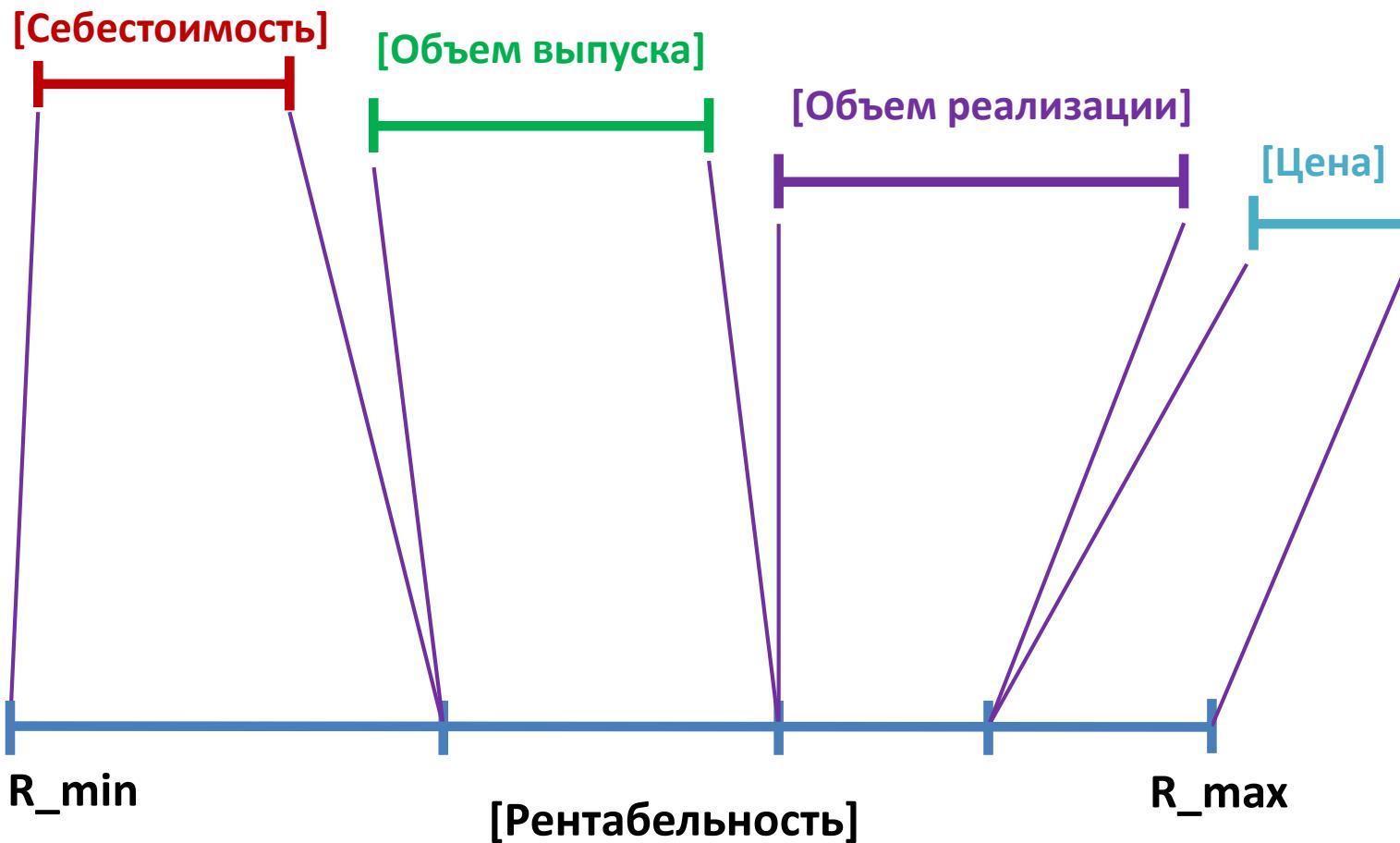
[Прибыль] =

[R] - ?

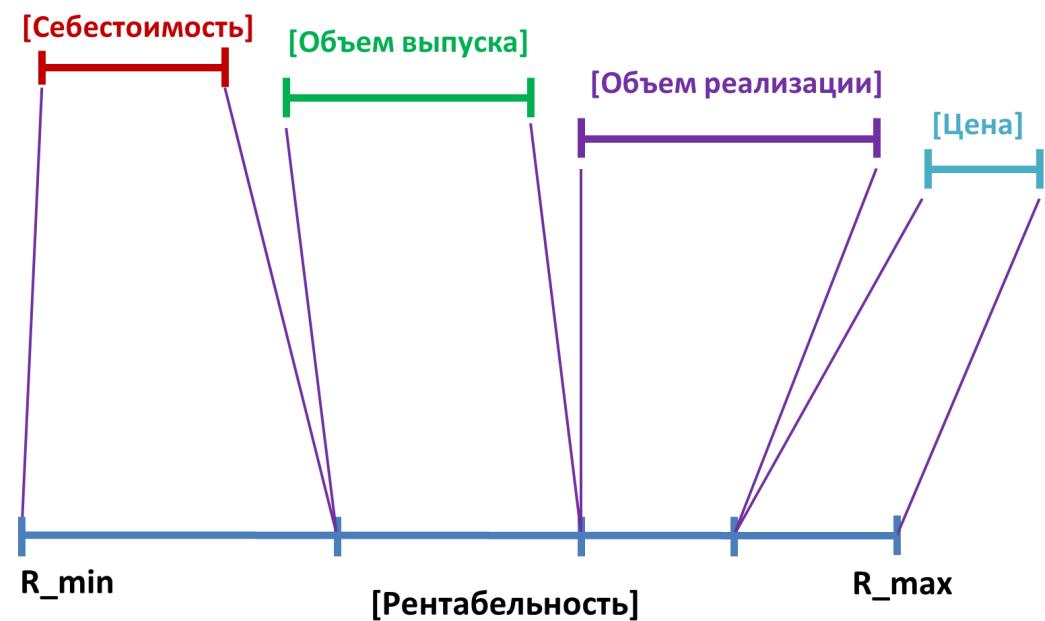
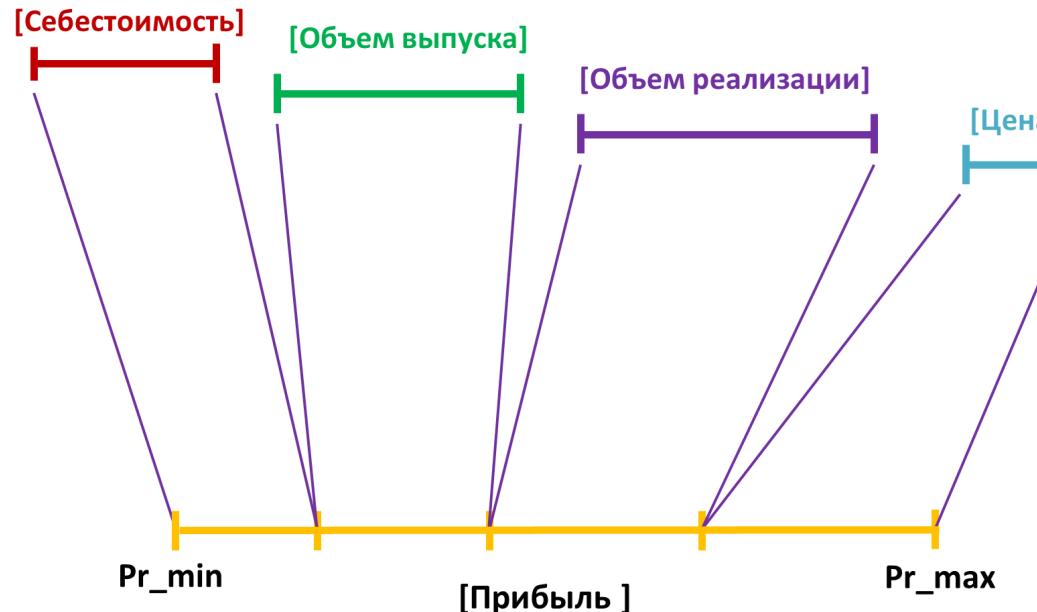
Возможный вклад неопределенности исходных данных в неопределенность решения



Возможный вклад неопределенности исходных данных в неопределенность решения



Вклад неопределенности исходных данных в неопределенность решения



Формулы интервальной арифметики

Вводится новое понятие - интервал

Число N заменяется интервалом: $N \rightarrow [N] = [a, b] \quad a \leq b$

Арифметические операции с интервалами

$$[a, b] + [c, d] = [a + c, b + d]$$

$$[a, b] - [c, d] = [a - d, b - c]$$

$$[a, b] * [c, d] = [\min(a*c, a*d, b*c, b*d), \max(a*c, a*d, b*c, b*d)]$$

$$[a, b] / [c, d] = [a, b] * [1/d, 1/c], \text{ причем интервал } [c, d] \text{ не содержит } 0$$

Свойства операций

Сложение и умножение коммутативны:

$$[a, b] + [c, d] = [c, d] + [a, b]$$

$$[a, b] * [c, d] = [c, d] * [a, b]$$

НО вычитание не обратно сложению, деление не обратно умножению

Примеры расчетов по формулам интервальной арифметики

$$[a, b] = [2, 5]$$

$$[c, d] = [1, 4]$$

$$[a, b] + [c, d] = [a + c, b + d] = [2, 5] + [1, 4] = [2+1, 5+4] = [3, 9]$$

$$[a, b] - [c, d] = [a - d, b - c] = [2, 5] - [1, 4] = [2 - 4, 5 - 1] = [-2, 4]$$

$$\begin{aligned} [a, b] * [c, d] &= [\min(a*c, a*d, b*c, b*d), \max(a*c, a*d, b*c, b*d)] = \\ &= [\min(2*1, 2*4, 5*1, 5*4), \max(2*1, 2*4, 5*1, 5*4)] = \\ &= [\min(2, 8, 5, 20), \max(2, 8, 5, 20)] = [2, 20] \end{aligned}$$

$$\begin{aligned} [a, b] / [c, d] &= [a, b] * [1/d, 1/c] = [2, 5] * [1/4, 1/1] = [2, 5] * [1/4, 1] = \\ &= [\min(1/2, 2, 5/4, 5), \max(1/2, 2, 5/4, 5)] = [1/2, 5] \end{aligned}$$

Примеры расчетов по формулам интервальной арифметики

$$[a, b] = [-2, 5]$$

$$[c, d] = [-1, 4]$$

$$[a, b] + [c, d] = [a + c, b + d] = [-2, 5] + [-1, 4] = [-2 - 1, 5 + 4] = [-3, 9]$$

$$[a, b] - [c, d] = [a - d, b - c] = [-2, 5] - [-1, 4] = [-2 - 4, 5 - (-1)] = [-6, 6]$$

$$\begin{aligned} [a, b] * [c, d] &= [\min(-2 * (-1), -2 * 4, 5 * (-1), 5 * 4), \max(-2 * (-1), -2 * 4, 5 * (-1), 5 * 4)] = \\ &= [\min(2, -8, -5, 20), \max(2, -8, -5, 20)] = [-8, 20] \end{aligned}$$

$$\begin{aligned} [a, b] / [c, d] &= [a, b] * [1/d, 1/c] = [-2, 5] * [1/4, -1/1] = [-2, 5] * [1/4, -1] = \\ &= [\min(-1/2, 2, 5/4, -5), \max(-1/2, 2, 5/4, -5)] = [-5, 5/4] \end{aligned}$$

Так считать нельзя

Особенности расчетов по формулам интервальной арифметики

$$[a, b] = [2, 5]$$

$$[c, d] = [1, 4]$$

Ширина исходного интервала $[a, b] = 5 - 2 = 3$

Ширина исходного интервала $[c, d] = 4 - 1 = 3$

Ширина $[a, b] + [c, d] = 9 - 3 = 6$

Ширина $[a, b] - [c, d] = 4 - (-2) = 6$

Ширина $[a, b] * [c, d] = 20 - 2 = 18$

Ширина $[a, b] / [c, d] = 5 - 1/2 = 4.5$

Пример задачи с интервальными данными. Постановка

Дано

Компьютерная фирма разрабатывает и реализует сложные комплексы защиты корпоративных данных. Есть несколько заказов, есть оценка затрат, известна примерная цена реализации.

- Себестоимость 1 комплекса $C = [5, 6]$ млн. руб
- Планируемый выпуск $Q_{prod} = [8, 12]$ шт.
- Цена реализации комплекса $P = [7, 10]$ млн. руб / шт.
- Объем реализации $Q_{sale} = [8, 12]$ шт.

Определить прибыль [Пр] и рентабельность [R] производства

Расчет выполнить, используя формулы интервальной арифметики

Сравнить и объяснить полученные результаты, выполненные двумя способами

Пример задачи с интервальными данными. Решение

Расчет по формулам интервальной арифметики

$Pr = \text{Выручка} - \text{Затраты}$

$\text{Выручка} = \text{Цена} * \text{Объем реализации}$

$\text{Затраты} = \text{Себестоимость} * \text{Объем выпуска}$

$$[\text{Выручка}] = [7, 10] * [8, 12] = [\min(56, 84, 80, 120), \max(56, 84, 80, 120)]$$

$$[\text{Выручка}] =$$

$$[\text{Затраты}] = [5, 6] * [8, 12] = [\min(40, 48, 60, 72), \max(40, 48, 60, 72)]$$

$$[\text{Затраты}] =$$

$$[\text{Прибыль}] = [56, 120] - [40, 72] =$$

$$[R] - ?$$

млн. руб

Сравнение результатов

Расчет по формулам интервальной арифметики

[Прибыль] = [-16, 80] млн. руб

[Рентабельность] = [-40, 200] %

Расчет через оценку границ исходных данных

[Прибыль] = [-16, 60] млн. руб

[Рентабельность] = [-22.2, 100] %

Функции библиотеки interval.R

Функция	Описание
<code>is.interval(x)</code>	проверка, может ли переданный вектор x быть интервалом
<code>as.interval(x)</code>	преобразование вектора x в интервал
<code>plus.i (x, y)</code>	сложение интервала x с интервалом y
<code>minus.i (x, y)</code>	вычитание интервала y из интервала x
<code>mult.i (x,y)</code>	умножение интервала x на интервал y
<code>div.i (x,y)</code>	деление интервала x на интервал y

Лукьянов Павел Борисович
профессор департамента Анализа данных, принятия решений
и финансовых технологий

Программирование в среде R

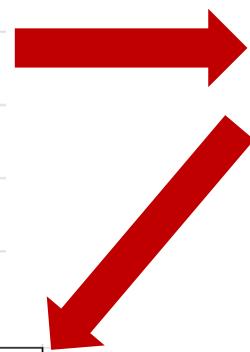
Лекция 6

Моделирование и принятие решений.
Числовые и категориальные данные

Финансовый университет, 2020

Механизм формирования баллов за работу в семестре

Период обучения	Баллы, макс
февраль-апрель	30
май	10
Итого	40



Вид обучения	Баллы, макс
Текущая работа	15
Выполнение и сдача КР	15
Итого	30

Текущая работа	Баллы, макс
Посещение занятий	5
Работа на семинарах	5
Выполнение домашних работ	5
Итого	15



Выполнение и сдача КР	Баллы, макс
Оформление отчета, скриптов и кода (пояснения, кодировка, форматирование)	
Качество кода (объем кода, наличие/отсутствие циклов, качество графиков и т.д.)	
Оригинальность кода (не скопирован/скопирован у товарища)	
Наличие комментариев для лучшего понимания работы частей программы	
Полнота выполнения заданий	
Срок отправки итоговой версии: 1 - вовремя, 0 - опоздал	
Использование самостоятельно изученных альтернативных пакетов (напр. ggplot2 и др)	
Итого	15

Итоговая таблица расчета баллов за работу в феврале - апреле

	Показатели	Баллы	
1	Посещение (присутствие) на семинарах и лекциях	0-5	5
2	Регулярная отправка работ после завершения семинара (что успел сделать)	0-5	5
3	Регулярная отправка выполненных заданий до следующего семинара (все самостоятельные задания из методички)	0-5	5
4	Отчет по КР. Оформление отчета, скриптов и кода (пояснения, кодировка, форматирование)	0-1	1
5	Отчет по КР. Качество кода (объем кода, наличие/отсутствие циклов, качество графиков и т.д.)	0-3	3
6	Отчет по КР. Оригинальность кода (не скопирован/скопирован у товарища)	0-3	3
7	Отчет по КР. Наличие комментариев для лучшего понимания работы частей программы	0-1	1
8	Отчет по КР. Полнота выполнения заданий	0-4	4
9	Отчет по КР. Срок отправки итоговой версии: 1 - вовремя, 0 - опоздал	0-1	1
10	Отчет по КР. Использование самостоятельно изученных альтернативных пакетов (напр. ggplot2 и др)	0-2	2
			30

Принятие управленческого решения

4



Принятие решений в петле управления бизнесом



Модель и моделирование

Моделирование – процесс создания модели, т.е. отражения реальности, более простой, чем сама реальность

- Модель – упрощенное представление о реальном объекте
- В модели отражаются основные свойства и характеристики объекта
- Чем больше параметров объекта учитывает модель, тем она сложнее и тем точнее описывает объект и предсказывает его поведение

Модель нужна, чтобы:

- Исследовать характеристики реального объекта:
 - его структуру и внутренние связи
 - основные свойства
 - законы развития и взаимодействия с другими объектами
- Научиться управлять объектом или процессом, определять наилучшие способы и параметры управления при заданных целях и критериях
- Прогнозировать последствия воздействия на объект разными способами

Прикладная область для моделирования. Игра на бирже



Принятие решений о покупке / продаже акций принимается инвестором на основании

- Динамики цены акции за прошедший период
 - Технического и фундаментального анализа рынка ценных бумаг
 - Прогнозов экспертов
 - Событий в мире и экономике
 - Настроения на рынке инвесторов и акционеров
 - Личных предпочтений, интуиции, настроения и т.д.

Цели и задачи инвестора

Цель:

- Получать прибыль от игры на бирже

Задачи:

- Предугадывать поведение цены акции
- Принимать решения о покупке или продаже акций в нужные периоды времени

Решение

Создание специализированного программного обеспечения (ПО) для

1. Моделирования поведения цены акции
2. Моделирования поведения Инвестора
3. Планирования решений о покупке или продаже акций
4. Оценки эффективности решений (оценка прибыли)

Использование
языка R

Моделирование поведения цены акции

Цель – разработка математических функций, отражающих основные закономерности в изменении цены акции



1. Случайные колебания цены внутри некоторого диапазона



2. Плавное и периодическое изменение цены в течение некоторого периода

USDRUB_TOM 74,0250 74,1275 73,1000 73,4000 2,2 млрд. ⏰ день



Роснефть 344,05 350,00 340,25 348,00 9,5 млн. ⏰ день



БСМПО-АВСМ 18080 18200 17980 18000 771 ⏰ день



3.1. ГЭП (GAP) – резкое изменение цены акции (падение, взлет)



3.2. ГЭП – разрыв цены акции между торговыми сессиями



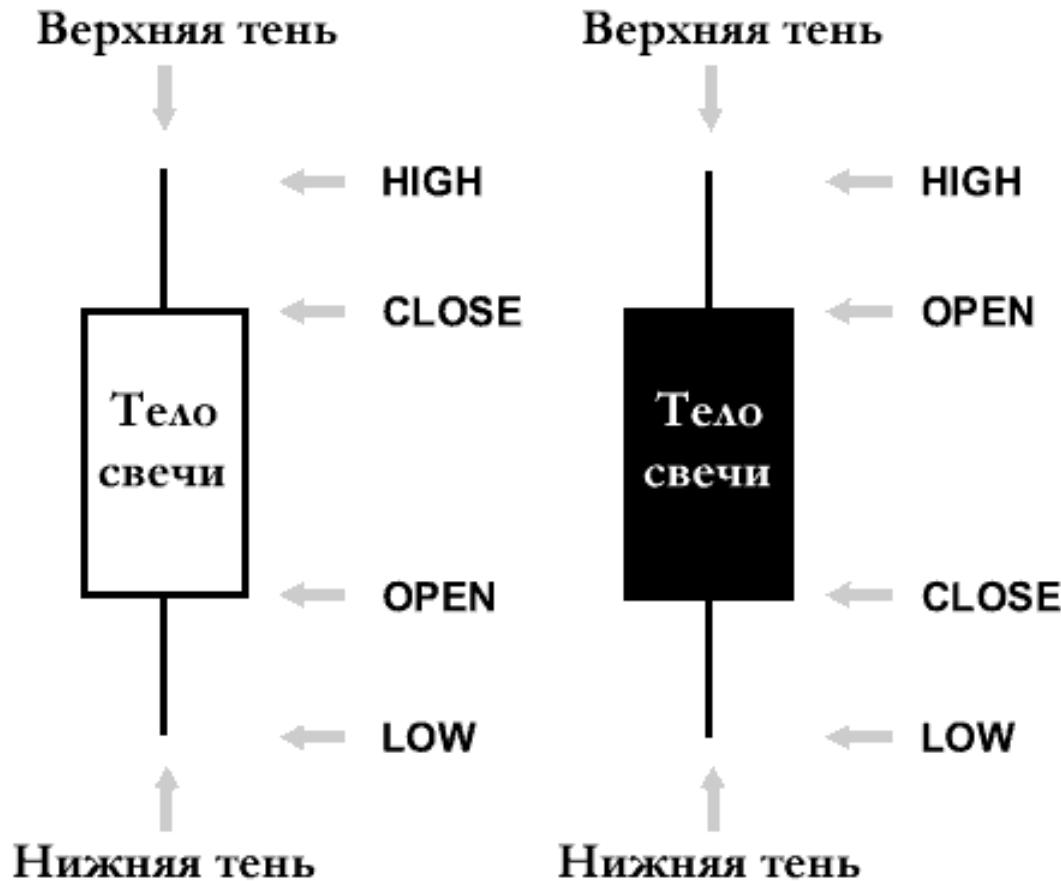
3.3. ГЭП. Разрыв цены при непрерывной торговле



4. Плавный рост или снижение цены акции



Что такое японская свеча на графике



Реальные котировки акций. Представления в виде японских свечей



Реальные котировки акций:

www.finam.ru

Анализ. Составляющие поведения цены акции

На основании анализа поведения цен выделяют следующие составляющие

1. Случайные колебания цены внутри некоторого диапазона
 $\text{price1} = f1(\dots)$
2. Плавное и периодическое изменение цены в течение некоторого периода
 $\text{price2} = f2(\dots)$
3. Резкие и значительные скачки цены вверх или вниз (гэпсы)
 $\text{price3} = f3(\dots)$
4. Плавный, устойчивый рост или снижение цены
 $\text{price4} = f4(\dots)$

Динамика изменения цены price – суперпозиция функций:
 $\text{price} = g(\text{price1}, \text{price2}, \text{price3}, \text{price4})$

Упрощения и допущения при моделировании цены

Представим функцию $g()$ как сумму составляющих функций $\text{price1}, \dots, \text{price4}$ в допущении, что функции $f1(), \dots, f4()$ независимы друг от друга, не влияют друг на друга

$$\text{price} = W1 * \text{price1} + W2 * \text{price2} + W3 * \text{price3} + W4 * \text{price4}$$

где $W = \{W1, W2, W3, W4\}$ – набор весовых коэффициентов, задающих вклад каждой составляющей в price

1.1. Механизм формирования случайной составляющей цены акции price1

Функция **random.price()** возвращает случайную составляющую цены **price1**

- Цена акции меняется один раз в день, в течение дня цена постоянна
- Цена акции на текущий день **price1(today)** зависит от цены акции днем ранее **price1(today-1)**
- Цена акции **price1(today)** зависит от двух коэффициентов **k1, k2: k1 <= k2**
- **k1** отражает вероятность негативного сценария формирования цены, **k2** отражает вероятность позитивного сценария
- значения **price1(today-1), k1** и **k2** передаются в функцию **randomPrice()** как параметры

Первый способ генерации цены

- цена **price1(today)** может принимать любое значение в диапазоне **H:**

$$\text{price1(today)} \in [H],$$

$$[H] = [k1, k2] * \text{price1(today-1)}$$

1.2. Механизм формирования случайной составляющей цены акции price1

Функция **randomPrice()** возвращает случайную составляющую цены **price1**

Второй способ генерации цены

- цена **price1(today)** может принимать любое значение в диапазоне **H**:

$$\text{price1(today)} \in [H],$$

$$[H] = [\text{price1(today-1)} - k1, \text{price1(today-1)} + k2]$$

Третий способ генерации цены

- цена **price1(today)** может принимать любое значение в диапазоне **H**:

$$\text{price1(today)} \in [H],$$

$$[H] = [\text{price1(today-1)} * (1 - k1), \text{price1(today-1)} * (1 + k2)]$$

Игра на бирже. Поведение инвестора

Суть игры на бирже - спекуляция

Покупка и продажа акций различных компаний с целью получения прибыли:

Акции компании А

- Купил дешево -> продал дорого
- Продал дорого -> купил дешево

Акции компании В

- Купил дешево -> продал дорого
- Продал дорого -> купил дешево

Акции компании С

- Купил дешево -> продал дорого
- Продал дорого -> купил дешево

В игре на бирже высокая доходность и большие риски

В изменении цен акций присутствует **цикличность**, что позволяет инвестору с некоторой вероятностью предугадывать движение цены акции и **зарабатывать** на этом или **терять**, если динамика цены не совпала с ожидаемой

Постановка задачи

- Вы – инвестор. Для игры на бирже, для покупки и продажи акций различных компаний у инвестора на старте, в первый день торгов $t=1$, имеются финансовые средства в размере $Q=Q(t=1)$ руб
- На бирже торгуются акции N компаний. У каждой компании цена акции $P_i(t)$ ($i=1, \dots, N$) меняется каждый день на протяжении периода торгов
- Инвестору известны экспертные оценки по динамике стоимости акций на некотором временном промежутке
- Инвестор, начиная с текущего дня, в течение определенного периода (количество дней $T=100$) покупает и продает акции различных компаний, **используя некоторый алгоритм**. Текущие финансовые средства после покупки и продажи акций меняются постоянно: $Q = Q(t)$
- В последний день периода $t = T$ инвестор продает все ранее купленные акции, подсчитывает выручку $Q(T)$ и рассчитывает полученную прибыль от игры на бирже (фиксирует прибыль):

$$Pr = Q(T) - Q(t=1)$$

Принятие решения о покупке или продаже акции. Функция `vote()`

Функция `vote()`

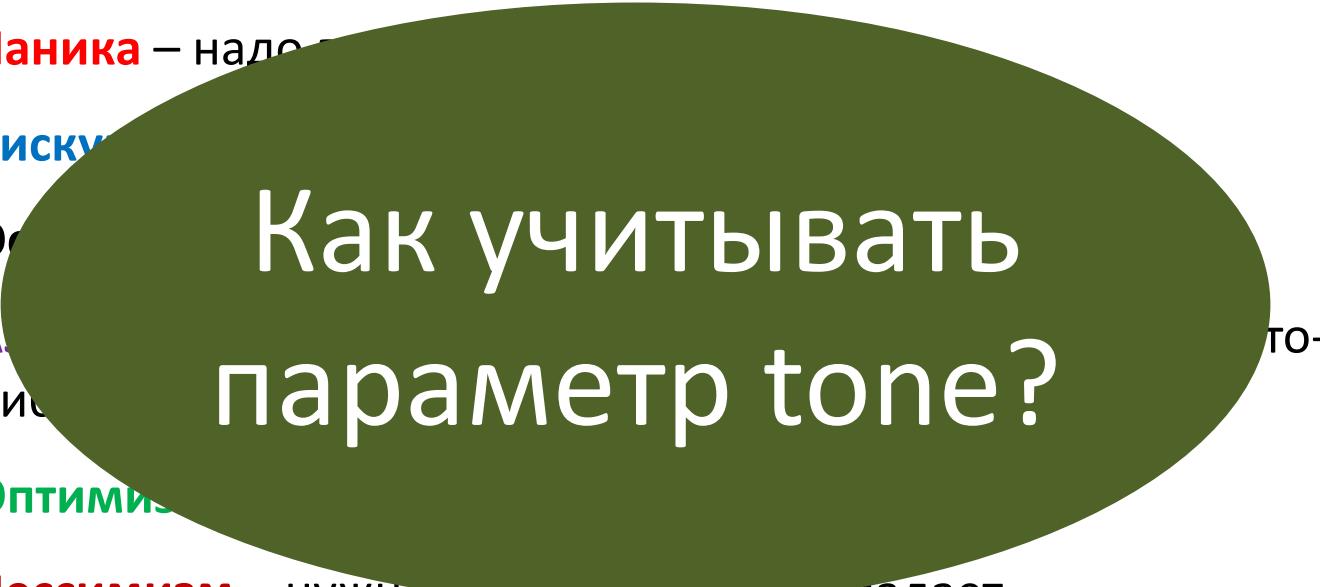
- Решение о покупке или продаже акции принимается инвестором на основании результата работы функции `vote()`, которая для акций разных компаний ($i=1, \dots, N$) с разной вероятностью возвращает одно из двух значений:
- **1 (покупать): позитивный** сценарий
Инвестор ожидает, что стоимость акций будет **растить**
- **-1 (продавать): негативный** сценарий
Инвестор ожидает, что стоимость акций будет **падать**
- Поведение функции `vote()` определяется управляющим параметром `tone`, от значения которого зависит результат, возвращаемый `vote()`
- Параметр `tone` отражает настроение инвестора
- Параметр `tone` в общем случае имеет **не числовую природу**

Параметр tone

Параметр **tone** в общем случае имеет **не числовую природу**

Варианты параметра

- **Паника** – надо продавать
- **Рискунок** – надо покупать
- **Оптимист** – надо покупать
- **Алгоритмический трейдер** – надо покупать
- **Оптимизация** – надо покупать
- **Пессимизм** – нужно продавать, что падает
- **Расчет** – мои графики мне говорят покупать (продавать)
- **Авось** – что-то надо купить и продать, или продать и купить..
- **«цвет настроения синий...»** - параметр настроения в песне
- ...



Как учитывать
параметр tone?

Фундамент любого управленческого решения – непротиворечивый и полный набор данных, описывающий исследуемое явление или хозяйственную деятельность

Получение данных

- Загрузка данных различной природы из разных источников в переменные (контейнеры данных) компьютерной программы
- Согласование форматов и представлений разных типов данных между собой
- Формирование набора взаимосвязанных исходных массивов, таблиц, структур, описывающих исследуемую прикладную область

Типы данных в R

- **Числовые (numeric):** целые и действительные числа
- **Строковые (character):** текст, имена, символы, описания
- **Логические (logical):** только два возможных значения, Истина / Ложь
- **Комплексные:** с ними не работаем
- **Необработанные (последовательность байт):** приходят из внешних источников, необходима обработка и интерпретация (извлечение строк, чисел, значений)

Типы данных, встречающиеся в исследованиях, экспериментах, деятельности

Первичные данные



Вторичные данные

Результаты обработки и преобразований

дели, проценты, расчетные значения, коэффициенты, числовые параметры, ...

Первичные данные: числовые данные

Числовые данные

Данные, меняющиеся непрерывно (интервальные данные)

- **Примеры:** масса, возраст, расстояние, температура, процентная ставка, зарплата и т.д.
- Непрерывные данные выражаются **действительными числами**, могут принимать любое значение внутри некоторого диапазона
- Значения двух переменных можно **упорядочить**, узнать, насколько одно значение больше другого
- Переменные можно **отобразить на мерной шкале**
- **Больше всего методов обработки** данных разработано для интервальных данных

Данные, меняющиеся дискретно (целочисленные данные)

- **Примеры:** годы, количество сотрудников в организации, число детей в семье, количество мест в самолете, число планет и т.д.
- Выражаются, как правило, целыми числами
- Промежуточных значений может не быть
- Значения двух переменных можно **упорядочить**
- Переменные можно **отобразить на мерной шкале**

R: тип `integer`

R: тип `double`

Первичные данные: нечисловые данные, порядковые

Нечисловые данные

Категориальные данные

Порядковые (шкальные) данные

Факторы

- **Примеры**
Состояние больного (тяжелое, средней тяжести, легкое недомогание, здоров)
Погода (ясно, облачно, пасмурно, дождь, буря, ураган)
Посещаемость занятий (всегда, часто, средне, редко, никогда)
- Придумывается **шкала, которая что-то отражает** (успеваемость, удобство, качество обслуживания и т.д.).
- Различным **состояниям** на шкале **присваиваются баллы**.
- Баллы условны: оценки школьнику, студенту в России – от 2 до 5, в Италии – от 1 до 30
- Каждому баллу соответствует некоторое описание,дается ответ о совпадении
- Можно ввести **отношение порядка** (холодно -> прохладно -> тепло -> жарко), но значения нельзя оценить количественно
- Существует проблема совместимости данных из разных шкал
- **Общая задача:** уходить от шкальных данных к интервальным, т.к. для интервальных данных разработано гораздо больше различных методов обработки данных, качество анализа выше.
Характеристика глубины моря около пляжа – от «Мелко», «Глубоко», «Опасно», «Безопасно»
к точным измерениям глубины

Первичные данные: нечисловые данные, номинальные



В R факторы создаются из векторов

Функции для работы с факторами

- Пусть имеется текстовый вектор **w** с набором категориальных данных
`w <- c('ясно', 'ясно', 'облачно', 'дождь', 'дождь', 'облачно', 'ясно', 'ураган')`

Функция **factor()** сохраняет категориальные данные в виде вектора из целых чисел от 1 до N, где N – число уникальных значений категориальной переменной

Пусть имеется числовой вектор **b** с набором категориальных данных

Используется функция **cut(x, breaks, ordered = TRUE)**, она делит вектор **x** на равные интервалы (факторы)

в качестве **breaks** может выступать либо необходимое число интервалов, либо вектор, который содержит перечень точек разрыва

```
cut(x, breaks=3, labels = letters[1:3])
```

- Конструирование фактора «напрямую»

```
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.

Реализация алгоритма торговли

Используются функции **price.stock()**, **vote()**, **quant()**

В цикле от $t = 1$ до T выполняем действия:

- **Шаг 1.** Принимается решение о покупке или продаже акции
Задается настроение торгов **tone**, вызывается функция **vote(tone)**
- **Шаг 2.** Рассчитывается текущая цена акции.
Вызывается функция **generate.price()**
- **Шаг 3.** Определяется количество акций для покупки или продажи. Вызывается функция **quant()**
- **Шаг 4.** Пересчитывается количество текущих денежных средств и акций инвестора

После завершения цикла все акции продаем по цене последнего дня

Считаем выручку $Q(T)$ и прибыль $Pr = Q(T) - Q(1)$

Лукьянов Павел Борисович
профессор департамента Анализа данных, принятия решений
и финансовых технологий

Программирование в среде R

Лекция 7

Характеристики выборки.
Специальные графические функции

Финансовый университет, 2020

Типы данных, встречающиеся в исследованиях, экспериментах, деятельности

Первичные данные



Вторичные данные

Результаты обработки и преобразований

дели, проценты, расчетные значения, коэффициенты, числовые параметры, ...

Первичные данные: числовые данные

Числовые данные

Данные, меняющиеся непрерывно (интервальные данные)

- **Примеры:** масса, возраст, расстояние, температура, процентная ставка, зарплата и т.д.
- Непрерывные данные выражаются **действительными числами**, могут принимать любое значение внутри некоторого диапазона
- Значения двух переменных можно **упорядочить**, узнать, насколько одно значение больше другого
- Переменные можно **отобразить на мерной шкале**
- **Больше всего методов обработки** данных разработано для интервальных данных

Данные, меняющиеся дискретно (целочисленные данные)

- **Примеры:** годы, количество сотрудников в организации, число детей в семье, количество мест в самолете, число планет и т.д.
- Выражаются, как правило, целыми числами
- Промежуточных значений может не быть
- Значения двух переменных можно **упорядочить**
- Переменные можно **отобразить на мерной шкале**

R: тип `integer`

R: тип `double`

Неопределенность будущего как свойство реального мира

Показатель

- Курс валют
- Уровень инфляции
- Динамика цен на нефть
- Европейский долг
- И т.д.

Как принимать решения в
условиях
неопределенности?

Будущее может быть любым
и предсказать его невозможно

Как

минимизировать
риски?

Прошлое

Текущий момент

Будущее



Первичные данные: нечисловые данные, порядковые

Нечисловые данные

Категориальные данные

Порядковые (шкальные) данные

Факторы

- **Примеры**
Состояние больного (тяжелое, средней тяжести, легкое недомогание, здоров)
Погода (ясно, облачно, пасмурно, дождь, буря, ураган)
Посещаемость занятий (всегда, часто, средне, редко, никогда)
- Придумывается **шкала, которая что-то отражает** (успеваемость, удобство, качество обслуживания и т.д.).
- Различным **состояниям** на шкале **присваиваются баллы**.
- Баллы условны: оценки школьнику, студенту в России – от 2 до 5, в Италии – от 1 до 30
- Каждому баллу соответствует некоторое описание,дается ответ о совпадении
- Можно ввести **отношение порядка** (холодно -> прохладно -> тепло -> жарко), но значения нельзя оценить количественно
- Существует проблема совместимости данных из разных шкал
- **Общая задача:** уходить от шкальных данных к интервальным, т.к. для интервальных данных разработано гораздо больше различных методов обработки данных, качество анализа выше.
Характеристика глубины моря около пляжа – от «Мелко», «Глубоко», «Опасно», «Безопасно»
к точным измерениям глубины

Первичные данные: нечисловые данные, номинальные



Задача на расчет показателей выборки

Постановка задачи

Получена выборка: 2, 14, 5, 7, -3, 7, 11, 6, 0.

Рассчитать следующие показатели: медиану, первый и третий квартили QQR.

Определить выбросы. Отметить рассчитанные значения на графике boxplot.

Наблюдения, исследования, эксперименты

Объекты исследования. Как называются

- Генеральная совокупность

Совокупность всех объектов или их характеристик, по которым проводится анализ или исследование

- Выборка

Некоторая часть генеральной совокупности

Результаты исследования. Где хранятся

- Данные записываются и хранятся в

- векторах (одномерные)
- таблицах, матрицах (двумерные)
- массивах (многомерные)
- списках (сложные структуры)

Использование данных при проведении исследований

Пример: Исследование предпочтений покупателей торговой сети

1. Получение данных
2. Обработка данных
3. Анализ данных

Откуда получить исходные данные?

- Наблюдение
- Эксперимент

Наблюдение: способ получения данных, при котором воздействие наблюдателя на наблюдаемый объект сведено к минимуму

Эксперимент: на исследуемый объект оказывается заранее рассчитанное воздействие, затем выполняется наблюдение

Сколько данных требуется для достоверности выводов при проведении исследования?

Сколько объектов исследования необходимо для выполнения анализа?

1. Чем больше, тем лучше
2. 30

Получение данных

- Какие характеристики важны?
- Что учитываем?
- Что можно отбросить?

Генеральная совокупность и выборка

Генеральная совокупность (ГС): совокупность всех объектов, по которым будут сделаны выводы при проведении исследования.
Другой термин – популяция.

Как правило, ГС задается указанием нескольких условий, например

- Женщины + пенсионеры + пенсия > 10 тыс. руб + ходят в Пятерочку чаще 1 раза в неделю
- Взрослое население региона (область, город) + имеющее право голоса
- Семьи + наличие маленького ребенка + проживающие с родителями + доход выше 60 тыс на человека

В идеальном случае **нужно исследовать ВСЕ объекты ГС**
но ЭТО НЕРЕАЛЬНО: дорого, долго, физически невозможно и т.д.

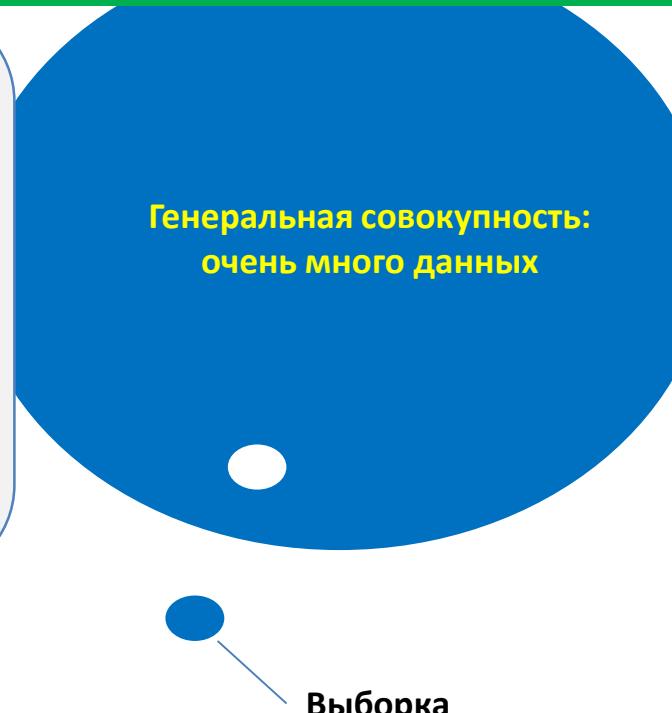
Пример: Прогноз результатов выборов в регионе

Поэтому

- исследуют лишь **часть объектов (создают выборку)**
- по результатам анализа выборки делаются выводы
- полученные выводы переносят на всю ГС

Но можно ли так делать?

- При выполнении определенных условий - МОЖНО**
- В разделе математики, занимающемся анализом выборок и условиями применимости выводов на ГС, разработана теория выборочных исследований. Занимается этим **математическая статистика**
- Выборка или выборочная совокупность** — часть генеральной совокупности элементов, которая охватывается экспериментом или наблюдением



Генеральная совокупность:
очень много данных

Выборка

Что такое правильная выборка?

Характеристики выборки

- **Способ формирования:** по каким правилам из ГС извлекаются элементы для выборки
 - Репрезентативность
 - Повторность
 - Рандомизация
- **Объем:** сколько элементов из ГС включается в выборку

Репрезентативность

- Репрезентативная выборка – выборка конечного объёма, обладающая всеми свойствами ГС, значимыми с точки зрения задач исследования
- Репрезентативность определяет, возможно ли обобщать результаты исследования, выполненные на выборке, на всю генеральную совокупность, из которой она была собрана
- Необходимым условием построения репрезентативной выборки является равная вероятность включения в нее каждого элемента генеральной совокупности

Повторность

- В выборке должно быть относительно много объектов из ГС
- Повторности должны быть независимы друг от друга. (Телефоны с конвейера должны быть из разных партий)

Рандомизация

- Каждый объект генеральной совокупности должен иметь равные шансы попасть в выборку

Что такое правильная выборка?

Объем

Выборки можно условно разделить на **большие и малые**.

В математической статистике используются различные методы анализа данных в зависимости от объема выборки.

Выборки объемом больше 30 элементов относят к большим

Примеры ошибок при работе с данными

1. Скорость зрительной реакции

Показывается предмет на доли секунды, затем спрашивается, что это было. Всего исследуется 10 человек, каждому показывается предмет пять раз. Авторы опыта считают, что у них 50 повторностей, однако на самом деле — только десять, так как каждый следующий показ не является независимым от предыдущего.

2. Случайный выбор 100 деревьев в лесу с целью измерения степени накопления тяжелых металлов в листьях. Внесение нового порядка в правило выбора деревьев.

3. Эффект поражения гусеницами на урожайность кукурузы.

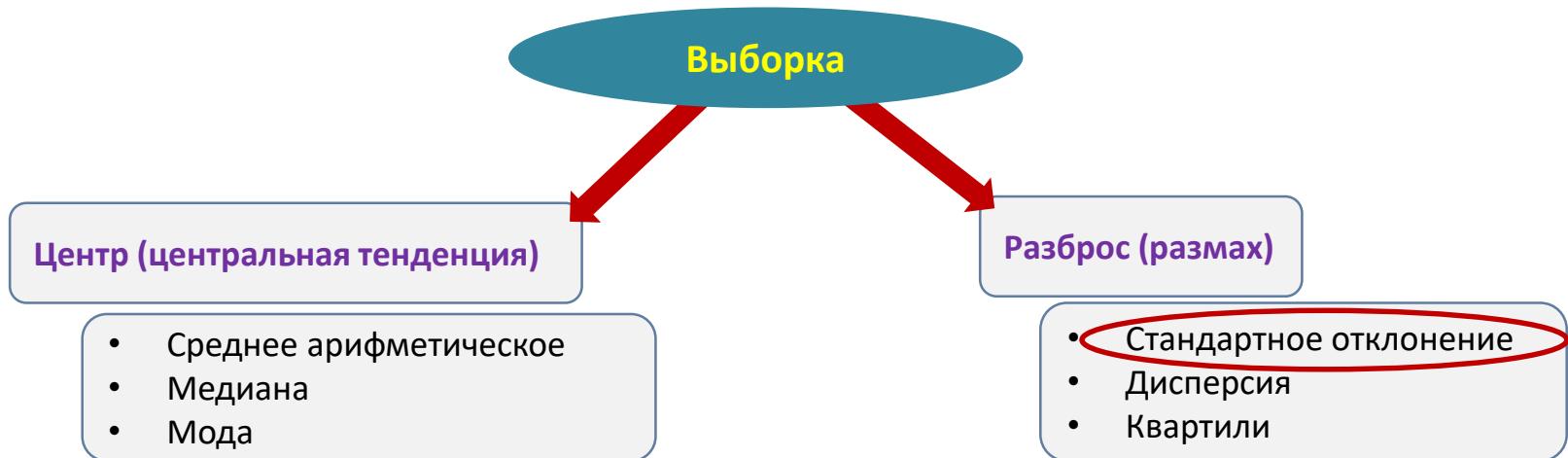
4. Исследование действия нового лекарства. Плацебо.

5. Влияние ядохимикатов на жуков-вредителей. Первый яд — самый действенный — ошибка! Жук, вылезающий первым, всегда более активный, быстрее набирает яд.

Вывод. Задача получения данных не тривиальна. В каких случаях исследованием данных вообще стоит заниматься?

- Когда результат неочевиден — выявление закономерностей и зависимостей. Звонки пот. клиентам.
- Когда результат кажется очевидным — проверка и подтверждение (опровержение) результата

Характеристики выборки



- Стандартное отклонение – характеристика величины разброса значений выборки
- Стандартное отклонение показывает, как в выборке распределены значения относительно среднего значения

Расчет стандартного отклонения

1. Находим среднее значение выборки
2. Определяем разность между каждым значением и средним
3. Возводим разность в квадрат
4. Суммируем квадраты разностей
5. Делим сумму квадратов на (количество элементов в выборке – 1)
6. Извлекаем корень

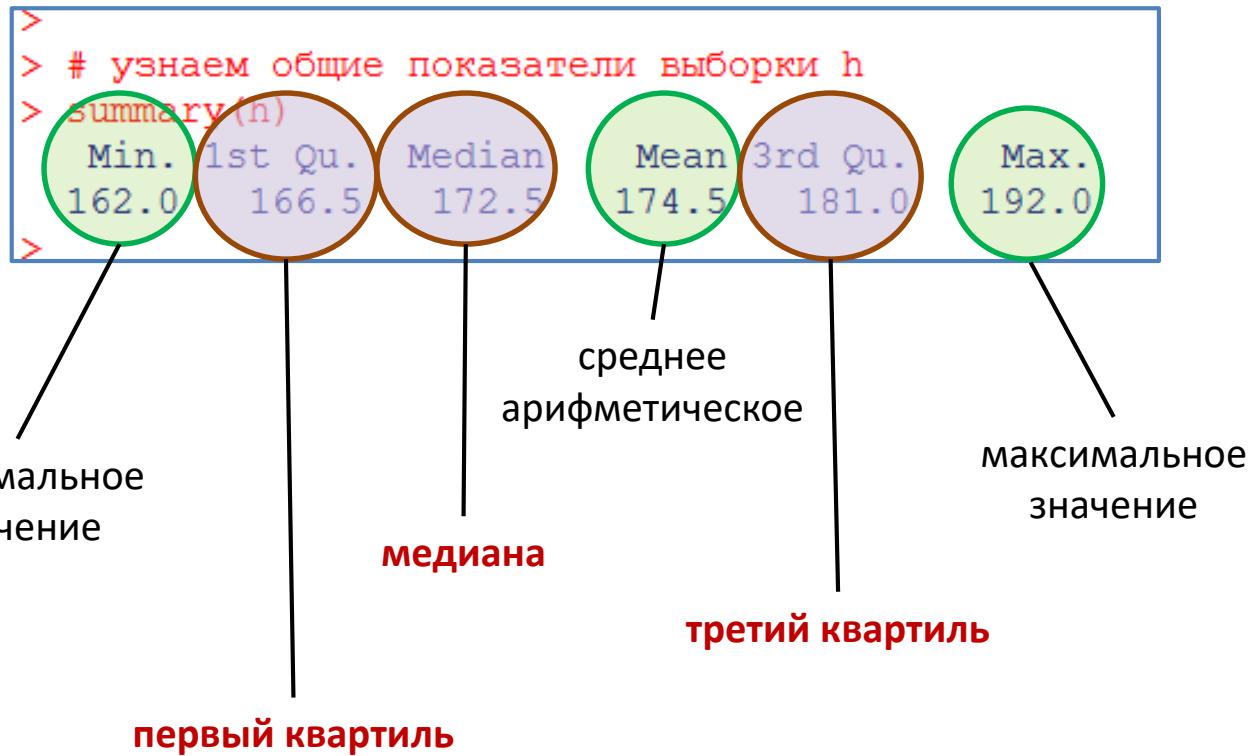
Показатели выборки – функция `summary()`

```
>  
> # регистрация роста сотрудников отдела  
>  
> h <- c(174, 162, 188, 192, 165, 168, 172.5)  
> h  
[1] 174.0 162.0 188.0 192.0 165.0 168.0 172.5  
>  
> # отсортируем выборку  
> sort(h)  
[1] 162.0 165.0 168.0 172.5 174.0 188.0 192.0  
>
```

Структура данных – функция `str()`

```
>  
> # узнаем структуру наших данных  
> str(h)  
num [1:7] 174 162 188 192 165 ...  
>  
. . .
```

Общие показатели выборки функция `summary()`



Примеры выборок с разными характеристиками

```
>  
> # зададим имена каждому значению выборки h  
> names(h) <- c("Николай", "Евгений", "Петр", "Александр", "Екатерина", "Василий", "Георгий")  
>  
> h  
 Николай    Евгений      Петр Александр Екатерина    Василий    Георгий  
    174.0      162.0      188.0      192.0      165.0      168.0      172.5  
>  
> |
```

Задание имен значениям выборки

Добавим зарплаты сотрудников

```
>  
> salary <- c(21, 19, 27, 11, 102, 25, 21)  
> names(salary)<-c("Грузчик", "Курьер", "Менеджер", "Уборщица", "Директор", "Бухгалтер", "Экспедитор")  
> salary  
 Грузчик    Курьер    Менеджер    Уборщица    Директор    Бухгалтер Экспедитор  
     21          19          27          11         102          25          21  
> |
```

Создание таблицы

```
>  
> # создадим таблицу, объединив рост сотрудников и их зарплату  
> company <-data.frame(h, names(salary), salary)  
>  
> names(company) <- c("Рост, см", "Должность", "Зарплата, тыс. руб")  
>  
> company  
           Рост, см   Должность Зарплата, тыс. руб  
 Николай      174.0   Грузчик            21  
 Евгений      162.0   Курьер            19  
 Петр          188.0   Менеджер          27  
 Александр     192.0   Уборщица          11  
 Екатерина    165.0   Директор          102  
 Василий      168.0   Бухгалтер          25  
 Георгий      172.5   Экспедитор        21  
>
```

Медиана и среднее арифметическое выборки

```
> summary(company)
```

Рост, см	Должность	Зарплата, тыс. руб
Min. :162.0	Бухгалтер :1	Min. : 11.00
1st Qu.:166.5	Грузчик :1	1st Qu.: 20.00
Median :172.5	Директор :1	Median : 21.00
Mean :174.5	Курьер :1	Mean : 32.29
3rd Qu.:181.0	Менеджер :1	3rd Qu.: 26.00
Max. :192.0	Уборщица :1	Max. :102.00
	Экспедитор:1	

Общие показатели по таблице

медиана и среднее
отличаются значительно

медиана и среднее близки

Сравнение выборок роста и зарплат

```
> sort(h)
Евгений Екатерина Василий Георгий Николай
    162.0      165.0     168.0    172.5     174.0
Пётр Александр
    188.0      192.0
```

```
>
> sort(salary)
Уборщица Курьер Грузчик Экспедитор Бухгалтер Менеджер Директор
    11        19        21        21        25        27      102
```

```
>
```

Расчет медианы. Медиана – центральная характеристика данных

Определение медианы

- Медиана – величина, находящаяся в центре ранжированной (отсортированной) выборки
- Медиана более устойчива (робастна) к выбросам и ошибочным данным по сравнению со средним арифметическим

Алгоритм вычисления медианы

- Выборка сортируется от меньшего значения к большему
- Пусть **N** – количество элементов выборки

Если **N** нечетное:

медиана = элементу с порядковым номером $= (N + 1) / 2$

Если **N** четное:

медиана = среднему арифметическому элементов с порядковыми номерами $(N / 2)$ и $(N / 2) + 1$

Свойства медианы

- медиана делит выборку пополам
- слева и справа от медианы находится одинаковое количество элементов

Пример. Рассчитать медиану для выборок

7, 3, 12, 21, 9, 5, 5, 17, 14
2, 7, 3, 9, 6, 7
16, 4, 17, 14, 11, 9, 30, 6

Расчет медианы в R: `median(числовой вектор)`

Центральные характеристики распределения данных. Мода

- Мода – значение в выборке, которое встречается чаще всего
- Мода, как правило, применяется для номинальных данных

Примеры номинальных данных

- Пол
- Цвет
- Ответы «Да» / «Нет»
- Наличие / Отсутствие
- Варианты выбора из заданного множества

```
>
> gender <- c("male", "female", "male", "male", "female", "male", "male")
> t.gender <- table(gender)
> t.gender
gender
female    male
      2       5
>
> mode <- t.gender[which.max(t.gender)]
> mode
male
  5
>
```

Расчет стандартного отклонения

Расчет стандартного отклонения

1. Находим среднее значение выборки
2. Определяем разность между каждым значением и средним
3. Возводим разность в квадрат
4. Суммируем квадраты разностей
5. Делим сумму квадратов на (количество элементов в выборке – 1)
6. Извлекаем корень

$$s = \sqrt{\frac{\sum_{i=1}^n (x_i - x_{\text{ср}})^2}{n - 1}}$$

```
>  
> nh <- length(h)  
> mean.h <- sum(h) / nh  
> s.h <- sum((h - mean.h)^2)  
> sd.h <- sqrt(s.h / (nh-1))  
> mean.h  
[1] 174.5  
> s.h  
[1] 781.5  
> sd.h  
[1] 11.41271  
>
```

```
x<-sample(15)  
x  
sd(x)
```

```
>  
> sd(h)  
[1] 11.41271  
>  
>
```

Расчет стандартного отклонения в R: `sd(числовой вектор)`

Стандартное отклонение. Пример использования

Остатки по неделям, склад № 1

Среднее						
33	31	32	36	31	31	32,3

Остатки по неделям, склад № 2

22	34	58	52	10	21	32,8
----	----	----	----	----	----	------

Склад № 1

Среднее						
33	31	32	36	31	31	32,3

2,0

Склад № 2

22	34	58	52	10	21	32,8
----	----	----	----	----	----	------

18,9

работа менеджмента на складе № 2 значительно хуже, чем на складе № 1;
склад № 2 «лихорадит»

Расчет в R:

Дисперсия?

d – дисперсия

sd – стандартное отклонение

d = sd * sd

Расчет в R: var(вектор)

```
> store1 = c(33, 31, 32, 36, 31, 31)
> store2 = c(22, 34, 58, 52, 10, 21)
>
> mean(store1)
[1] 32.33333
> mean(store2)
[1] 32.83333
>
> median(store1)
[1] 31.5
> median(store2)
[1] 28
>
> sd(store1)
[1] 1.966384
> sd(store2)
[1] 18.87238
```

Характеристики разброса данных: квартили, процентили, IQR

Определение квартиля

Квартили (кварта, четверть) – значения, которые делят отсортированную выборку на четыре группы, содержащие приблизительно равное количество наблюдений.

Общий объем выборки делится на четыре равные части: 25%, 50%, 75% 100%.

- Первый (нижний) квартиль **Q1** отсекает слева 25 % выборки
- Второй (средний) квартиль **Q2** отсекает слева 50 % выборки
- Третий (верхний) квартиль **Q3** отсекает слева 75 % выборки

Определение процентиля

n-й процентиль - это значение, ниже которого расположено **n** процентов элементов выборки

Первый процентиль – значение, ниже которого располагается **1** процент элементов выборки

25-й процентиль совпадает с первым квартileм **Q1**

90-й процентиль – значение, ниже которого расположено **90** процентов всей выборки

Определение IQR

Интерквартильный размах (интервал, отрезок) **InterQuartile Range**

$$\text{IQR: } \text{IQR} = Q_3 - Q_1$$

IQR – интервал, содержащий центральные 50% наблюдений выборки, т.е. интервал между 25-м и 75-м процентилями

Расчет квартилей: два способа

1 способ. Простой, грубый

1. Сортируем выборку от меньших значений к большим
2. Находим медиану
3. Рассматриваем левую половину выборки. Находим медиану, это **Q1**
4. Рассматриваем правую половину выборки. Находим медиану, это **Q3**

Пример 1. $y <- c(1, 7, 4, 3)$

Отсортированная выборка = (1, 3, 4, 7)

$$\text{Медиана} = (3 + 4) / 2 = 3.5$$

$$\text{Первый quartиль Q1} = \text{медиана выборки } (1, 3) = (1 + 3) / 2 = 2$$

$$\text{Третий quartиль Q3} = \text{медиана выборки } (4, 7) = (4 + 7) / 2 = 5.5$$

Расчет в R

```
>  
> y<-c(1,7,4,3)  
> summary(y)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1.00	2.50	3.50	3.75	4.75	7.00

Значения quartилей не совпадают

Точный алгоритм расчета квартилей

1. Сортируем выборку от меньших значений к большим
2. Каждому значению выборки сопоставляем действительное число f в диапазоне от 0 до 1 по формуле $f(i) = (i - 1) / (N - 1)$, где
 - i – номер элемента в выборке
 - N – общее число элементов выборки
3. Первый quartиль **Q1**, соответствующий $f = 0.25$, вычисляется по двум соседним f -значениям, находящимся ниже и выше 0.25
4. Второй quartиль **Q2**, соответствующий $f = 0.5$ вычисляется по двум соседним f -значениям, находящимся ниже и выше 0.5
5. Третий quartиль **Q3**, соответствующий $f = 0.75$ вычисляется по двум соседним f -значениям, находящимся ниже и выше 0.75

Любой процентиль рассчитывается аналогично

Пример 2. Даны выборка

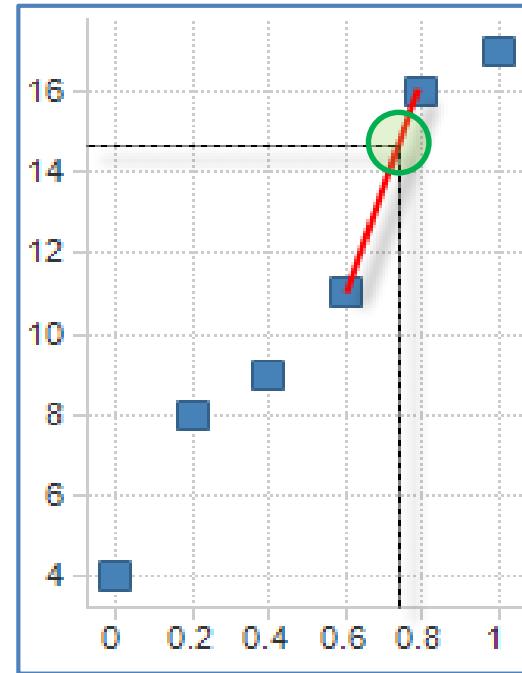
11, 17, 8, 16, 4, 9

- Рассчитать таблицу f -значений
- Найти $Q1$ и $Q3$
- Определить IQR

Решение примера 2

Выборка	f-значение
4	0
8	0,2
9	0,4
11	0,6
16	0,8
17	1

Поиск Q3



Для нахождения точки пересечения отрезка с координатами $[(x = 0,6; y = 11), (x=0,8; y = 16)]$ с прямой $x=0,75$ используем формулу линейной интерполяции

$$y = Y_1 + \frac{Y_2 - Y_1}{X_2 - X_1} (x - X_1)$$

X1	0,6
X2	0,8

Y1	11
Y2	16

Ответ

$$\begin{aligned}Q3 &= 14.75 \\Q1 &= 8.25 \\IQR &= 6.5\end{aligned}$$

```
>  
> z<- c(11, 17, 8, 16, 4, 9)  
> summary(z)  
Min. 1st Qu. Median Mean 3rd Qu. Max.  
4.00 8.25 10.00 10.83 14.75 17.00  
>
```

Пример 3

Пример. Даны выборка

30, 44, 4, 1, 52, 17, 19, 35, 41, 8, 11

- Рассчитать таблицу f-значений
- Найти Q1, Q2, Q3
- Определить IQR

Выборка	1	4	8	11	17	19	30	35	41	44	52
f-значение	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1

$$y = Y_1 + \frac{Y_2 - Y_1}{X_2 - X_1} (x - X_1)$$

Ответы

$$Q1 = 9.5$$

$$Q2 = 19$$

$$Q3 = 38$$

$$IQR = 28.5$$

Правильное решение примера 1

Пример 1. Даны выборка

1, 3, 4, 7

- Рассчитать таблицу f-значений
- Найти Q1, Q2, Q3
- Определить IQR

Выборка (Y)	f-значение (X)
1	0
3	1/3
4	2/3
7	1

Решение

$$Q1 = 1 + (3 - 1) / (1/3 - 0) * (0.25 - 0) = 2.5$$

$$Q2 = 3 + 1 / (1/3) * (0.5 - 1/3) = 3 + 3/2 - 1 = 3.5$$

$$Q3 = 4 + (7 - 4) / (1 - 2/3) * (0.75 - 2/3) = 4 + 27/4 - 6 = (27 - 8) / 4 = 4.75$$

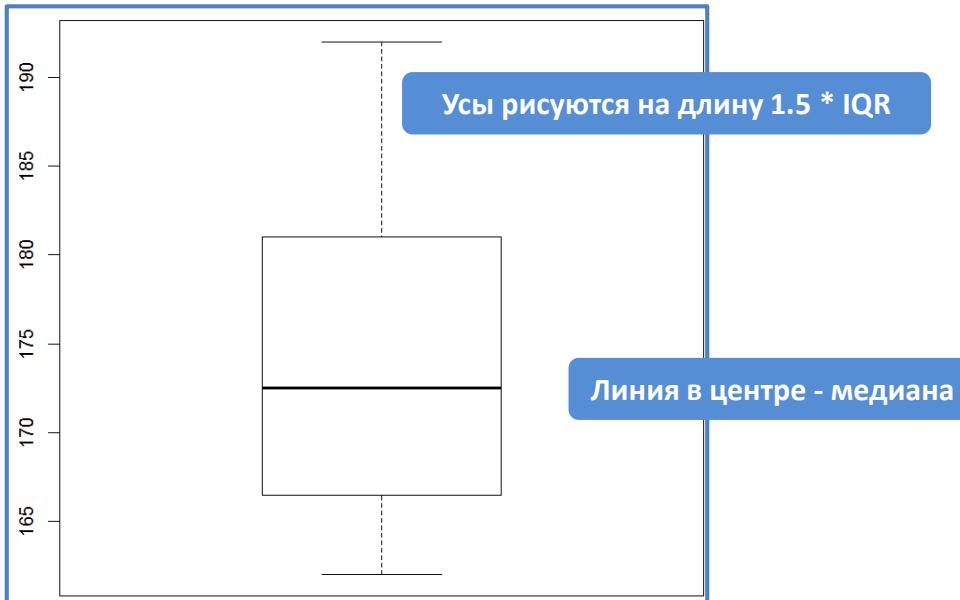
```
>  
> y<-c(1,7,4,3)  
> summary(y)  
Min. 1st Qu. Median Mean 3rd Qu. Max.  
1.00 2.50 3.50 3.75 4.75 7.00  
>
```

Графическое представление центральной характеристики и разброса

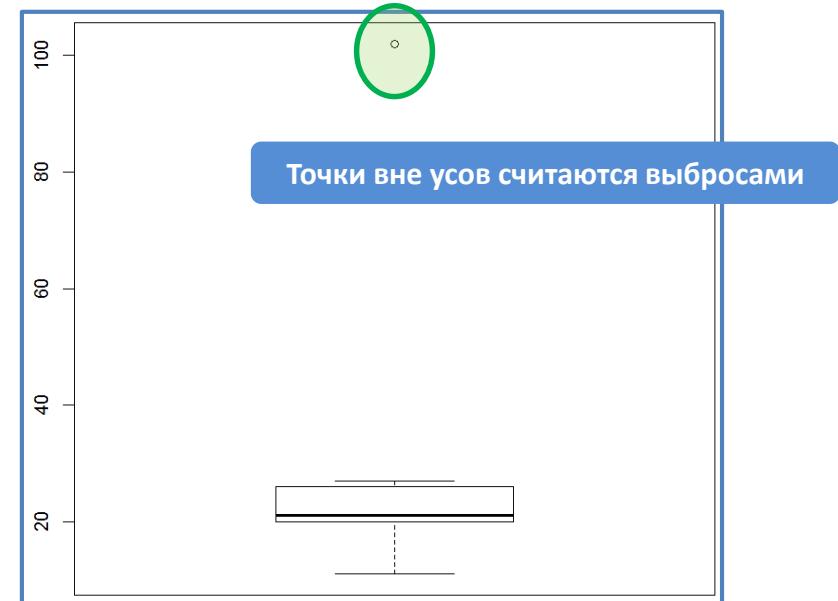
Характеристики выборок роста и зарплат

```
>  
> summary(h)  
  Min. 1st Qu. Median     Mean 3rd Qu.     Max.  
 162.0   166.5  172.5   174.5  181.0   192.0  
>  
> summary(salary)  
  Min. 1st Qu. Median     Mean 3rd Qu.     Max.  
 11.00   20.00  21.00   32.29  26.00  102.00  
>
```

Ящик с усами (боксплот) для выборки h



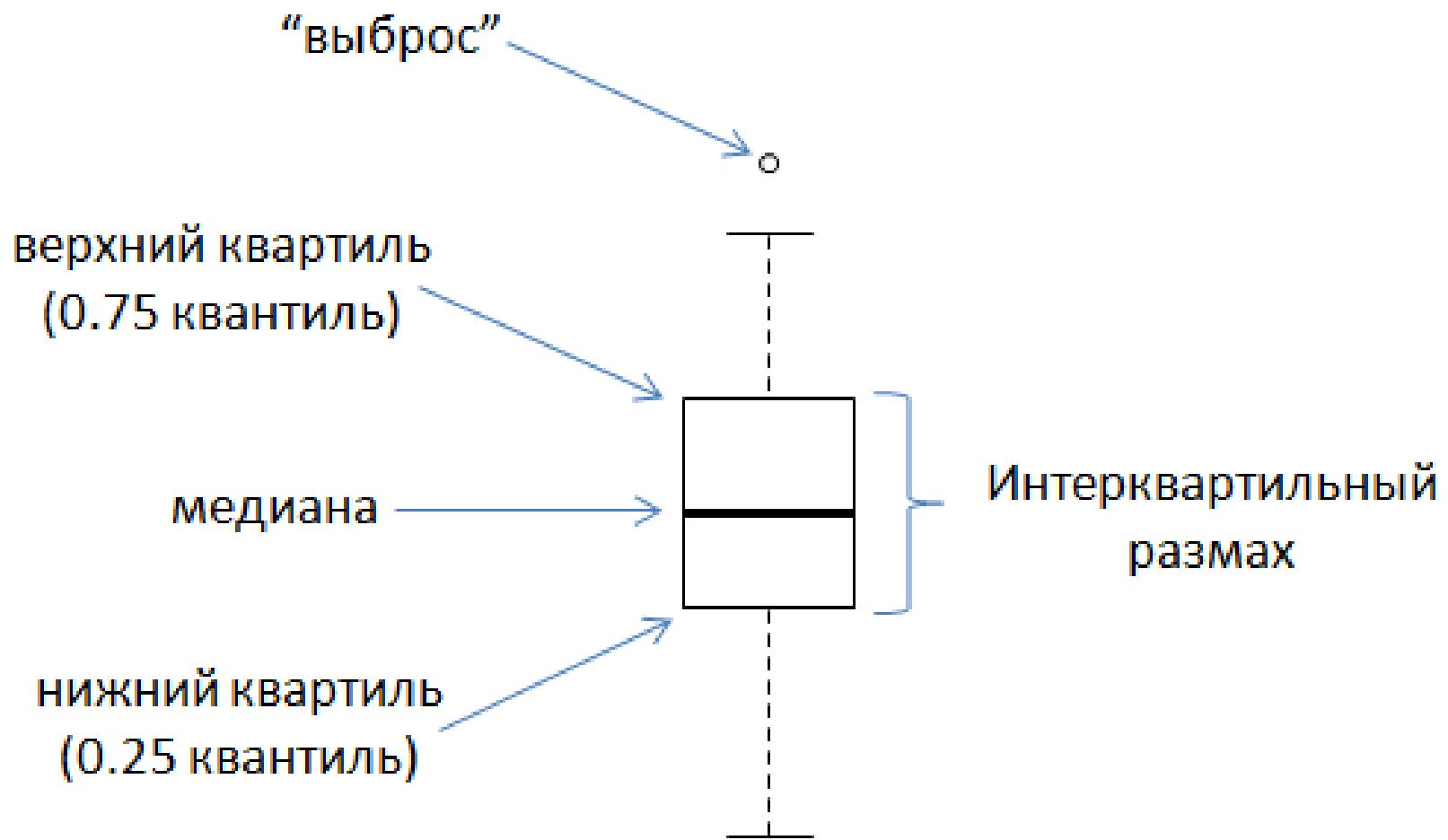
Ящик с усами (боксплот) для выборки salary



Высота ящика = IQR

Функция `boxplot()` – диаграмма размаха

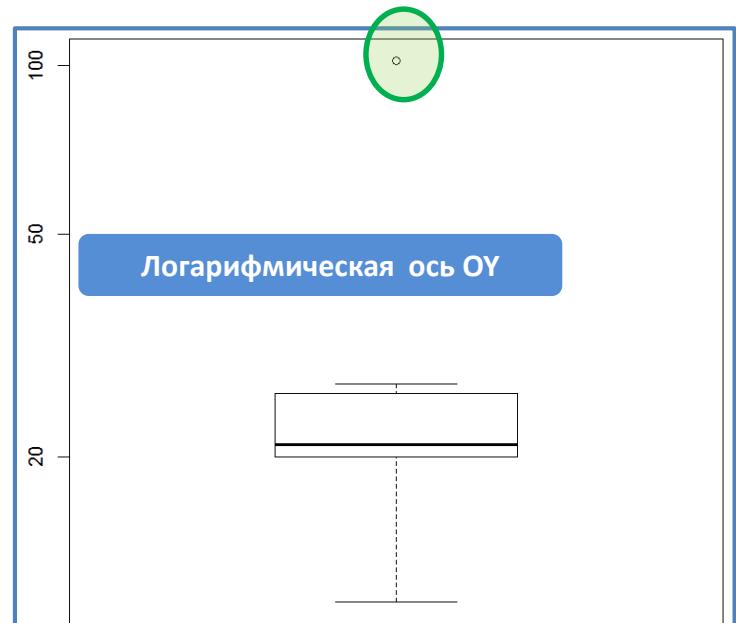
Диаграмма размаха (ящик с усами, изобретен 50 лет назад) – график, использующийся в описательной статистике, отображающий одномерное распределение вероятностей некоторой выборки данных



Команды для рисования боксплотов

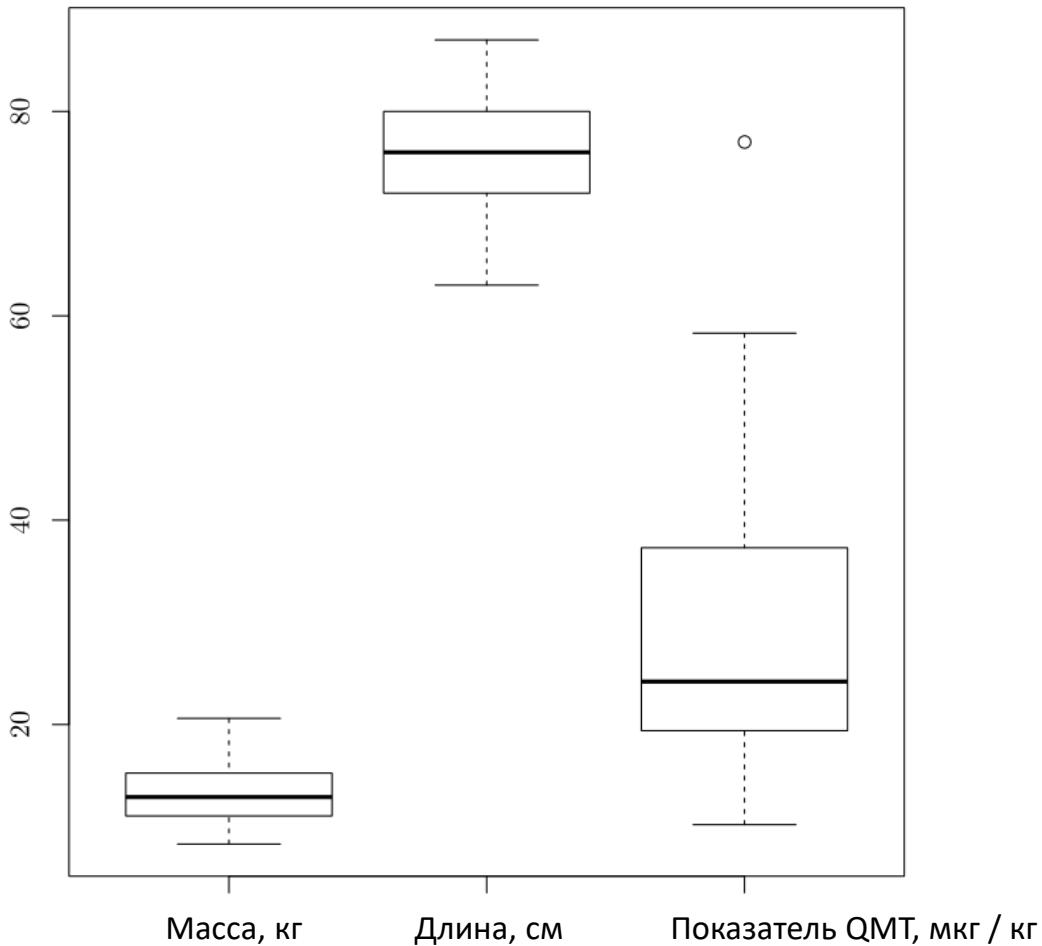
```
>  
> boxplot(h)  
>  
> boxplot(salary)  
> boxplot(salary, log="y")  
>
```

Ящики с усами (боксплоты) для выборки salary



Варианты размещения боксплотов

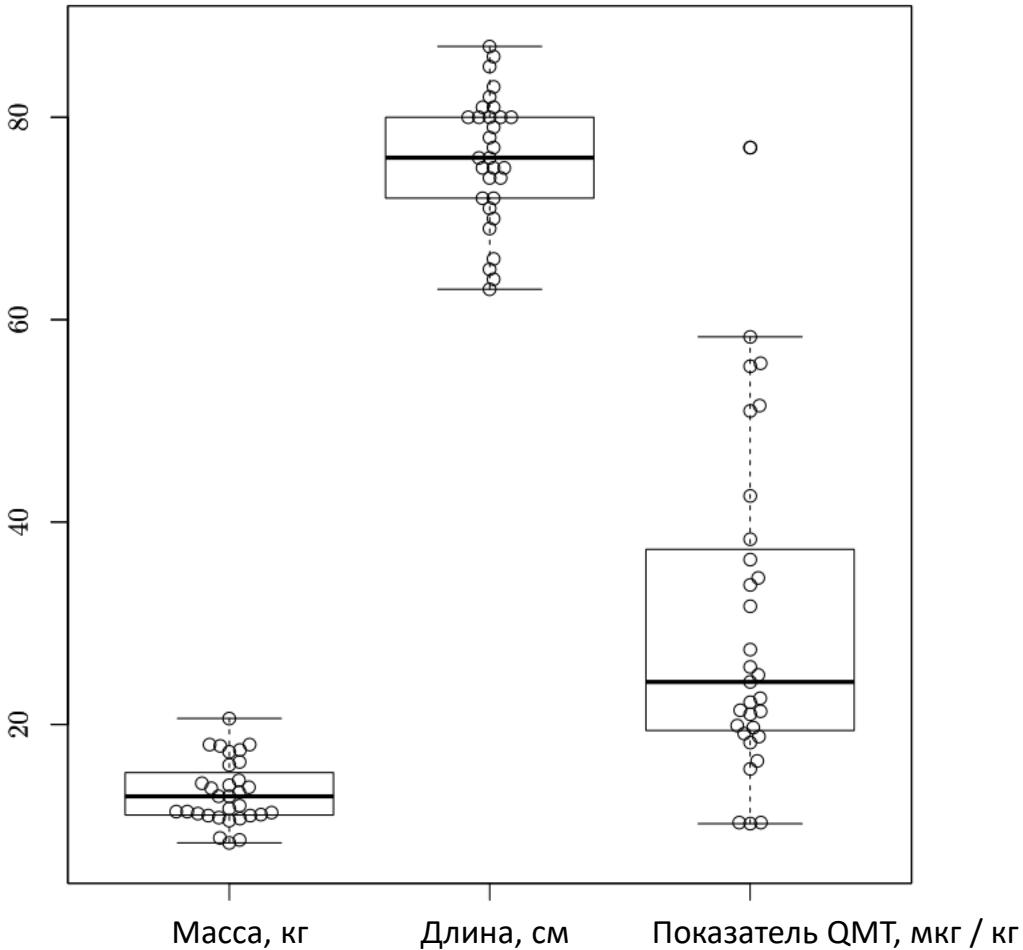
Представление данных, имеющих разный масштаб и ед. измерения



Результаты исследования диких обезьян, пораженных вирусом Quarella Mourus
атолл Баа, 2018

Варианты размещения боксплотов

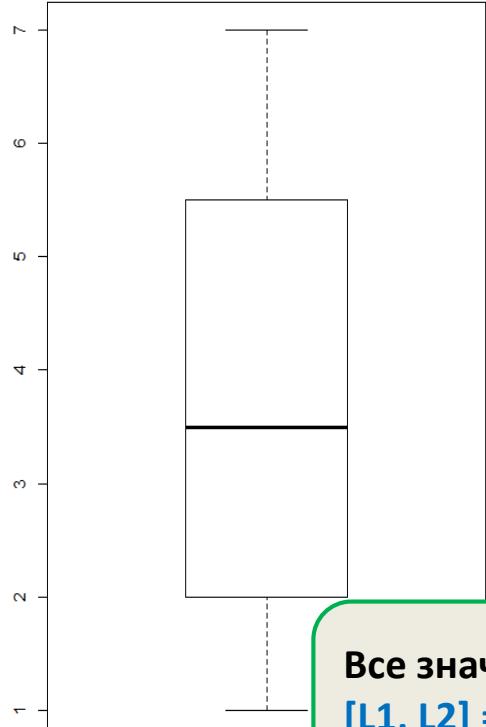
Бокспоты -ульи



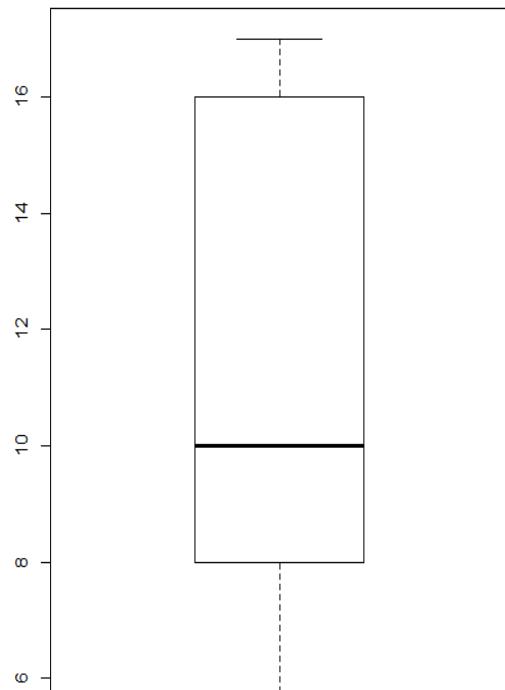
Результаты исследования диких обезьян, пораженных вирусом Quarella Mourus
атолл Баа, 2018

Рисуем усы у boxplot

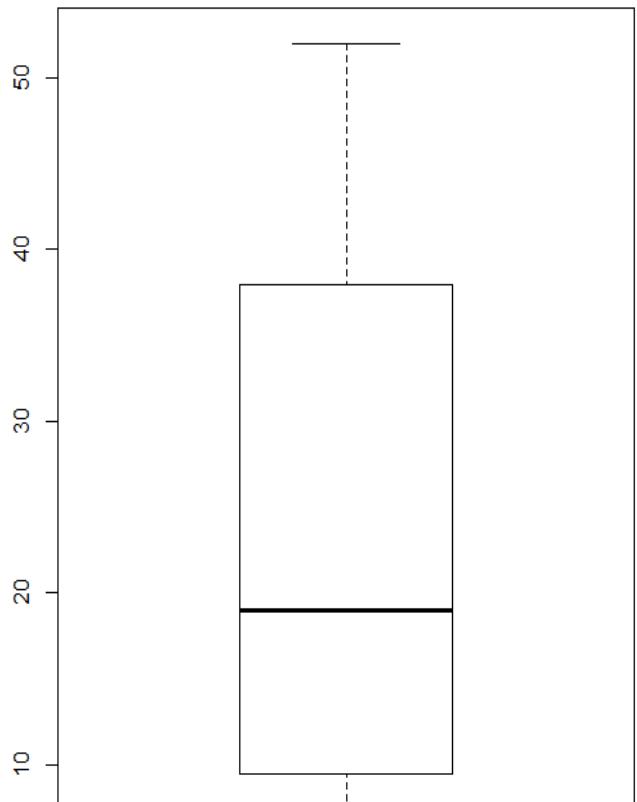
```
y<-c(1,7,4,3)  
boxplot(y)
```



```
y<-c(11, 17, 8, 16, 4, 9)  
boxplot(y)
```



```
y<-c(30, 44, 4, 1, 52, 17, 19, 35, 41, 8, 11)  
boxplot(y)
```

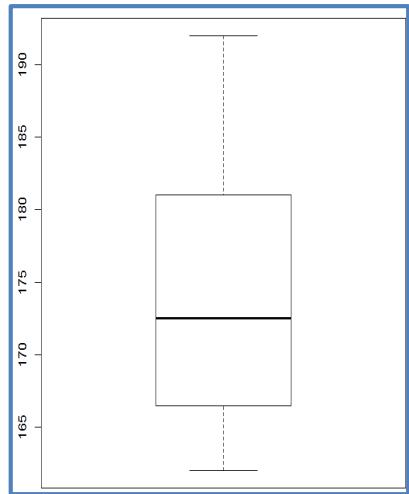


Все значения выборки находятся внутри диапазона [L1, L2]

$$[L1, L2] = Q3 + IQR * 1.5 - (Q1 - IQR * 1.5)$$

$$[L1, L2] = Q3 - Q1 + IQR * 3 = 4 * IQR$$

Начинаем рисовать boxplot



$Q3 = 4.75$

Медиана = 3.5

$Q1 = 2.5$

$IQR = 4.75 - 2.5 = 2.25$

Вопросы

Что такое усы у boxplot ?

В каких случаях усы есть, в каких случаях их нет?

Как их рисовать?

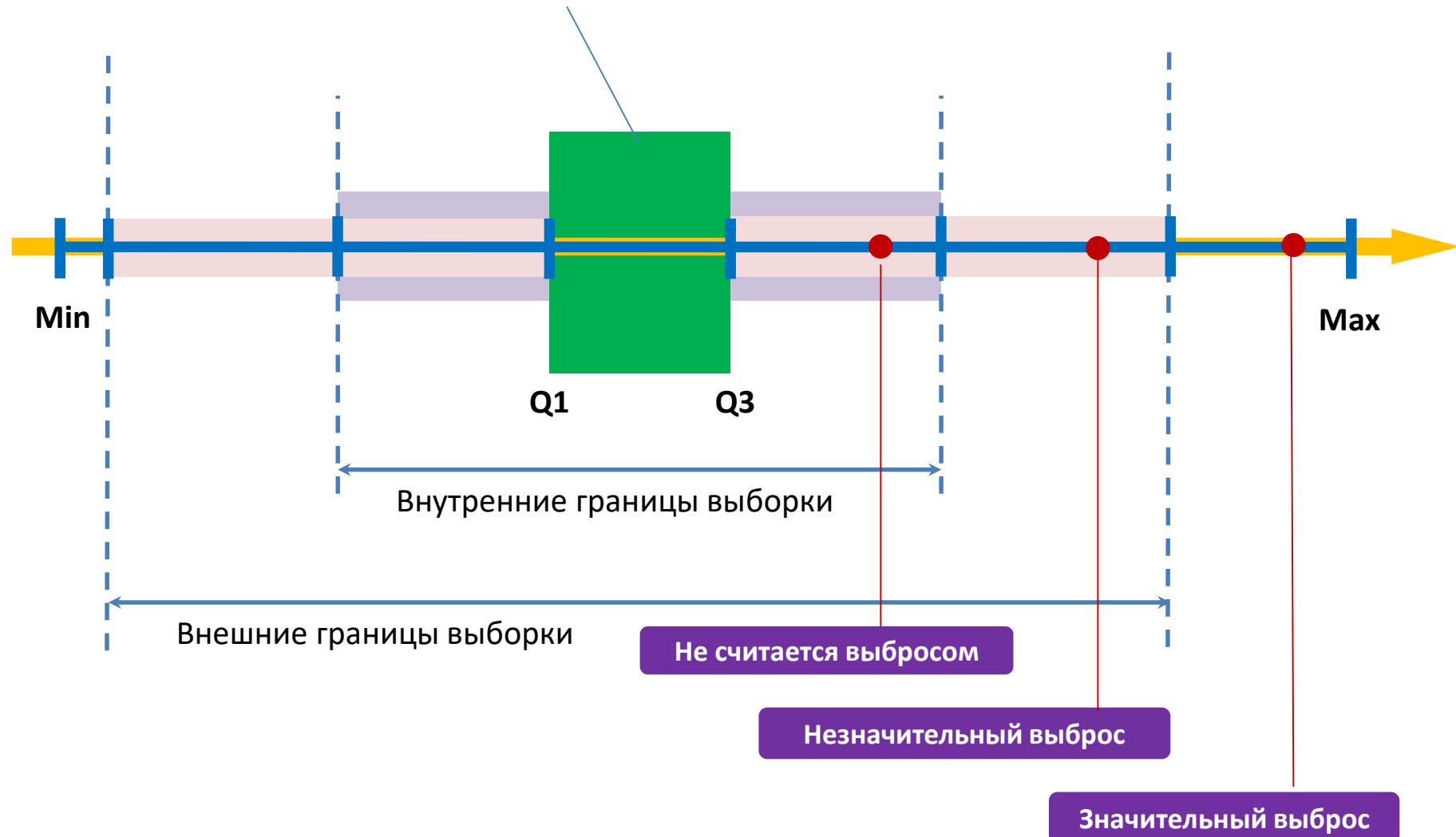
Что такое выбросы?

Как их рисовать?

Внутренние и внешние границы выборки

Диапазон значений выборки

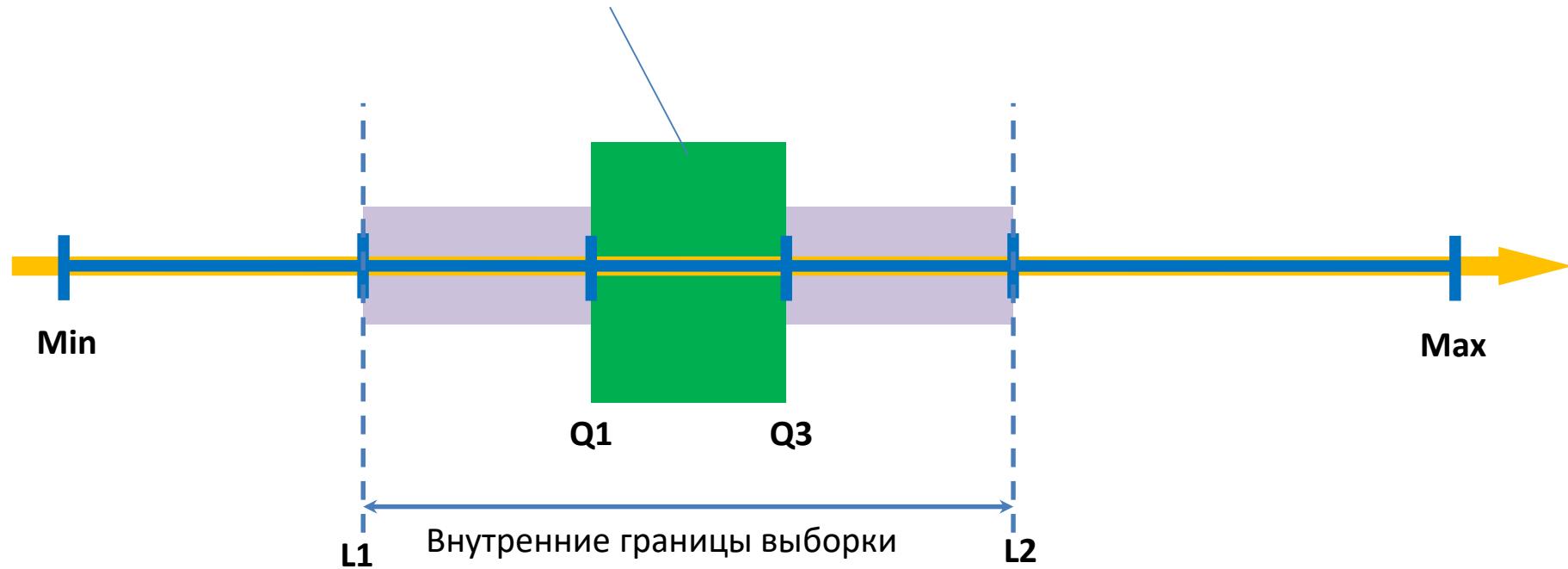
В полосу шириной IQR попадает 50 % всех значений выборки



Внутренние границы выборки – усы у boxplot

Диапазон значений выборки

В полосу шириной IQR попадает 50 % всех значений выборки



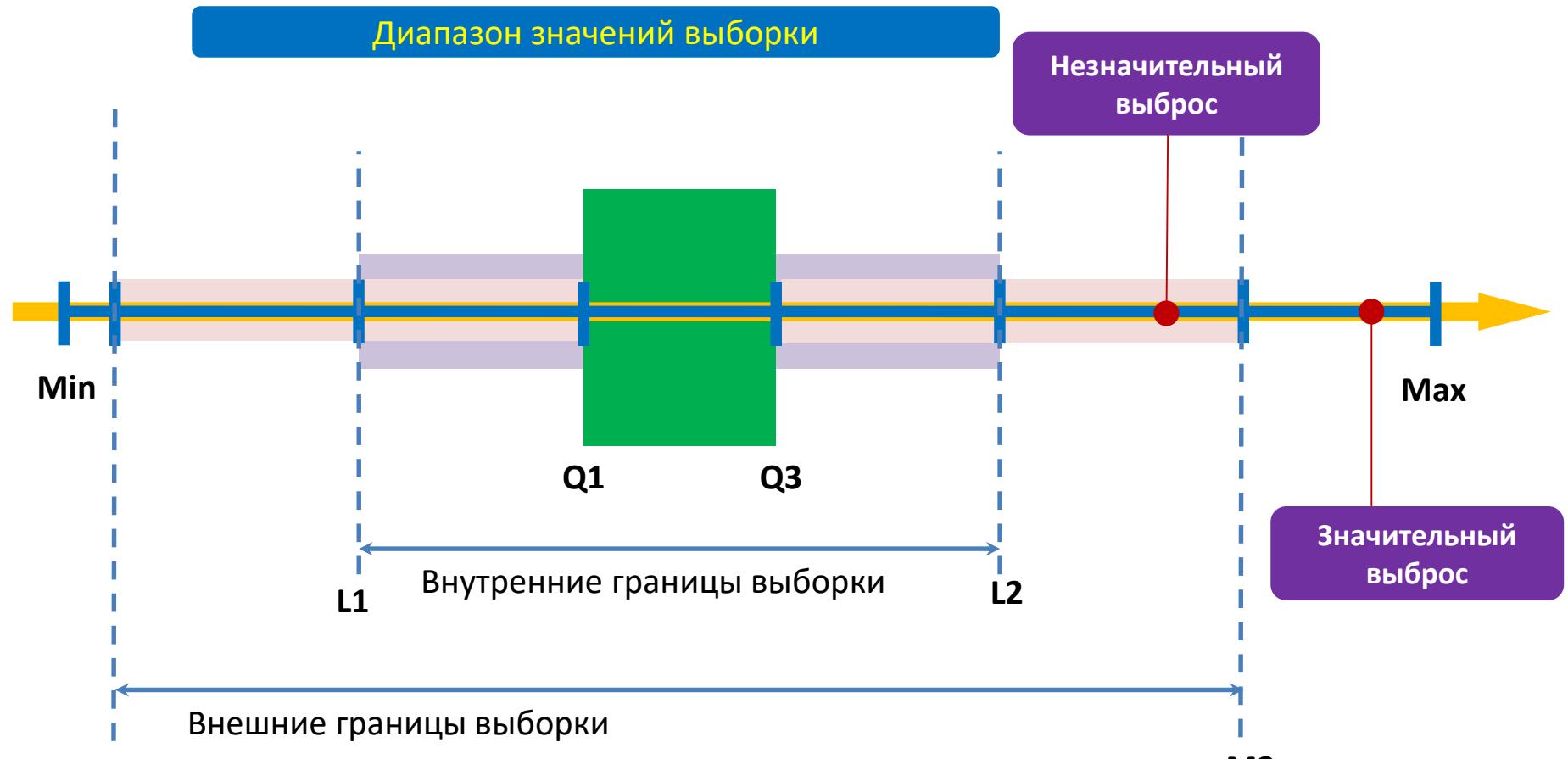
$$\text{Ширина прямоугольника} = \text{IQR} * 1.5$$

$$L1 = Q1 - \text{IQR} * 1.5$$

$$L2 = Q3 + \text{IQR} * 1.5$$

Все значения из диапазонов [L1, Q1], [Q3, L2] рисуются усами

Внешние границы выборки. Выбросы



$$\text{Ширина прямоугольника} = \text{IQR} * 3$$

$$M1 = Q1 - \text{IQR} * 3$$

$$M2 = Q3 + \text{IQR} * 3$$

Все значения из диапазонов [Min, L1], [L2, Max] рисуются точками

Природа выбросов

Что делать с выбросами?

Исключить выбросы из выборки Не исключать выбросы из выборки

Выполнить анализ

- Ошибки ввода
- Ошибки оборудования

История о летающих рыбах

- Выполнить углубленный анализ
- Выяснить природу выбросов

Сигнал о новых, ранее не известных науке зависимостях и фактах

- Уточнение зависимостей
- Новые закономерности и понимания
- Открытия
- Раскрытие преступлений

Принятие решения о покупке или продаже акции. Функция `vote()`

Функция `vote()`

- Решение о покупке или продаже акции принимается инвестором на основании результата работы функции `vote()`, которая для акций разных компаний ($i=1, \dots, N$) с разной вероятностью возвращает одно из двух значений:
- **1 (покупать): позитивный** сценарий
Инвестор ожидает, что стоимость акций будет **расти**
- **-1 (продавать): негативный** сценарий
Инвестор ожидает, что стоимость акций будет **падать**
- Поведение функции `vote()` определяется управляющим параметром `tone`, от значения которого зависит результат, возвращаемый `vote()`
- Параметр `tone` отражает настроение инвестора
- Параметр `tone` в общем случае имеет **не числовую природу**

Реализация алгоритма торговли

Используются функции **price.stock()**, **vote()**, **quant()**

В цикле от $t = 1$ до T выполняем действия:

- **Шаг 1.** Принимается решение о покупке или продаже акции
Задается настроение торгов **tone**, вызывается функция **vote(tone)**
- **Шаг 2.** Рассчитывается текущая цена акции.
Вызывается функция **price.stock()**
- **Шаг 3.** Определяется количество акций для покупки или продажи. Вызывается функция **quant()**
- **Шаг 4.** Пересчитывается количество текущих денежных средств и акций инвестора

После завершения цикла все акции продаем по цене последнего дня

Считаем выручку $Q(T)$ и прибыль $Pr = Q(T) - Q(1)$

Параметры функции quant()

Перечень параметров функции **quant()**

- **nAll** – количество акций компании, имеющихся в наличии у инвестора до сделки
- **nMax** – максимально возможное количество акций для покупки или продажи за одну сделку
- **nMin** – минимально возможное количество акций для покупки или продажи
- **tSaleBuy** – тип сделки; **tSaleBuy = -1** означает продажу, **tSaleBuy = 1** означает покупку.
- В зависимости от значения **tSaleBuy** функция **quant()** возвращает целое случайное число в следующих диапазонах:

если **tSaleBuy = -1**, диапазон возможных значений **quant()** равен:

$$[\min(nMin, nAll), \min(nMax, nAll)]$$

если **tSaleBuy = 1**, диапазон возможных значений **quant()** равен:

$$[nMin, nMax]$$

Управляющий параметр `tone`

Характеристики параметра `tone`:

- `tone` является целым числом
- `tone` принимает значения в диапазоне от **0** до **100**
- `tone = 0` отражает **крайне негативный сценарий**, при котором функция `vote(tone=0)` всегда возвращает **-1**
- `tone = 100` отражает **крайне оптимистический сценарий**, при котором `vote(tone=100)` всегда возвращает **1**
- при `tone = 50` функция `vote(tone=50)` возвращает **1** или **-1** с равными вероятностями

Лукьянов Павел Борисович
профессор департамента Анализа данных, принятия решений
и финансовых технологий

Программирование в среде R

Лекция 8

Стратегии для игры на бирже.
Метод скользящей средней

Финансовый университет, 2020

Получение баллов за работу в мае-июне

	Показатели для получения баллов за май-июнь	Баллы	Макс
1	Посещение (присутствие) на семинарах и лекциях	0-1	1
2	Регулярная отправка работ после завершения семинара	0-1	1
3	Регулярная отправка выполненных заданий до следующего семинара (все задания из методички)	0-1	1
4	Торговая стратегия. Оформление отчета, графиков, скриптов, комментарии	0-3	3
5	Торговая стратегия. Реализация скользящей средней	0-2	2
6	Торговая стратегия. Оригинальность кода (не скопирован у товарища)	-5 - 2	2
			10

ZIP-архив с фамилией студента и группой

Например: ИвановПМ19-2.zip

1. Файлы со скриптами

- библиотека **trade.lib.R** со всеми функциями
- Основной скрипт **trader.R**, в нем

генерация данных, графики, торговый алгоритм, расчеты, формирование итоговой таблицы, комментарии по алгоритму

2. Файлы с графиками

Шесть графиков по торговле акциями с отмеченными точками покупки-продажи

3. Файл с итоговой таблицей с результатами торгов

в формате csv или Excel

План выполнения самостоятельной работы «Стратегия торговли на бирже»

1. Генерация цен для шести акций (тикеров)

Различные параметры при использовании `generate.price()`

Шесть графиков цен, разная волатильность, разное кол-во волн, рост / снижение

2. Последовательное тестирование торговой стратегии на полученных шести наборах цен. Регистрация точек покупки (зеленая) / продажи (красная) на графиках цен

3. Расчет прибыли, рентабельности по окончании периода торгов по каждой акции

4. Заполнение итоговой таблицы

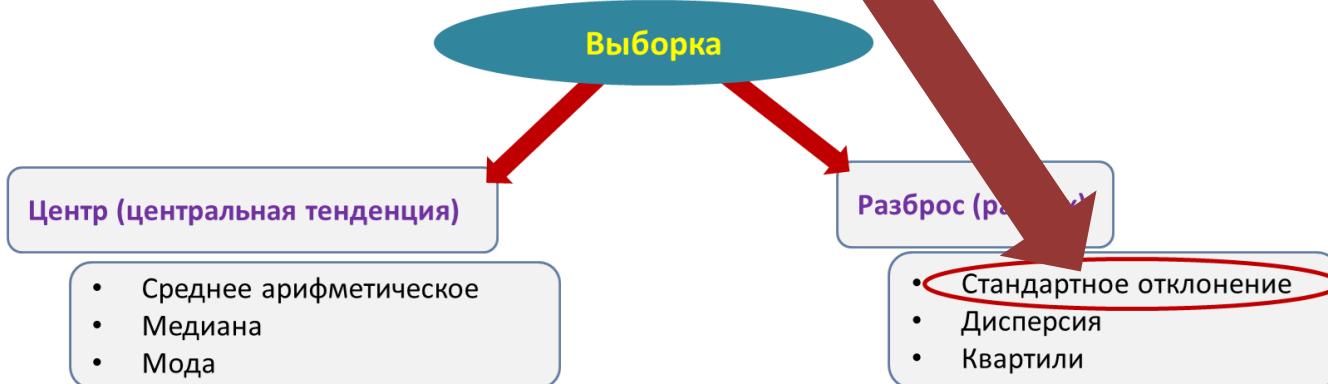
№	Тикер	sd.norm	Прибыль	Рентабельность	Особенности стратегии
1					
2					
3					
4					
5					
6					

Расчет значения волатильности акции в течение некоторого периода

1. Рассматривается **выборка** из **генеральной совокупности**:

значения суточных цен акции за определенный период – вектор **price**

2. Рассматривается характеристика акции: **волатильность** – мера изменчивости финансового актива: как сильно меняются цены в течение заданного отрезка времени



- Стандартное отклонение – характеристика величины разброса значений выборки
- Стандартное отклонение показывает, как в выборке распределены значения относительно среднего значения

Расчет стандартного отклонения

1. Находим среднее значение выборки
2. Определяем разность между каждым значением и средним
3. Возводим разность в квадрат
4. Суммируем квадраты разностей
5. Делим сумму квадратов на (количество элементов в выборке – 1)
6. Извлекаем корень

Расчет нормированного значения волатильности акции

№	Тикер	sd.norm	Прибыль	Рентабельность	Особенности стратегии
1					
2					
3					
4					
5					
6					

$$s = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}}$$

Недостаточно знать **абсолютное значение волатильности $sd(price)$**

необходимо соотнести это значение со **средней ценой акции $mean(price)$**

sd.norm – нормированное значение волатильности

$$sd.norm = sd(price) / mean(price)$$

Примерная реализация алгоритма торговли

Используются функции из библиотеки **trade.lib.R**

В течение дня инвестор по каждой акции может совершать **только одну сделку**.

В первый день торгов ($t=1$) у инвестора имеются денежные средства в размере **Q** руб

Выбрать акцию. В цикле от $t = 1$ до $T=100$ выполнять следующие действия:

- **Шаг 1.** Принять решение о торговле (торговать / не торговать)
- **Шаг 2.** Если принято решение о торговле, принять решение о покупке или продаже акций
- **Шаг 3.** Принять решение о количестве акций для покупки / продажи
- **Шаг 4.** Выполнить покупку / продажу, пересчитать денежные средства, количество акций

После завершения цикла все акции продать по цене последнего дня

Подсчитать прибыль $Pr = Q(T) - Q(t=1)$
рентабельность $R = Pr / Q(t=1) * 100 \%$

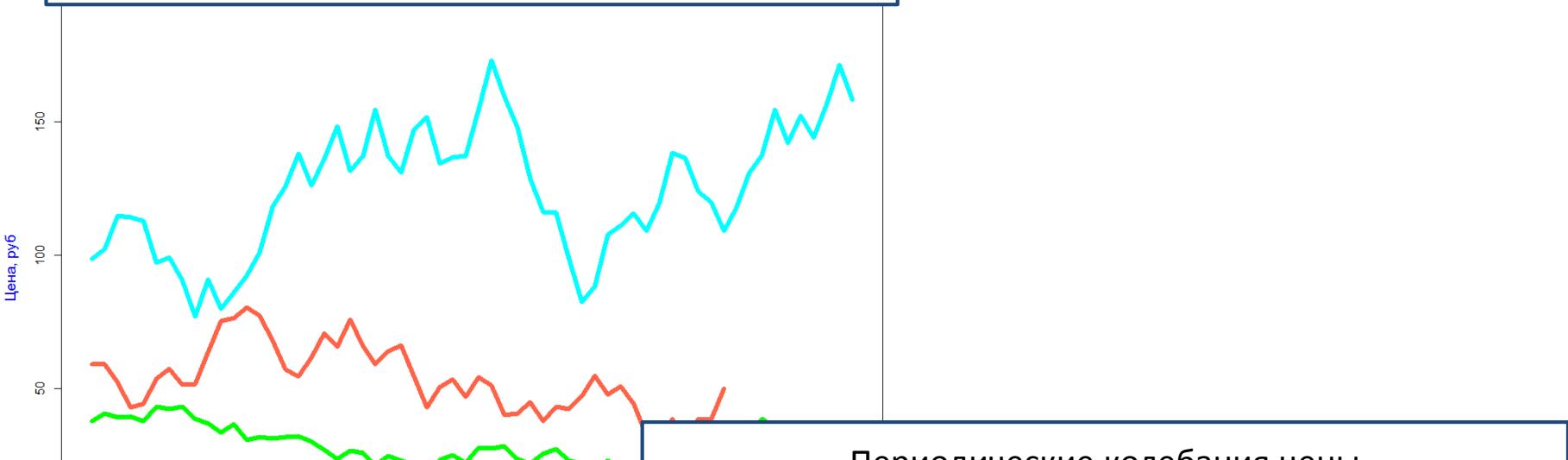
Метод скользящей средней

	Показатели для получения баллов за май-июнь	Баллы	Макс
1	Посещение (присутствие) на семинарах и лекциях	0-1	1
2	Регулярная отправка работ после завершения семинара	0-1	1
3	Регулярная отправка выполненных заданий до следующего семинара (все задания из методички)	0-1	1
4	Торговая стратегия. Оформление отчета, графиков, скриптов, комментарии	0-3	3
5	Торговая стратегия. Реализация скользящей средней	0-2	2
6	Торговая стратегия. Оригинальность кода (не скопирован у товарища)	-5 - 2	2
			10

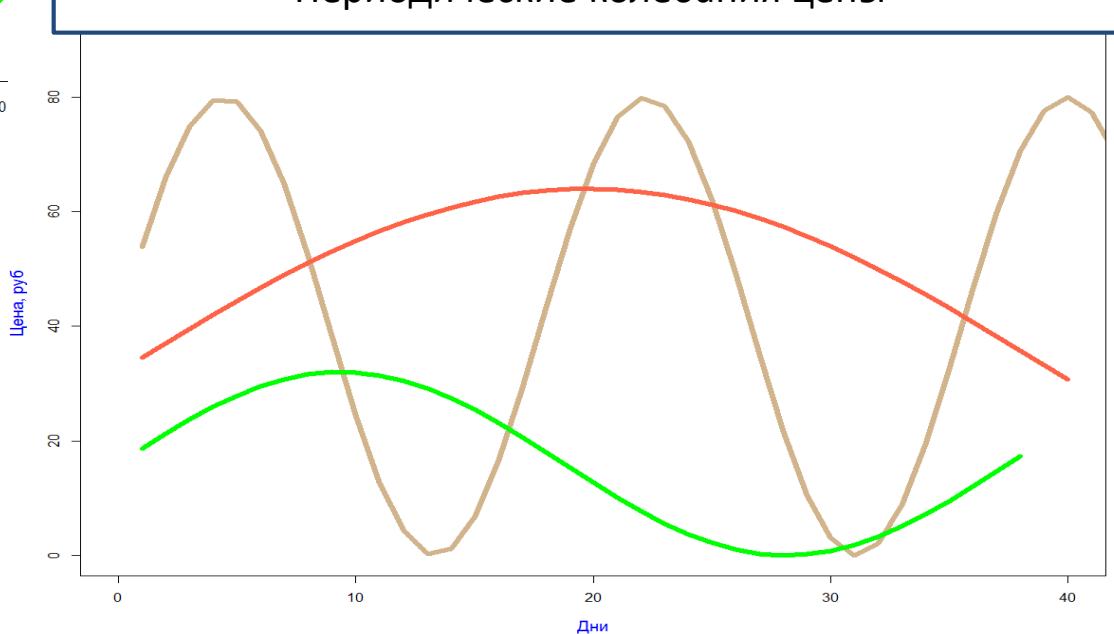
Метод скользящей средней – один из базовых прогнозных инструментов для построения стратегии торговли (**индикатор Moving Average (MA)**)

Составляющие динамики цены акции

Случайные колебания цены



Периодические колебания цены



Метод скользящей средней – фильтрация случайных колебаний

Задача – на основании анализа динамики цены принимать решения о покупке или продаже актива

Необходима торговая стратегия, в соответствии с которой принимаются решения о сделке



Метод скользящей средней

Определение

Временной ряд – множество пар значений (X, Y), где

X – независимая переменная (интервалы времени)

Y – зависимая переменная: $Y(X) = f(X)$

Y – характеристика исследуемого явления

- объем продаж товара
- уровень добычи нефти
- количество больных (здоровых)
- температура воздуха
- ВВП и т.д.
- цена акции на момент X

Суть метода

Метод скользящей средней – один из методов сглаживания и прогнозирования временных рядов.

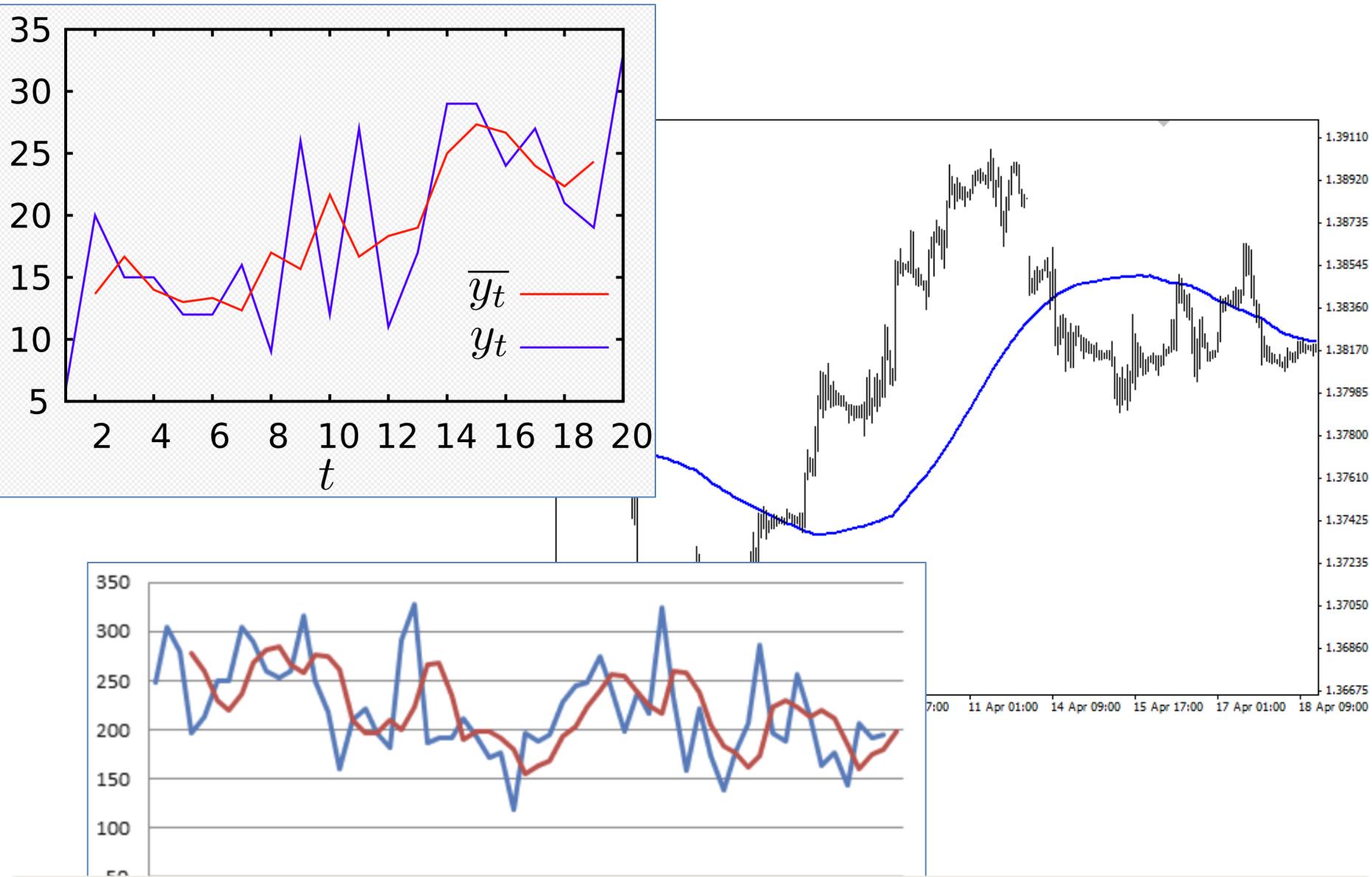
Абсолютные значения $Y(X)$ меняются на **средние арифметические** значения $Y(X)$, взятые на определенном временном интервале T (Окно T).

Окно T скользит: начальные значения $Y(X)$ постепенно убираются, последующие $Y(X)$ подключаются.

В результате формируется ряд сглаженных значений $Z(X) = g(Y, X)$.

Вид $Z(X)$ позволяет проследить общую тенденцию изменений $Y(X)$

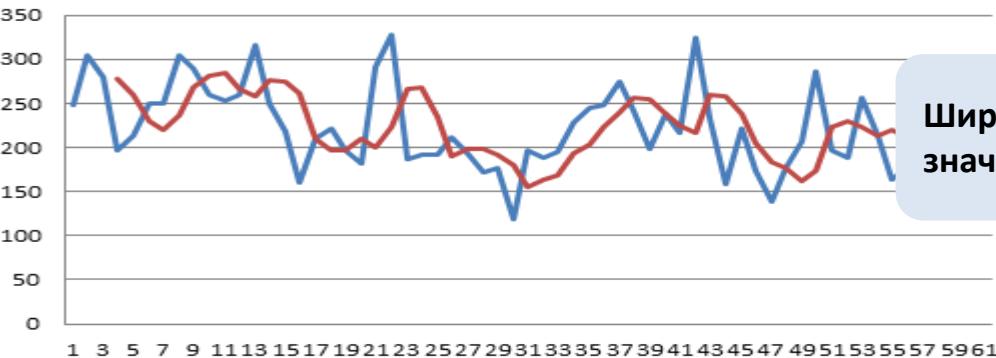
Примеры сглаживания временных рядов методом скользящей средней



Скользящая средняя является **фильтром низких частот**: пропускает низкочастотную активность (долгосрочные циклы и их линии трендов), отсекая высокочастотные случайные колебания

Простая скользящая средняя (Simple Moving Average)

Скользящая средняя Т=3



Ширина окна Т определяет, сколько прошлых значений будет учитываться при прогнозировании

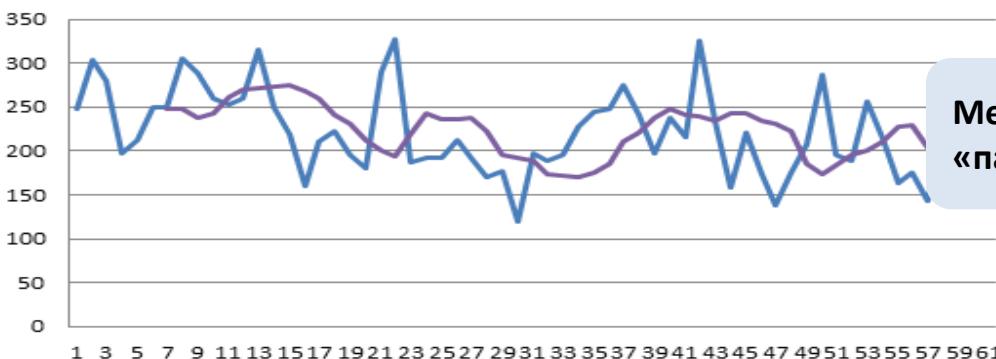
Скользящая средняя Т=4



Чем больше Т, тем более гладким и плавным получится прогноз.

Если взять ширину окна Т равной всему промежутку времени, то получим среднее за весь период

Скользящая средняя Т=6



Меняя размер окна, мы контролируем «память» скользящей средней

Простая скользящая средняя. Сглаживание по разным временным окнам

Скользящая средняя Т=3



Прогноз начинается с 4-го дня

Прогноз на **один** шаг вперед, в качестве прогнозного значения принимается **среднее** за предыдущие **три дня**

Скользящая средняя Т=4

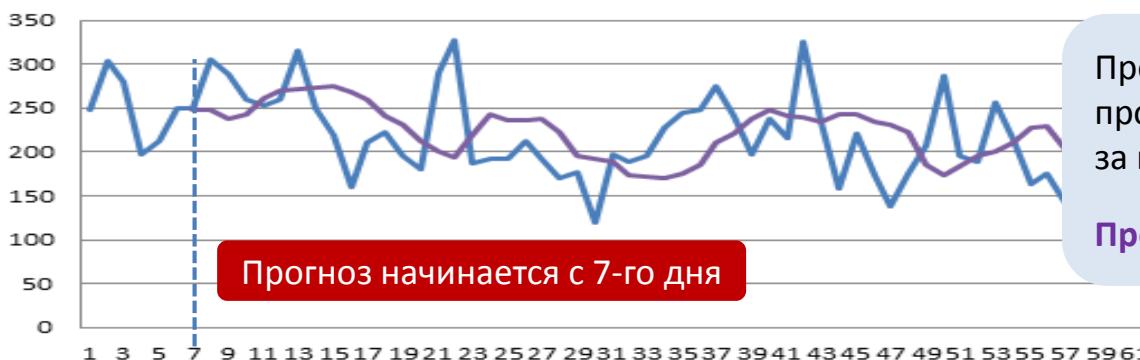


Прогноз начинается с 5-го дня

Прогноз на **один** шаг вперед, в качестве прогнозного значения принимается **среднее** за предыдущие **четыре дня**

Прогноз стал более гладким

Скользящая средняя Т=6



Прогноз начинается с 7-го дня

Прогноз на **один** шаг вперед, в качестве прогнозного значения принимается **среднее** за предыдущие **шесть дней**

Прогноз еще более гладкий

Пример сглаживания ряда. Ширина окна = 3 месяца

Месяцы	Производство продукции (тыс. шт.)	Расчет скользящих средних	Сглаженные уровни ряда
Январь	151	-	-
Февраль	146	$(151+146+152):3$	149,7
Март	152	$(146+152+151):3$	149,7
Апрель	151	$(152+151+154):3$	152,3
Май	154	$(151+154+142):3$	149,0
Июнь	145	$(154+145+149):3$	149,3

Простая скользящая средняя. Алгоритм и формула

Алгоритм

1. Для временного ряда Y_1, Y_2, \dots, Y_N определяется интервал сглаживания M ($M < N$)
 - Если нужно сгладить мелкие колебания, интервал сглаживания берут большим
 - Если нужно сохранить мелкие колебания, интервал сглаживания уменьшают
2. Для первых M значений временного ряда вычисляется их средняя арифметическая; это первое сглаженное значение ряда
3. Затем интервал сглаживания сдвигается на один уровень вправо, вычисление средней арифметической повторяется и т.д.

Расчетная формула

$$Z_j = (Y_{j-M} + Y_{j-M+1} + \dots + Y_{j-1}) / M, \quad j = M + 1, \dots, N$$

Лукьянов Павел Борисович
профессор департамента Анализа данных, принятия решений
и финансовых технологий

Программирование в среде R

Лекция 8

Импорт / экспорт данных в среде R. Файлы CSV

Финансовый университет, 2020

Импорт данных в среду R

Статистические программы и специализированные пакеты

SAS, SPSS, Stata, ...

Текстовые файлы
в различных кодировках

txt, csv, doc, ...

результаты поиска данных
в интернете

web scraping: xml, html, ...

Базы данных – основное хранилище информации

Более 90 % всей информации

Клавиатура

Электронные таблицы

Excel, Google, Gnumeric, OpenOffice Calc

Другое

Oracle +
MySQL +
Microsoft SQL Server +
PostgreSQL +
MongoDB +
DB2 +
Redis +
Elasticsearch +
Microsoft Access
SQLite +

Преимущества и особенности СУБД

- **Целостность** данных, полнота и непротиворечивость
- Возможность **многократного использования** данных
- **Стандартизованные форматы** хранения данных
- **Развитые функции поиска** и получение информации по запросу
- **Высокое быстродействие**, малое время отклика на запрос Пользователя
- **Простота обновления** данных, наличие функций добавления, изменения, удаления
- **Логическая и физическая независимость данных.**
Логическая структура данных может быть изменена без изменения прикладных программ, обрабатывающих данные.
Физическая организация и расположение данных может меняться без изменений в логической структуре данных и прикладных программах
- **Безопасность и защита** от несанкционированного доступа, искажения и уничтожения. Доступ по паролю, шифрование данных
- **Надежность**, децентрализованное хранение. Восстановление данных при сбое техники
- **Совместное использование** данных многими Пользователями
- **Масштабируемость**. Способность СУБД справляться с увеличением рабочей нагрузки, увеличивать производительность путем наращивания аппаратных ресурсов

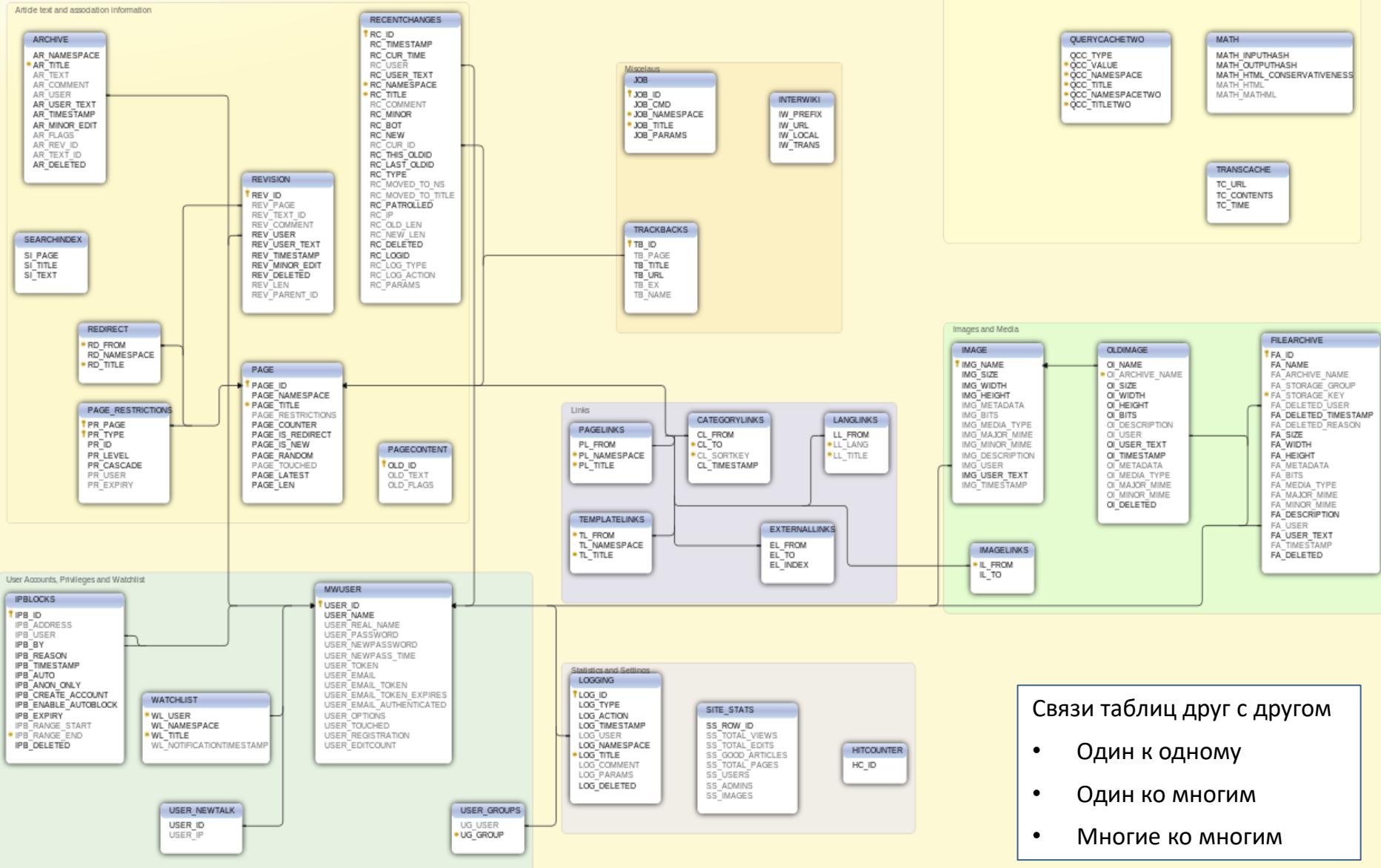
Рейтинг современных СУБД (Database Management System, DBMS)

Рейтинг современных СУБД с сайта www.db-engines.com

357 systems in ranking, May 2020									
Rank				DBMS	Database Model	Score			May 2020
	May 2020	Apr 2020	May 2019			May	Apr	May	
1.	1.	1.	1.	Oracle	Relational, Multi-model	1345.44	+0.02	+59.89	
2.	2.	2.	2.	MySQL	Relational, Multi-model	1282.64	+14.29	+63.67	
3.	3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1078.30	-5.12	+6.12	
4.	4.	4.	4.	PostgreSQL	Relational, Multi-model	514.80	+4.95	+35.91	
5.	5.	5.	5.	MongoDB	Document, Multi-model	438.99	+0.57	+30.92	
6.	6.	6.	6.	IBM Db2	Relational, Multi-model	162.64	-2.99	-11.80	
7.	7.	7.	7.	Elasticsearch	Search engine, Multi-model	149.13	+0.22	+0.51	
8.	8.	8.	8.	Redis	Key-value, Multi-model	143.48	-1.33	-4.93	
9.	9.	↑ 11.	11.	SQLite	Relational	123.03	+0.84	+0.14	
10.	10.	↓ 9.	9.	Microsoft Access	Relational	119.90	-2.02	-23.88	
11.	11.	↓ 10.	10.	Cassandra	Wide column	119.16	-0.91	-6.57	
12.	12.	12.	12.	MariaDB	Relational, Multi-model	90.09	+0.19	+3.57	
13.	13.	13.	13.	Splunk	Search engine	87.75	-0.33	+2.51	
14.	14.	14.	14.	Hive	Relational	81.54	-2.51	+3.64	
15.	15.	15.	15.	Teradata	Relational, Multi-model	73.89	-2.70	-2.15	
16.	16.	↑ 19.	19.	Amazon DynamoDB	Multi-model	64.72	+0.45	+8.78	
17.	↑ 19.	↑ 21.	21.	SAP Adaptive Server	Relational	53.99	+1.37	-1.45	
18.	↓ 17.	↓ 16.	16.	Solr	Search engine	52.58	-1.01	-8.22	
19.	↑ 20.	↓ 18.	18.	FileMaker	Relational	50.96	-1.12	-7.55	
20.	↓ 18.	20.	20.	SAP HANA	Relational, Multi-model	50.54	-2.76	-5.20	
21.	21.	↑ 22.	22.	Neo4j	Graph	49.76	-1.05	-1.27	
22.	22.	↓ 17.	17.	HBase	Wide column	49.72	-0.43	-10.05	
23.	23.	↑ 25.	25.	Microsoft Azure SQL Database	Relational, Multi-model	42.76	+3.80	+13.99	
24.	24.	↑ 26.	26.	Microsoft Azure Cosmos DB	Multi-model	30.68	-1.37	+3.08	
25.	25.	↓ 23.	23.	Couchbase	Document, Multi-model	28.58	-1.83	-6.09	
26.	26.	↑ 29.	29.	Google BigQuery	Relational	27.59	-0.38	+5.52	

Реляционные базы данных – стандарт БД, массовое распространение

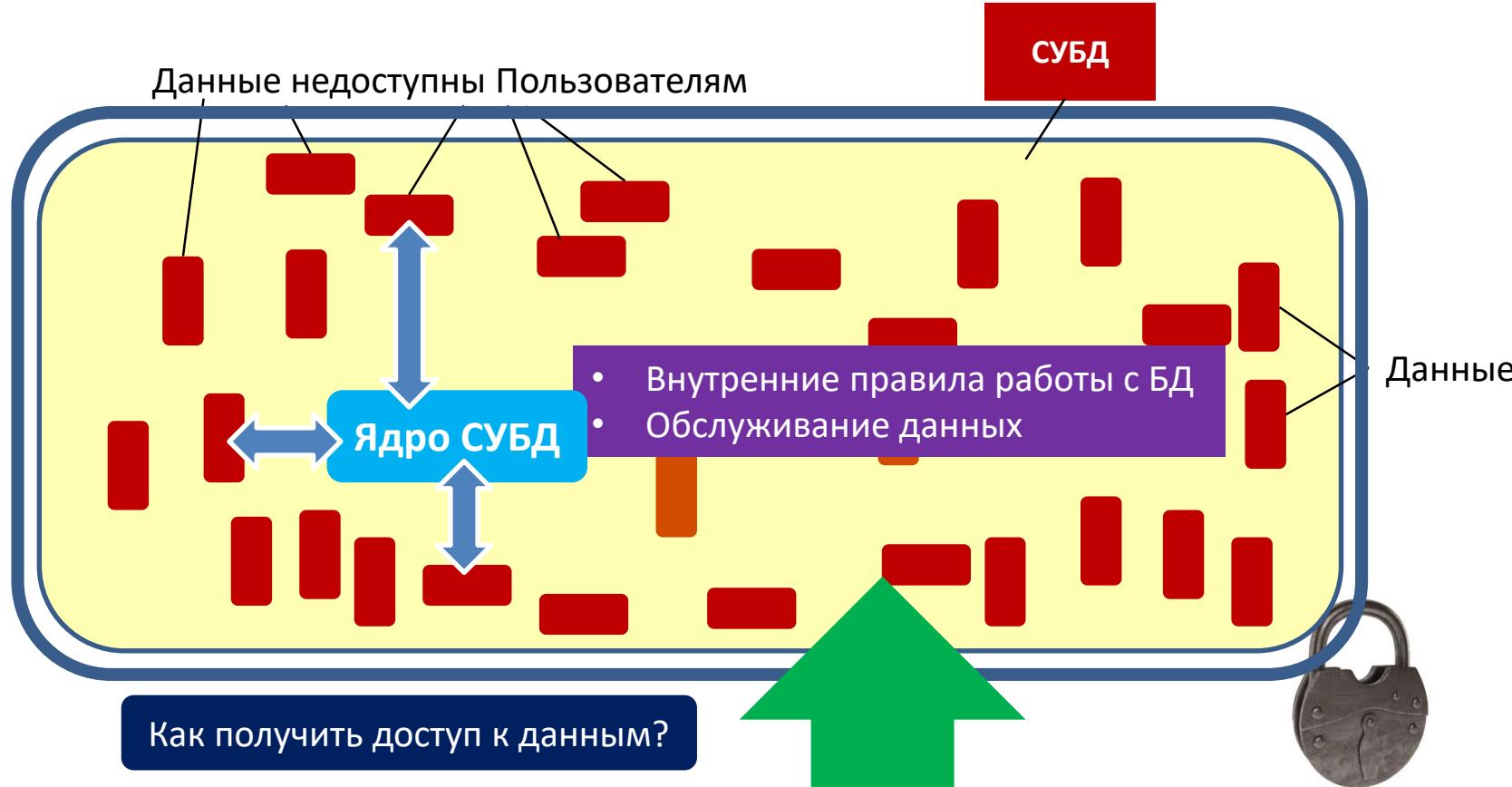
Реляционные БД: представление данных в виде **плоских таблиц**.
Данные расположены на пересечении строк (записей) и столбцов (полей)



Связи таблиц друг с другом

- Один к одному
- Один ко многим
- Многие ко многим

Организация доступа к данным, хранящимся в СУБД



Существуют способы получить информацию из БД,
чтобы «вытащить» необходимые данные



Строение СУБД: представление в виде двух слоев

Ядро СУБД

Законы функционирования

Управление внутренней структурой БД, обеспечение доступа к БД, вопросы надежности, защиты, скорости доступа, прав, хранения, управления транзакциями, масштабирование, сетевые возможности и т.д.

Разрабатывается специалистами-проектировщиками БД.
Разработка СУБД - аналог низкоуровневого
программирования сложного устройства

Интерфейс СУБД

Правила использования

Организация способов доступа к данным, получение и обработка данных, выборки, фильтры, сортировки, анализ, коррекция данных Пользователями и Прикладными программами

Правила обращения к БД: необходим язык обращения к Базе Данных

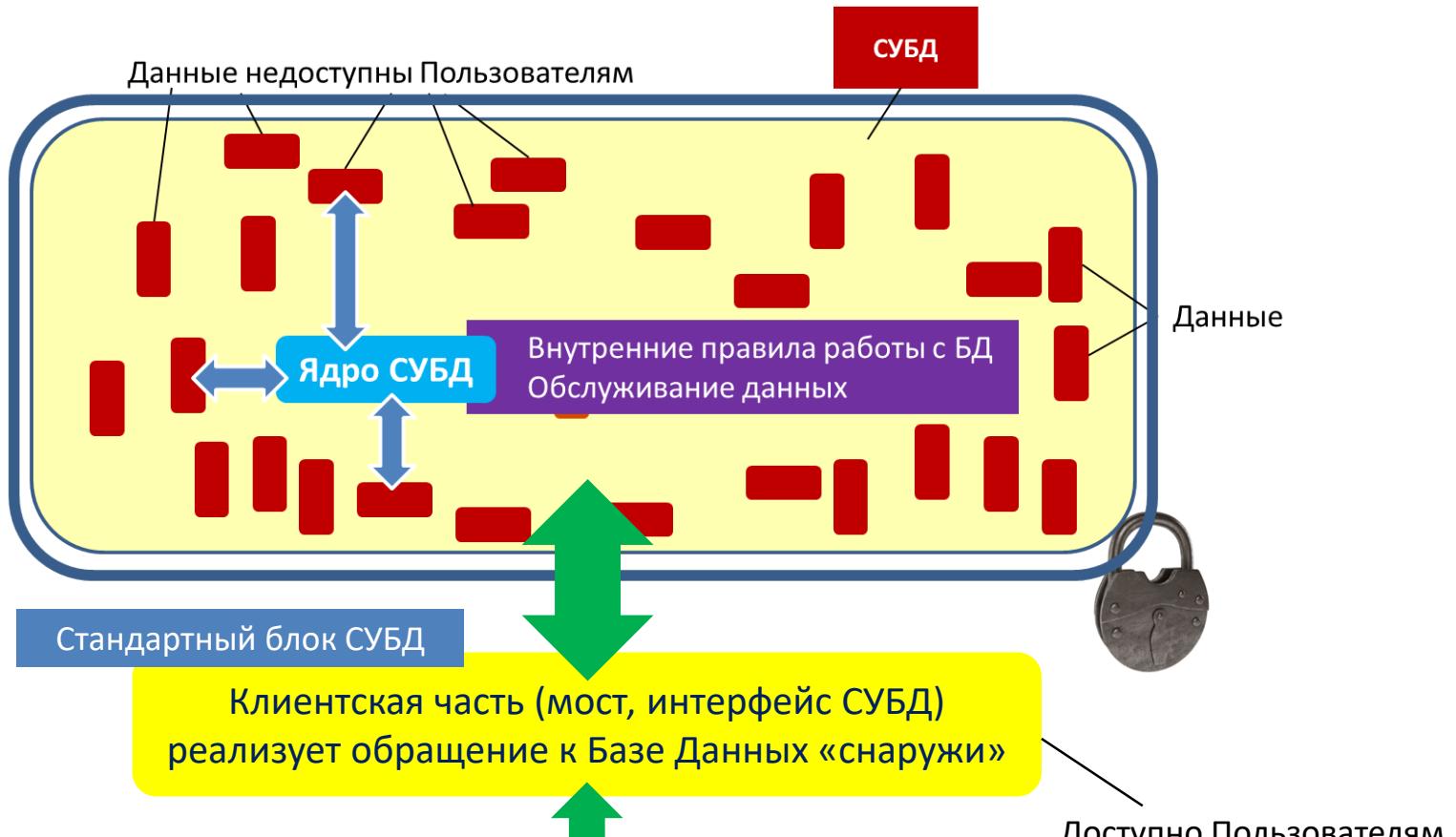
Сторонние программы

Службы и сервисы

Пользователи

Разработчики ПО

Интерфейс к СУБД. Использование языка запросов



Драйвер БД – специальная программа для доступа к клиентской части СУБД.
Зависит от ОС и типа СУБД

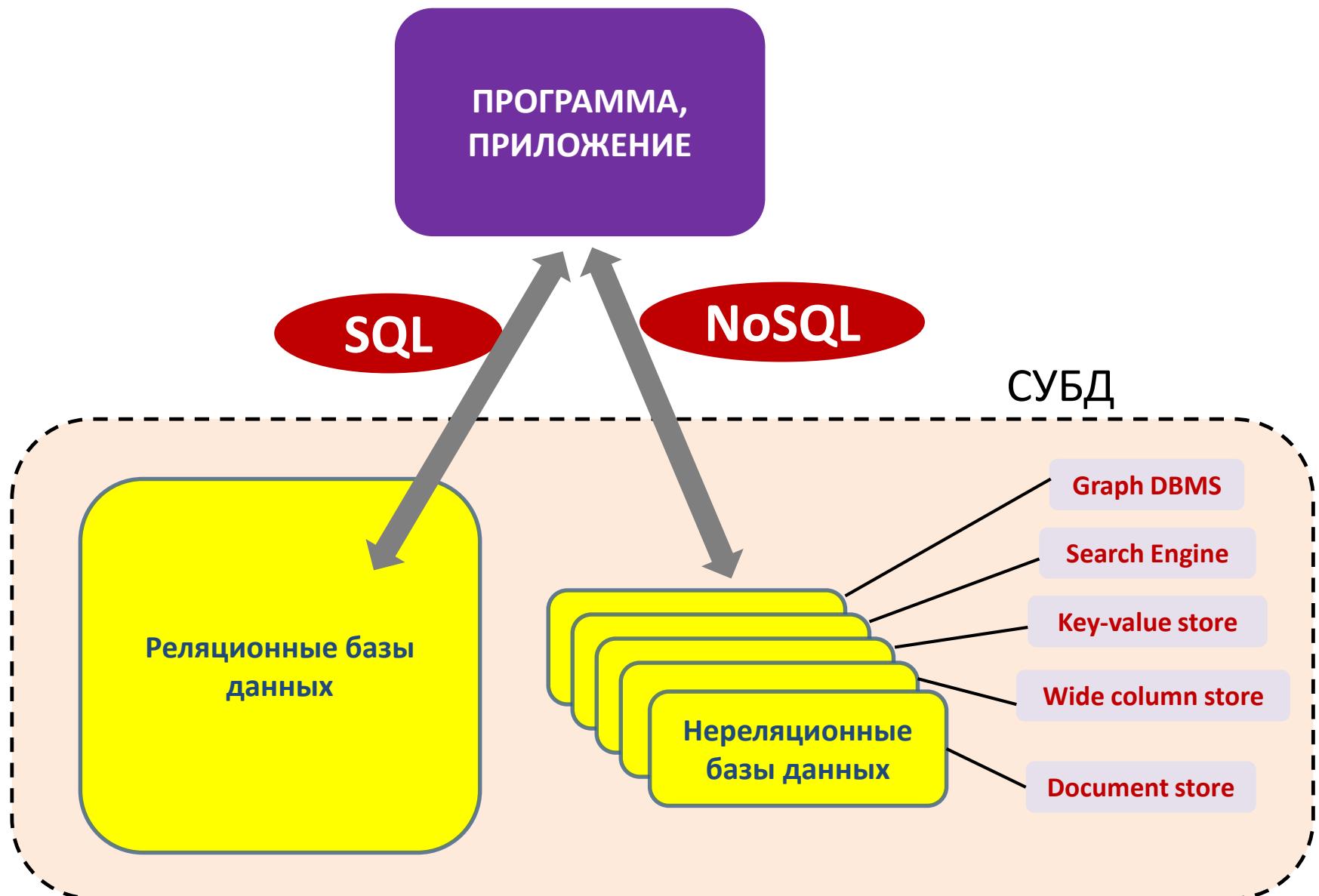
Как работает драйвер

Для доступа к данным в программу-драйвер передаются управляющие конструкции,
написанные на специальном языке запросов:

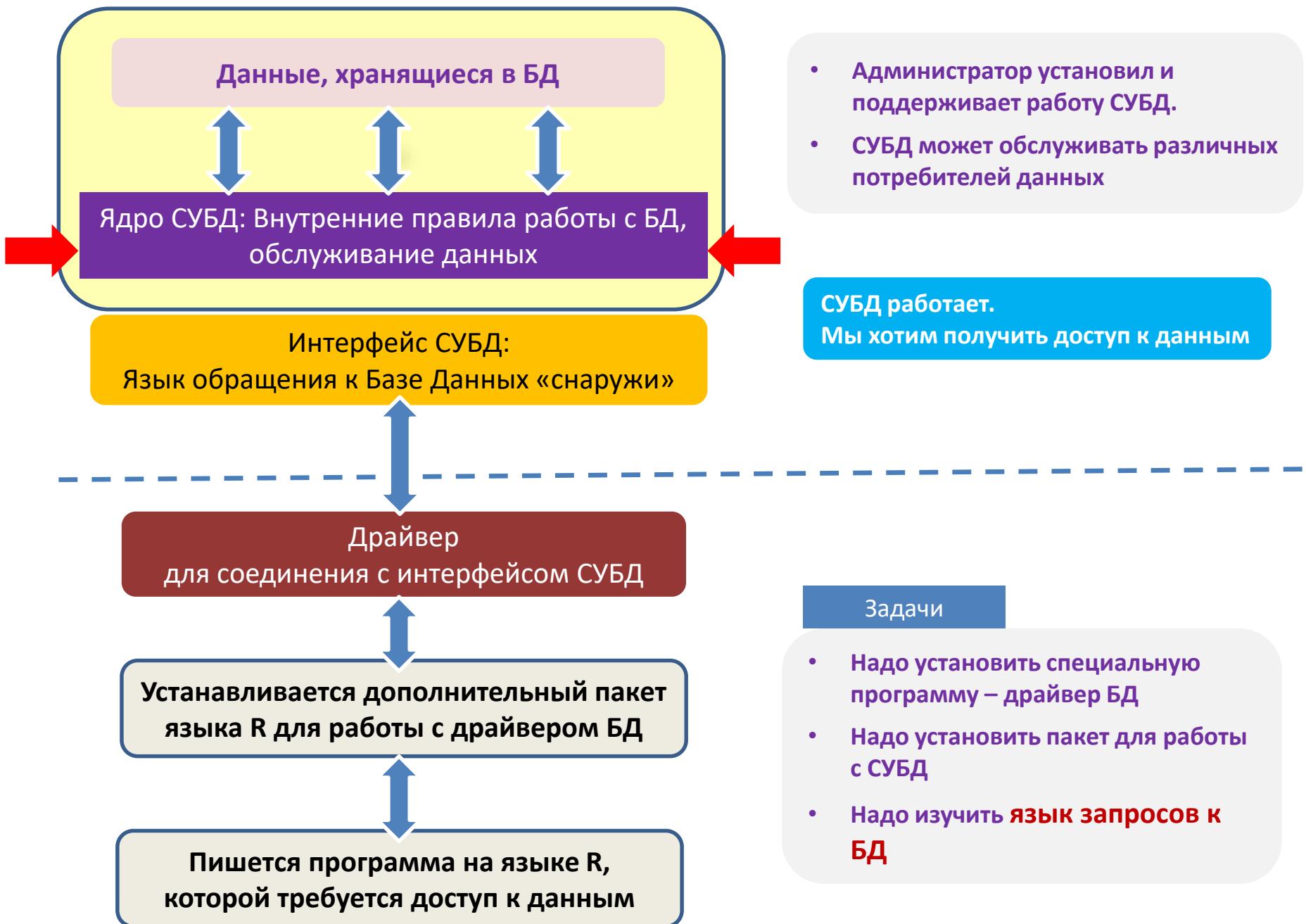
SQL – для реляционных СУБД, NoSQL – для прочих СУБД

В ответ драйвер возвращает запрошенные данные

Языки обращения к данным СУБД из программ и приложений



Использование языка запросов для доступа к СУБД



Доступ к Базе Данных, листинг 1

Листинг 1. BLU for Cloud – R-скрипт

```
1 library(bluR)
2
3 # Установить соединение с сервером баз данных BLU (локальное соединение,
4 # поскольку среда R исполняется на том же сервере)
5 samplescon <- bluConnect("SAMPLEDB", "", "")
6
7 # Создать простой запрос данных в виде строковой переменной
8 query<-paste('select * from DB2INST1.US_FUEL_ECONOMY_AUGUST_2013')
9
10 # Создать кадр данных R на основе SQL-оператора
11 cars <- bludf(samplescon, query)
12
13 # Вывести на печать характеристики кадра данных и некоторые данные из первой строки
14 nrow(cars)
15 ncol(cars)
16 print (cars[1,1:4], row.names=FALSE)
17
18 # Выполнить визуализацию в виде коробчатой диаграммы
19 boxplot(COMB_FE_CONVENTIONAL_FUEL ~ CYL,
20         cars, names = levels(cars$CYL),
21         main="Fuel Consumption - 2013",
22         xlab = "Number of Cylinders",
23         ylab = "Miles/Gallon (mpg)")
24
25 # Закрыть соединение с сервером BLU
26 bluClose(samplescon)
```

Доступ к Базе Данных, листинги 2,3

Листинг 2. RJDBC – извлечение данных

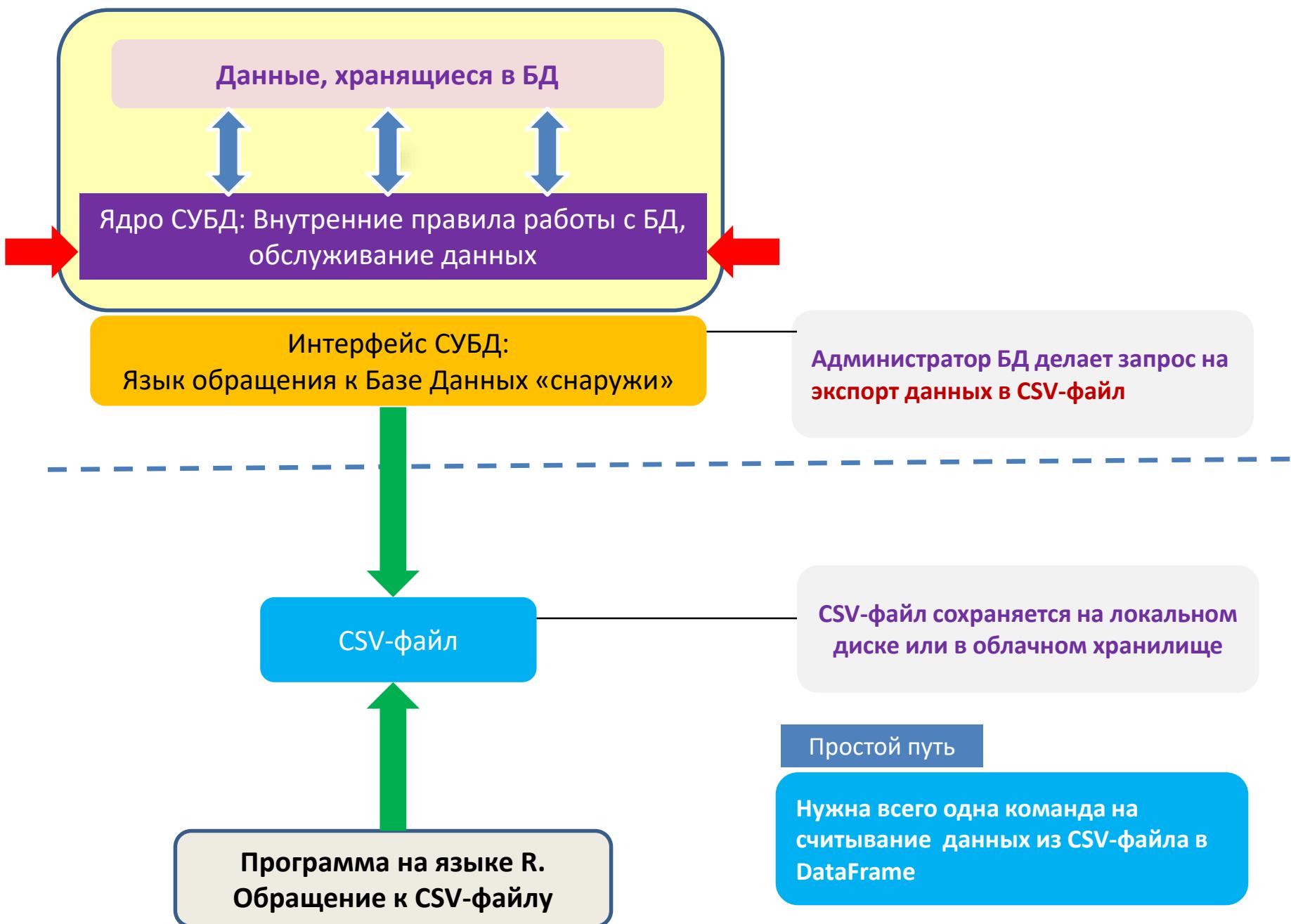
```
1 query <- paste("select * from db2inst1.us_fuel_economy_august_2010")
2
3 # Отправить запрос серверу баз данных
4 rs <- dbSendQuery(conn, query)
5
6 # извлечь все строки данных
7 df <- fetch(rs, -1)
```

Листинг 3. RJDBC – отключение

```
1 # Удалить все результаты с отсутствующими значениями
2 df <- na.omit(df)
3
4 cat ("There are", nrow(df), "fuel economy results available.\n")
5 cat (ncol(df), "different variables.\n\n")
6
7 # Найти в кадре данных наилучшие показатели по потреблению топлива
8 best_fe <- max(df$COMB_FE_CONVENTIONAL_FUEL,na.rm=TRUE)
9
10 cat("\nCar(s) with the best fuel consumption of",best_fe,"miles/gallon.\n\n")
11
12 print (df[df$COMB_FE_CONVENTIONAL_FUEL==best_fe,c(1:4)],row.names=FALSE)
13
14 # Отключиться от сервера баз данных
15 dbDisconnect(conn)
16
17 ---- OUTPUT from Script
18
19 There are 1165 fuel economy results available with 18 different variables.
20
```

- Сложный и долгий путь работы с БД
- Необходимо знание языка SQL (язык запросов к БД)
- Как получить данные быстрее?

Доступ к данным через импорт CSV-файла, экспортируемого из СУБД



Файлы CSV. Определение

CSV (Comma Separated Values) — значения, разделённые запятыми

Формат CSV - специальный текстовый формат, предназначенный для представления табличных данных и передачи данных между программами

Используется, если модель базы данных – Relational DBMS

- Каждая строка файла — одна строка таблицы, одна запись (в терминах БД)
- Запись состоит из полей – значений столбцов
- Все поля записи, расположенные в одной строке, выводятся друг за другом
- В качестве разделителя полей (столбцов) используется символ «запятая»
- В качестве разделителя полей (столбцов) может использоваться пробел, символ табуляции, точка с запятой и т.д.
- Если в полях таблицы содержатся зарезервированные символы (двойная кавычка, запятая, точка с запятой и т.д.), то эти символы помещают в двойные кавычки.
- Если в значении поля встречаются кавычки, в файле они будут представлены в виде двух кавычек подряд.

Tab Separated Values (TSV) – значения, разделенные горизонтальной табуляцией

Delimiter Separated Values (DSV) – значения, разделенные разделителем – общее название файлов табличной структуры в текстовом формате

Файлы CSV. Пример

__NounName	Name	Handles	VM	WS	PM	NPM	Path	Company	CPU	Fileversion	Proc
"Process",	"atieclxx",	"207",	"66068480",	"6533120",	"2588672",	"9736",	"8",	"207",	"1324"
"Process",	"atiesrxx",	"122",	"33804288",	"4505600",	"1765376",	"7632",	"8",	"122",	"936",
"Process",	"audiodg",	"121",	"50356224",	"15941632",	"15011840",	"9424",	"8",	"121",	"2908"
"Process",	"CCC",	"699",	"674254848",	"6746112",	"61571072",	"76332",	"C:\Program Files (x86)\ATI Te				
"Process",	"conhost",	"58",	"85331968",	"7323648",	"2846720",	"7920",	"C:\windows\system32\conhost.e				
"Process",	"csrss",	"747",	"57634816",	"5296128",	"3284992",	"13568",	"13",	"747",	"424",
"Process",	"csrss",	"587",	"110956544",	"14876672",	"8941568",	"19896",	"13",	"587",	"504",
"Process",	"DCPSysMgr",	"169",	"109682688",	"12632064",	"8765440",	"13208",	"C:\Program Files\DELL\DC				
"Process",	"DCPSysMgrSvc",	"138",	"59002880",	"7610368",	"2768896",	"11408",	"8",	"138",	"2
"Process",	"dell.ControlPoint",	"384",	"605573120",	"49803264",	"43466752",	"42724",	"C:\Program Fil				
"Process",	"dpupdchk",	"221",	"116756480",	"10899456",	"4263936",	"15816",	"C:\Program Files\Microsc				
"Process",	"dwm",	"146",	"201625600",	"50298880",	"38662144",	"23224",	"C:\windows\system32\dwm.exe"				
"Process",	"explorer",	"1302",	"451448832",	"113147904",	"88936448",	"80720",	"C:\windows\Explorer.E				
"Process",	"Flashutil10i_ActiveX",	"84",	"72155136",	"5959680",	"1814528",	"9664",	"C:\Windows\SysW				
"Process",	"IAANotif",	"112",	"78155776",	"7221248",	"2453504",	"11832",	"C:\Program Files (x86)\Int				
"Process",	"IAANTmon",	"173",	"55066624",	"6647808",	"2437120",	"19696",	"8",	"173",	"2100"
"Process",	"Iap",	"82",	"43925504",	"290816",	"2215936",	"8776",	"8",	"82",	"1796",
"Process",	"Idle",	"0",	"0",	"24576",	"0",	"0",	"0",	"0",	"0",
"Process",	"ielowutil",	"79",	"57462784",	"5152768",	"1527808",	"9184",	"C:\Program Files (x86)\Inte				
"Process",	"iexplore",	"498",	"270987264",	"56807424",	"43868160",	"45256",	"C:\Program Files (x86)\				
"Process",	"iexplore",	"717",	"234233856",	"35090432",	"16297984",	"53896",	"C:\Program Files (x86)\				
"Process",	"iexplore",	"805",	"398778368",	"88436736",	"71942144",	"67092",	"C:\Program Files (x86)\				
"Process",	"iexplore",	"819",	"410828800",	"90341376",	"89747456",	"77412",	"C:\Program Files (x86)\				
"Process",	"inetinfo",	"135",	"76029952",	"15511552",	"9428992",	"14136",	"8",	"135",	"1852"
"Process",	"ipoint",	"266",	"147828736",	"21655552",	"10031104",	"19832",	"C:\Program Files\Microsof				
"Process",	"LMS",	"118",	"42680320",	"5070848",	"1871872",	"19248",	"8",	"118",	"1916",
"Process",	"lsass",	"795",	"51118080",	"14733312",	"6144000",	"26736",	"9",	"795",	"564",
"Process",	"lsm",	"162",	"22786048",	"4567040",	"2777088",	"7480",	"8",	"162",	"576",
"Process",	"mdm",	"87",	"55402496",	"5668864",	"2301952",	"9248",	"8",	"87",	"1940",
"Process",	"MOM",	"363",	"603275264",	"4317184",	"40218624",	"34076",	"C:\Program Files (x86)\ATI Te				
"Process",	"MSCam564",	"122",	"54157312",	"7057408",	"6586368",	"10696",	"8",	"122",	"1996"
"Process",	"MsMpEng",	"481",	"338337792",	"103079936",	"188940288",	"84488",	"8",	"481",	"8

Файлы CSV. Пример

The screenshot shows a web browser window with the URL <http://localhost:3000/products.csv> in the address bar. The page content displays a CSV file with the following data:

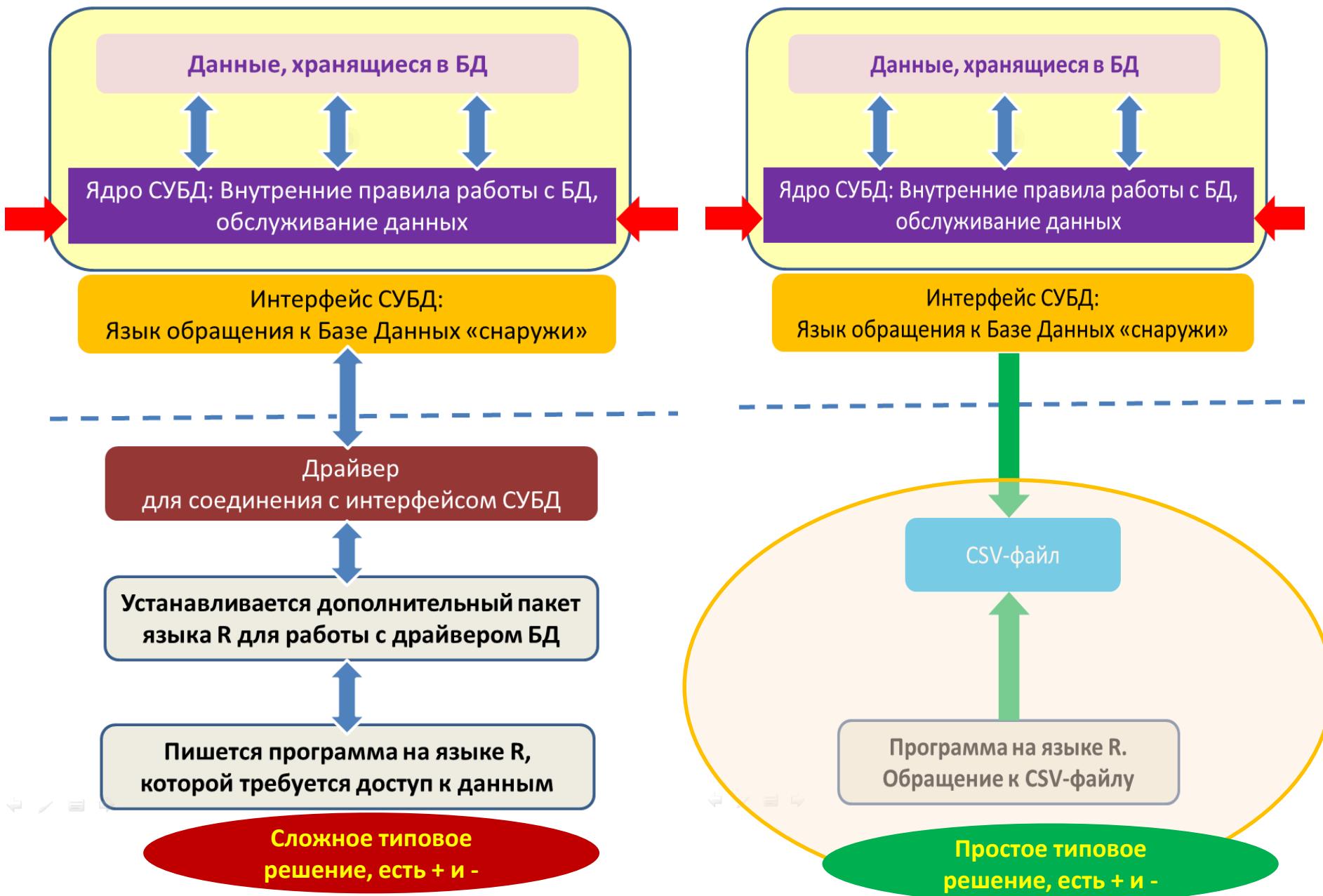
id	name	released_on	price	created_at	updated_at
24	1000 Piece Jigsaw Puzzle	2012-07-03	14.99	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
30	360° Protractor	2012-05-03	3.99	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
17	7 Wonders	2012-04-21	28.75	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
13	Acoustic Guitar	2012-06-06	1025.0	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
15	Agricola	2012-05-22	45.99	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
22	Answer to Everything	2012-07-03	42.0	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
23	Box Kite	2012-05-19	63.0	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
29	CanCan Music Record	2012-05-09	2.99	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
12	Chocolate Pie	2012-04-12	3.14	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
9	Dog Toy Bone	2012-06-13	2.99	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
11	Flux Capacitor	2012-06-01	19.55	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
6	Game Console	2012-06-06	299.95	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
10	Heated Blanket	2012-07-19	27.95	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
19	Knights of Catan	2012-06-10	19.95	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
8	Lawn Chair	2012-05-29	34.99	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
21	Millennium Falcon	2012-04-10	3597200.0	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
14	Model Enterprise	2012-04-18	27.99	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
28	Model Train Rails	2012-06-30	45.0	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
3	Oak Coffee Table	2012-07-08	223.99	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
5	Oh's Cereal	2012-04-17	3.95	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC
27	Rock	2012-07-06	22.40	2012-07-09 16:50:49 UTC	2012-07-09 16:50:49 UTC

Файлы CSV. Пример, таблица с заголовком

The screenshot shows a Windows Notepad window titled "sales — Блокнот". The menu bar includes "Файл", "Правка", "Формат", and a red-highlighted "Названия полей таблицы". The main content area displays a table with two columns: "День недели" and "Продажи, тыс. руб". The first row contains the header names. Below the header, there are seven data rows, each consisting of a day of the week and its corresponding sales value. A red circle highlights the separator ";" in the second row, and another red circle highlights the separator ";" in the fourth row. A red box labeled "Разделитель" is positioned over the circled separator in the fourth row.

День недели	Продажи, тыс. руб
1;127	
2;203	
3;155	
4;159	
5;142	
6;112	
7;95	

Два способа доступа к данным БД



Как считать CSV-файл в таблицу среды R?

```
getwd()  
setwd('c:/R/e-shop/result')  
z<-read.table("sales.csv", sep=";", head=TRUE)
```

Имя файла

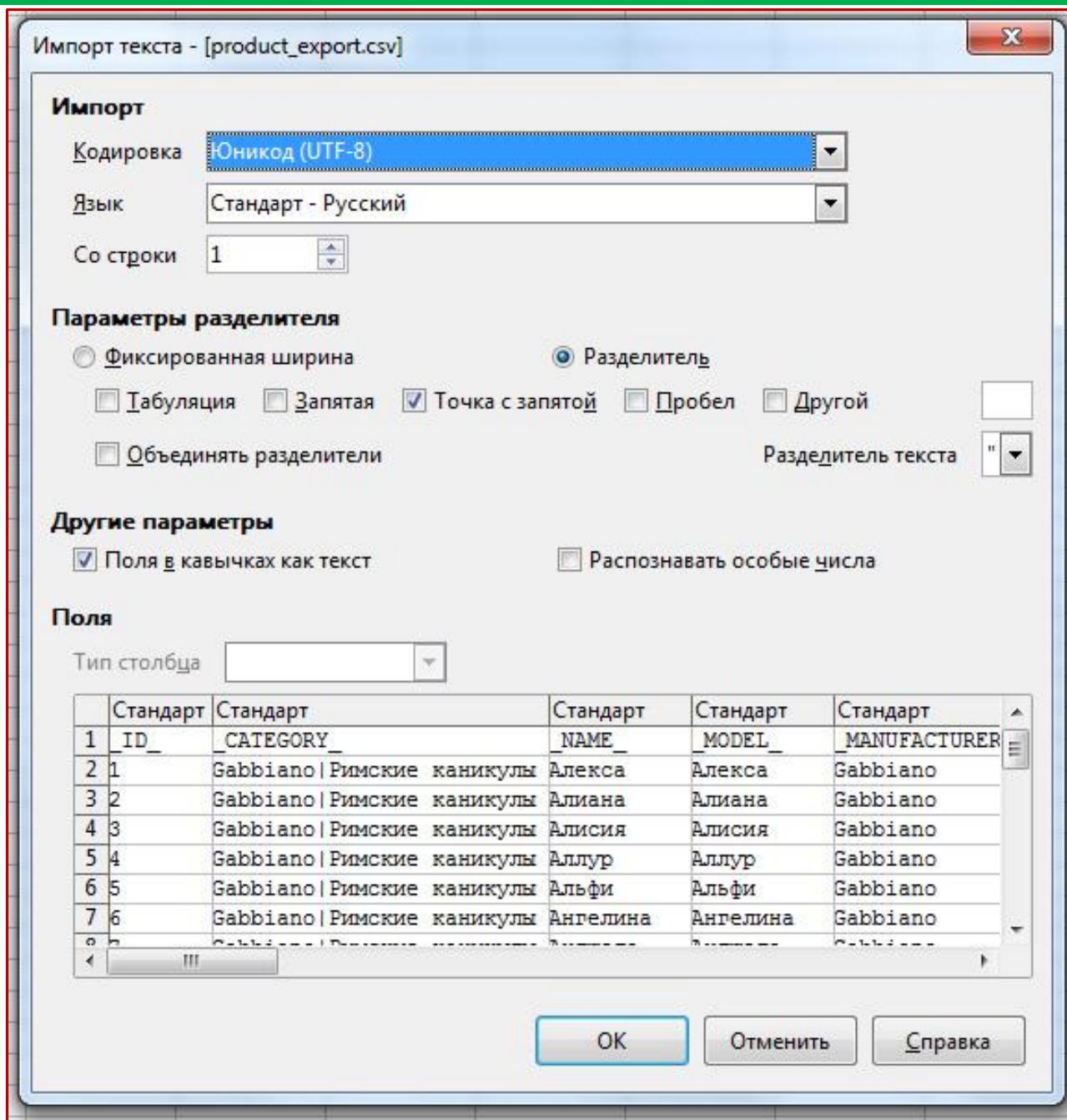
Разделитель полей

Считывать имена столбцов

```
read.table("results/sales.csv", sep=";", head=TRUE, dec = ",")
```

Разделитель целой и дробной частей числа

Файлы CSV – активное использование в прикладных программах



Файлы CSV – активное использование в торговых программах

1	A Наименование ↓	B Код артикула ↓	C Валюта ↓	D Цена ↓	E Доступен для заказа ↓	F Зачеркнутая цена ↓	G Цена заводская ↓	H В наличии ↓	I Краткое описание ↓
	Наименование	Код арт.	Валют	Цена	Доступен для за	Зачеркнутая	Цена завод	В нал	Краткое ог
2	Лампы светодиодные								
3	⚠ !Лампы подкатегория 1								
4	LB-91 (7W) 230V E27 ...	25444	RUB	127	1			500	Oх какая лаг
5	LB-70 (3.5W) 230V E2...	25277	RUB	60	1			500	
6	LB-72 (5W) 230V E14 ...	25400	RUB	101	1			150	
7	⚠ !Лампы подкатегория 2								
8	LB-37 (1W) 230V E27 ...	25118	RUB	78	1			115	
9	LB-41 6LED(3,5W) 230...	25328	RUB	163	1			9	
Проверка 9 из ~9 строк									

Столбец идентификации товаров

Ссылка

Выберите столбец (свойство), значения которого однозначно идентифицируют Каждый товар. В зависимости от значений этого столбца процедура импорта либо обновит существующий товар, либо создаст новый.

Столбец идентификации артикулов

Код артикула

Так же, как и для товаров, выберите столбец (свойство) идентификации артикулов, который уникальным образом отличает Каждый артикул. В зависимости от значения в столбце идентификации, указанного в CSV-файле, операция импорта либо обновит существующий артикул, либо создаст новый артикул для данного товара.

Список столбцов

13 столбцов будут обработаны

Наименование	Код артикула	Валюта	Цена	Доступен для заказа	Зачеркнутая цена	Цена заводская	В наличии	Краткое описание
Описание	Наклейка	Теги	Ссылка на витрину					

✓ Загрузка файла и выбор соответствия столбцов

► Проверка параметров

3. Импорт

В CSV-файле обнаружена и готова к импорту следующая информация:

- ✓ 3 новые категории
- ✓ 5 новых товаров
- ✓ 5 новых артикулов
- ⚠ Избыточное описание категорий на 1 строке файла

Если результаты проверки не соответствуют вашим ожиданиям, измените соответствие столбцов и выбор параметров идентификации и выполните проверку повторно.

Файлы CSV – получение котировок акций

МосБиржа акции ▾ Сбербанк

Интервал и периодичность

01.11.2018



– 18.11.2018



1 час

Имя выходного файла

SBER_181101_181118

.csv

Имя контракта

SBER

Формат

даты

dd/mm/gг

времени

ЧЧММСС

Выдавать время

начала свечи

окончания свечи

МОСКОВСКОЕ

Разделитель

полей

запятая (,)

разрядов

нет

Формат записи в файл

TICKER, PER, DATE, TIME, OPEN, HIGH, LOW, CLOSE, VOL

Добавить заголовок файла



Заполнять периоды без сделок



Получить файл

Файлы CSV – файл котировок

SBER_181101_181118.csv									
<TICKER>;<PER>;<DATE>;<TIME>;<OPEN>;<HIGH>;<LOW>;<CLOSE>;<VOL>									
SBER; 60; 01/11/18; 110000; 189.5400000; 190.8200000; 188.5000000; 189.1500000; 15578920									
SBER; 60; 01/11/18; 120000; 189.1900000; 191.3600000; 189.0200000; 190.0500000; 11573650									
SBER; 60; 01/11/18; 130000; 190.0500000; 190.2000000; 187.2600000; 188.3500000; 10361370									
SBER; 60; 01/11/18; 140000; 188.2800000; 189.2700000; 187.6100000; 188.2500000; 4056270									
SBER; 60; 01/11/18; 150000; 188.2700000; 189.4400000; 187.5900000; 189.2200000; 4432780									
SBER; 60; 01/11/18; 160000; 189.2200000; 189.4500000; 187.5000000; 187.6000000; 3782160									
SBER; 60; 01/11/18; 170000; 187.6000000; 188.5000000; 187.0900000; 187.9900000; 8646820									
SBER; 60; 01/11/18; 180000; 187.9900000; 189.5000000; 186.7000000; 189.4900000; 11321120									
SBER; 60; 01/11/18; 190000; 189.4900000; 190.2300000; 188.6700000; 188.6700000; 7735210									
SBER; 60; 02/11/18; 110000; 189.9700000; 190.6700000; 188.5700000; 190.0000000; 8498460									
SBER; 60; 02/11/18; 120000; 189.9700000; 192.5000000; 189.1900000; 192.4800000; 13100840									
SBER; 60; 02/11/18; 130000; 192.4800000; 192.9800000; 191.6000000; 191.9500000; 6851680									
SBER; 60; 02/11/18; 140000; 192.0000000; 192.9200000; 191.8500000; 192.2600000; 4359510									
SBER; 60; 02/11/18; 150000; 192.2600000; 192.7000000; 191.2600000; 192.3000000; 5475080									
SBER; 60; 02/11/18; 160000; 192.3300000; 192.5000000; 191.4200000; 192.0200000; 4297410									
SBER; 60; 02/11/18; 170000; 192.0100000; 192.1000000; 190.7000000; 191.6300000; 8155600									
SBER; 60; 02/11/18; 180000; 191.5900000; 192.6400000; 190.9000000; 192.2200000; 9542550									
SBER; 60; 02/11/18; 190000; 192.2200000; 192.7500000; 191.5300000; 192.6000000; 8820230									
SBER; 60; 06/11/18; 110000; 194.5100000; 195.7500000; 193.7300000; 195.3300000; 14996830									
SBER; 60; 06/11/18; 120000; 195.3400000; 196.7900000; 195.1700000; 196.2500000; 10048130									
SBER; 60; 06/11/18; 130000; 196.2500000; 196.7400000; 195.6800000; 196.1100000; 5364540									
SBER; 60; 06/11/18; 140000; 196.1100000; 197.2500000; 195.3800000; 196.3000000; 8865700									
SBER; 60; 06/11/18; 150000; 196.2500000; 196.6000000; 196.1000000; 196.6000000; 3334320									
SBER; 60; 06/11/18; 160000; 196.5900000; 197.0000000; 196.0500000; 196.7600000; 5406320									
SBER; 60; 06/11/18; 170000; 196.7700000; 197.7300000; 195.9700000; 197.4100000; 6535780									
SBER; 60; 06/11/18; 180000; 197.4100000; 197.8500000; 196.6300000; 197.1500000; 5761090									
SBER; 60; 06/11/18; 190000; 197.1600000; 197.8000000; 197.0300000; 197.8000000; 5163060									
SBER; 60; 07/11/18; 110000; 197.4100000; 197.7000000; 195.4000000; 195.7900000; 12002450									
SBER; 60; 07/11/18; 120000; 195.8000000; 197.3500000; 195.7100000; 197.1600000; 8262750									
SBER; 60; 07/11/18; 130000; 197.1900000; 202.2700000; 197.1000000; 201.8800000; 19327040									
SBER; 60; 07/11/18; 140000; 201.8200000; 203.0000000; 200.6500000; 202.0900000; 13946540									
SBER; 60; 07/11/18; 150000; 202.0800000; 202.1500000; 200.2500000; 200.5000000; 8289760									
SBER; 60; 07/11/18; 160000; 200.5900000; 201.5200000; 198.2200000; 198.7500000; 12254460									