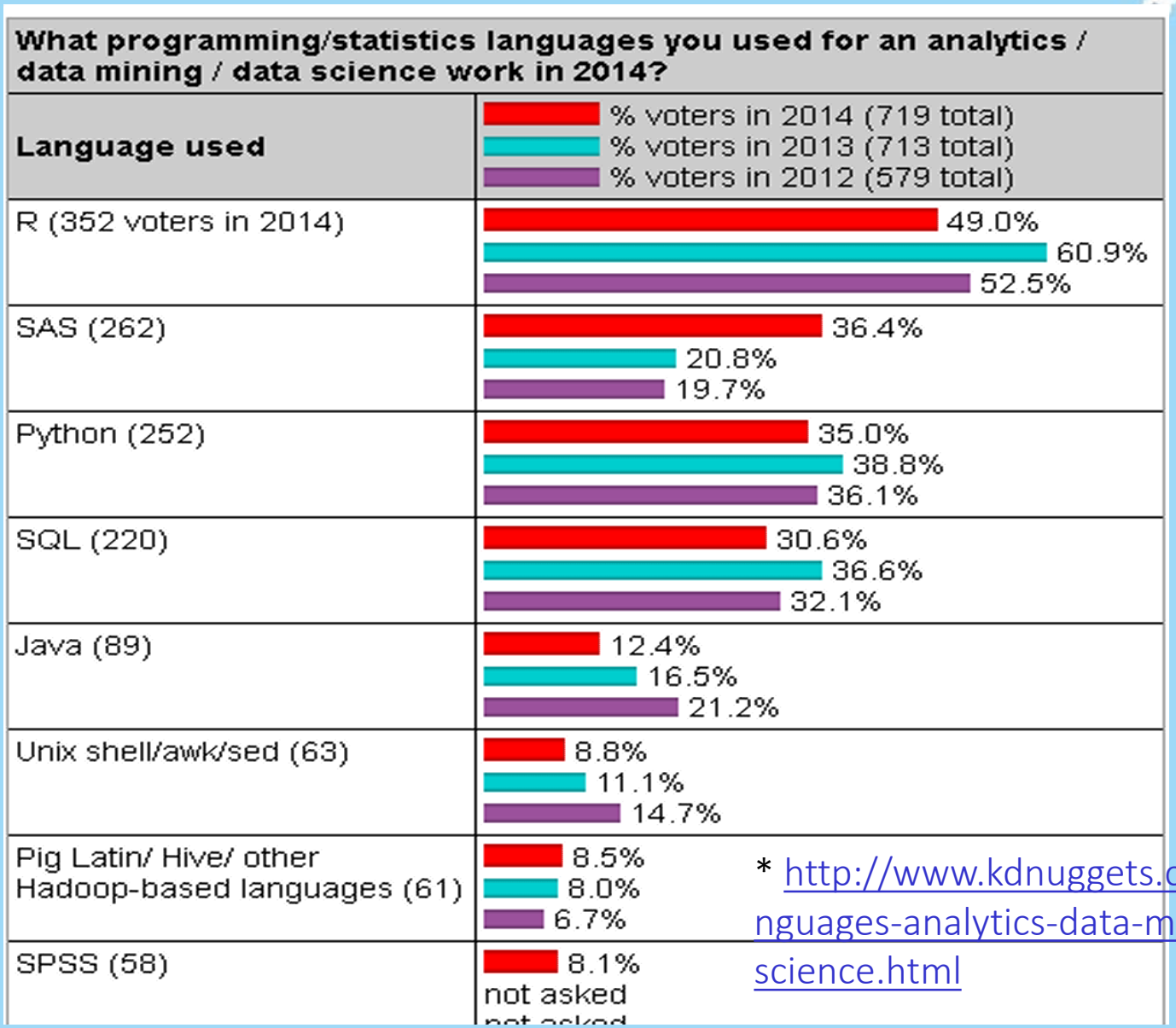
Курс Обработка больших данных

Лекция 2

Инструменты для обработки больших данных

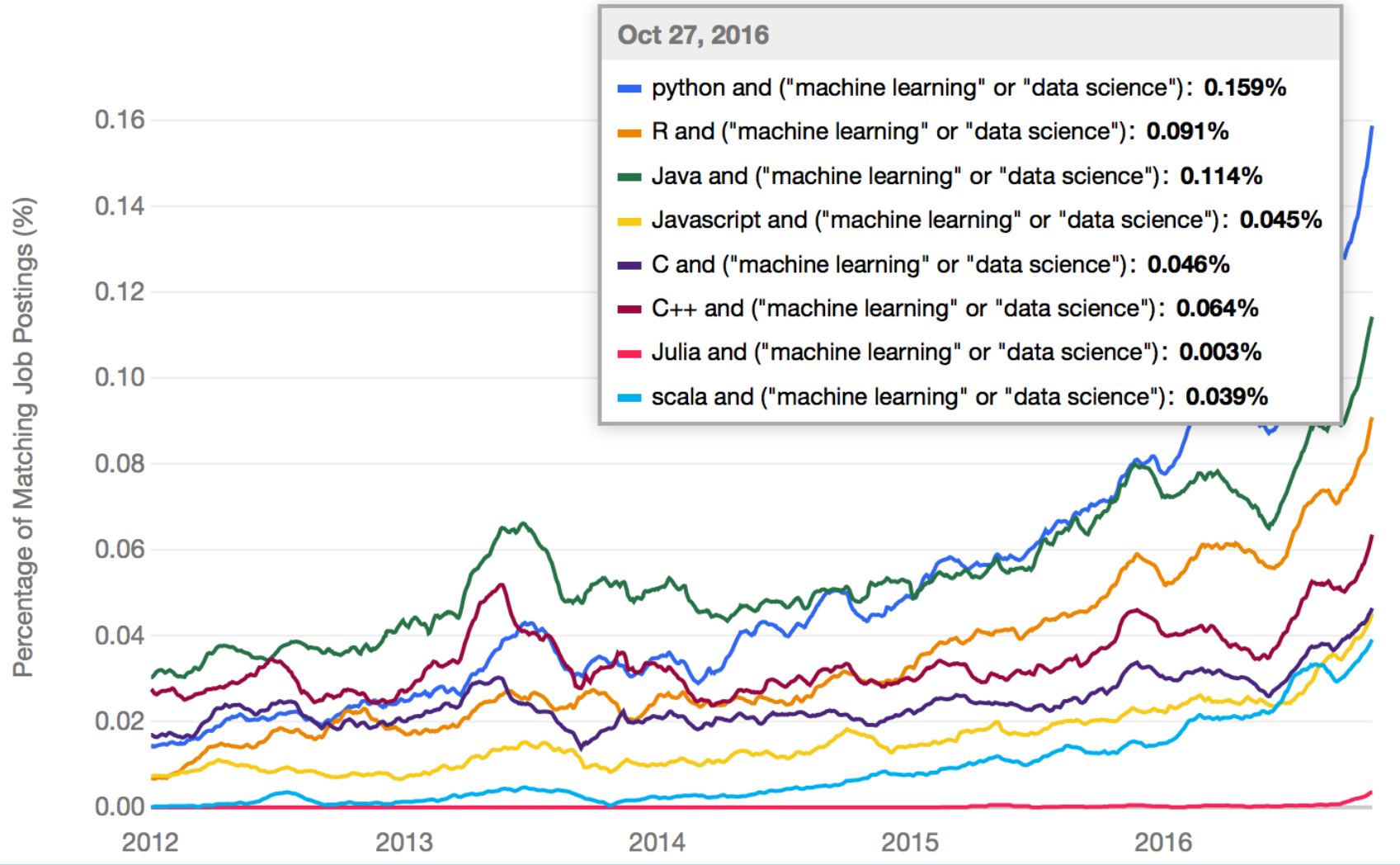
Содержание

Чем работать с большими данными?



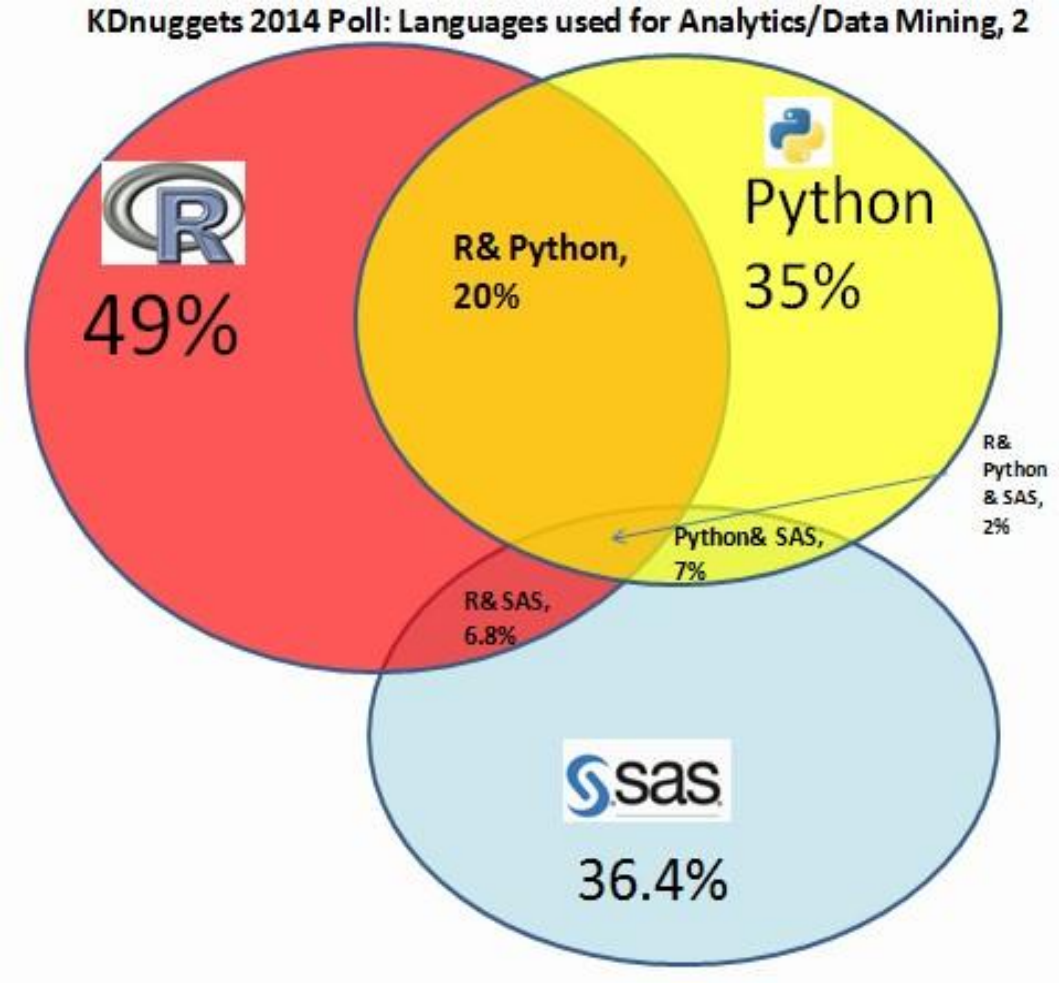
* http://www.kdnuggets.com/polls/2013/languages-analytics-data-mining-data-science.html

Чем работать с большими данными?



http://www.kdnuggets.com/2017/01/most-popular-language-machine-learning-data-science.html

Чем работать с большими данными?

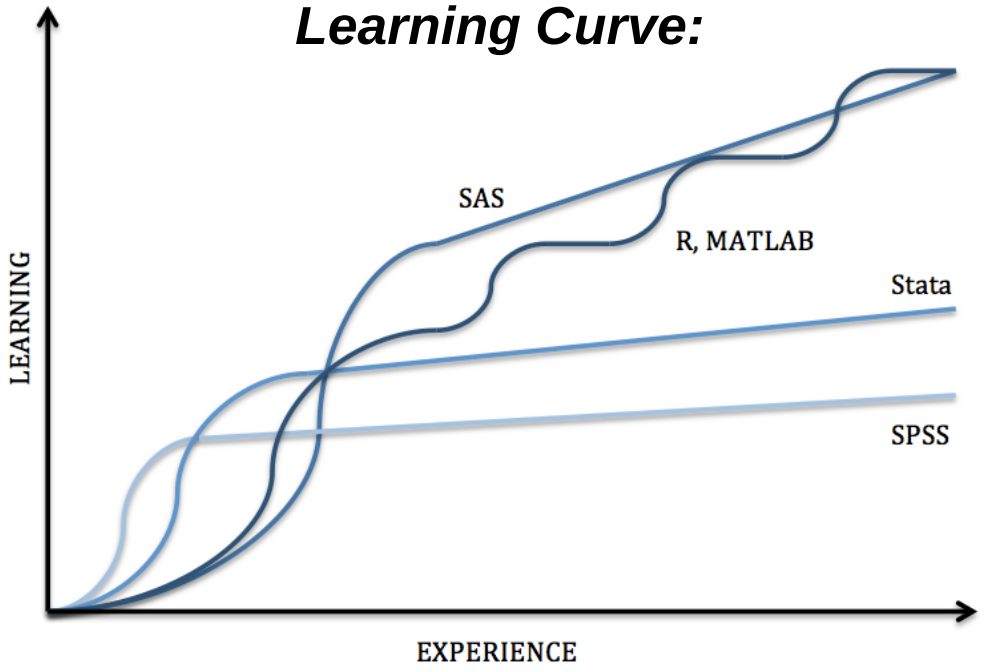


2 Сравнение продуктов Data Mining

Сравнение продуктов DM

В некоторых источниках предлагается разделять программные продукты для анализа данных на две большие группы:

- ✓ Решения, ориентированные на программирование: R, MATLAB, SciPy.
- ✓ Решения, ориентированные на анализ данных: MS Excel, SAS, SPSS, Stata.



Так называемая «дуга обучения», говорит нам о сложности освоения инструментов DM: чем круче кривая – тем быстрее можно освоить решение некого усредненного набора задач (множественная регрессия, непараметрический анализ…)

Summary - Which Statistical Software to use?
https://sites.google.com/a/nyu.edu/statistical-software-guide/summary

SAS — это аббревиатура от Statistical Analysis System, которая со временем стала использоваться в качестве имени собственного для обозначения как самой компании, так и её продуктов, давно уже вышедших за рамки только приложений для статистического анализа.

Основные приложения SAS — настраиваемые системы Business Intelligence для финансового менеджмента, управления рисками, маркетинга, управления цепочками поставок. В решениях учитывается отраслевая специфика, поставляются различные решения для разных отраслей.

http://www.sas.com/ru_ru/software/university-edition.html

КУРС: ИСПОЛЬЗОВАНИЕ СИСТЕМЫ SAS ДЛЯ АНАЛИЗА И ПРЕДСТАВЛЕНИЯ ДАННЫХ: **http://pubhealth.spb.ru/SASDIST/**

Уроки по SAS: **http://sas-system.blog.ru/?attempt=1**

SAS — также позиционируется как язык программирования для статистических исследований, т.е язык, который специализируется на обработке данных. Он находится на высоком уровне, поэтому SAS часто называют «языком четвертого поколения» - 4GL.

Шаги обработки данных, которые присущи SAS, условно делятся на два типа:

-шаги данных;

-шаги процедур.

- Первая категория - средство обработки данных в SAS System. Это цикл по записям источника данных, который имеет возможность осуществлять ряд произвольных операций в течении каждой из итераций данного цикла. Благодаря возможностям шага данных реализуется любой алгоритм их обработки.
- Вторая категория - вызов одной из операций Base SAS, а также других продуктов SAS System. Каждая отдельная процедура совершает определенный вид обработки данных, реализует генерацию стандартных отчетов.

Сравнение продуктов DM

SAS — Statistical Analysis System - большая и сложная система для статистической обработки данных (есть бесплатное web-приложение SAS University Edition с лицензией на год).

Достоинства:

- гибкий интерфейс обмена данными (интеграции);
- наличие инструментария для работы с кластерами (распределенными системами);
- быстрота расчетов на громадных массивах данных.

Недостатки:

- примитивный язык написания скриптов SAS macro;
- сложность поддержки уже написанных скриптов;
- дороговизна лицензий на полноценную версию.

Сравнение программных продуктов для анализа данных: R, MATLAB, SciPy, MS Excel, SAS, SPSS, Stata http://www.mbureau.ru/blog/sravnenie-programmnyh-produktov-dlya-analiza-dannyh-r-matlab-scipy-ms-excel-sas-spss-stata#sas

Презентация курсов по SAS

http://www.sas.com/offices/europe/russia/training/Navigation_courses_related_to_programming.pdf

Сравнение продуктов DM

Стоимость некоторых курсов по SAS

Код курса	Название курса	Длит. (дни)	Стоимость, за 1 чел., руб без НДС	2017 год					
				январь	февраль	март	апрель	май	июнь
SAS FOUNDATION									
PRG1_rus @	Программирование на языке SAS. Часть 1: Основы	3	64500	30.01 01.02		13.03 15.03	19.04 21.04	31.05 02.06	
PRG2_rus @	Программирование на языке SAS. Часть2. Методы обработки данных	3	64500		06.02 08.02	20.03 22.03	24.04 26.04		13.06 15.06
PRG3_rus @	Программирование на языке SAS. Часть 3. Расширенные методы программирования и приемы повышения производительности	3	64500		15.02 17.02	29.03 31.03		10.05 12.05	26.06 28.06
DS2E	DS2 Programming: Essentials	1,5	32250				06.04 07.04		08.06 09.06
MAC1 @	SAS Macro Language 1: Essentials	2	43000	16.01 17.01	13.02 14.02	27.03 28.03			19.06 20.06
MAC2	SAS Macro Language 2: Advanced Techniques	2	43000		20.02 21.02		10.04 11.04		
SQL1 @	SAS SQL1: Essentials	2	43000			06.03 07.03		29.05 30.05	13.06 14.06
EG1_rus @	SAS Enterprise Guide. Часть 1. Составление запросов и отчетов		43000	12.01 13.01		01.03 02.03 23.03 24.03		18.05 19.05	15.06 16.06
EG2_rus @	SAS Enterprise Guide. Часть 2. Составление расширенных запросов и отчетов	2	43000	23.01 24.01		09.03 10.03	06.04 07.04	29.05 30.05	22.06 23.06

Презентация курсов по SAS

http://www.sas.com/ru_ru/training/home.html

Сравнение продуктов DM

SPSS Statistics — компьютерная программа для статистической обработки данных для проведения прикладных исследований в социальных науках. Комментарий пользователя: «*По моим впечатлениями SPSS используют люди, которые хотят выполнять общепринятый статистический анализ наиболее простым путем*».

Достоинства:

- удобный графический интерфейс;
- ориентация на социальныхе науки.

Недостатки:

- дороговизна лицензий;
- отсутствие гибкости в расчетах.

Сравнение программных продуктов для анализа данных: R, MATLAB, SciPy, MS Excel, SAS, SPSS, Stata http://www.mbureau.ru/blog/sravnenie-programmnyh-produktov-dlya-analiza-dannyh-r-matlab-scipy-ms-excel-sas-spss-stata#sas

Сравнение продуктов DM

Название	Достоинства	Недостатки	Open source?	Типичные области применения
R	Поддержка библиотек, визуализация, гибкость, открытый код	Сложность обучения	Да	Финансы, Статистика
Matlab	1. «элегантная поддержка матриц»; 2. удобный графический интерфейс; 3. простота в работе.	1. Дороговизна лиц-й; 2. неполная поддержка статистич. функций; 3. довольно запутанная интеграция с JAVA и C++ приложениями	Нет	Инженерная
SciPy/NumPy /Matplotlib	1. хорошая интеграция с языком Python; 2. высокая производитель-ность матем. операций; 3. наличие готовых средств для визуальной отладки; 4. простота освоения.	незрелость решения	Да	Инженерная
Excel	Легкость работы, визуализация.	• отсутствие какой-либо гибкости; • ограниченный набор функций для анализа данных;	Нет	Бизнес
SAS	1. гибкий интерфейс обмена данными; 2. наличие инструментария для работы с кластерами (распредел. системами); 3. быстрота расчетов на громадных массивах данных.	4. примитивный язык скриптов SAS macro; 5. сложность поддержки написанных скриптов; 6. дороговизна лицензий; 7. сложность освоения.	Нет	Бизнес; Правительство
Stata	Легкий статистический анализ	Довольно узкая специализация	Нет	наука
SPSS	Как Stata но дороже и хуже			

Comparison of data analysis packages: R, Matlab, SciPy, Excel, SAS, SPSS, Stata
http://brenocon.com/blog/2009/02/comparison-of-data-analysis-packages-r-matlab-scipy-excel-sas-spss-stata/

R: общие сведения

- Язык программирования с динамической типизацией, разработанный на основе языка S, предназначенный для статистической обработки данных и работы с графикой.
- Создатели: **Р**осс Айхэк и **Р**оберт Джентлмен из Оклендского университета, 1993
- На язык оказали влияние: *Scheme*, *S*

Среда разработки

Кроме возможности использования различных текстовых редакторов, есть и другие возможности:

- **Emacs:** Это кросс-платформенный редактор, который выполняет все, что любой текстовый редактор и больше. В нем существуют плагины, для того, чтобы сделать Ваше R программирование намного проще, но инструмент не слишком прост в изучении.

- **RStudio:** . RStudio не является текстовым редактором, но является свободным кросс-платформенным IDE, которое предоставляет мощные комплексные средства, специально предназначенные для R программистов.

*www.rochester.edu/college/psc/thestarlab/help/rcourse/**R**-Course.pdf*

R: общие сведения

Про EMACS есть такая поговорка: о чем бы вы ни подумали, это уже написано для EMACS. Для него разработан пакет дополнения ESS. ESS — Emacs Speaks Statistics.

ESS - пакет дополнения для текстовых редакторов Emacs, таких как GNU Emacs и XEmacs . Он предназначен для поддержки редактирования сценариев и взаимодействие с различными программами статистического анализа, таких как R, S-Plus, SAS, Stata и OpenBUGS/JAGS.

В ESS действительно есть все что нужно: редактор кода с подсветкой, терминал с автозавершением, интегрированная справочная система и показ картинок.

Взять можно здесь: http://ess.r-project.org/index.php?Section=download

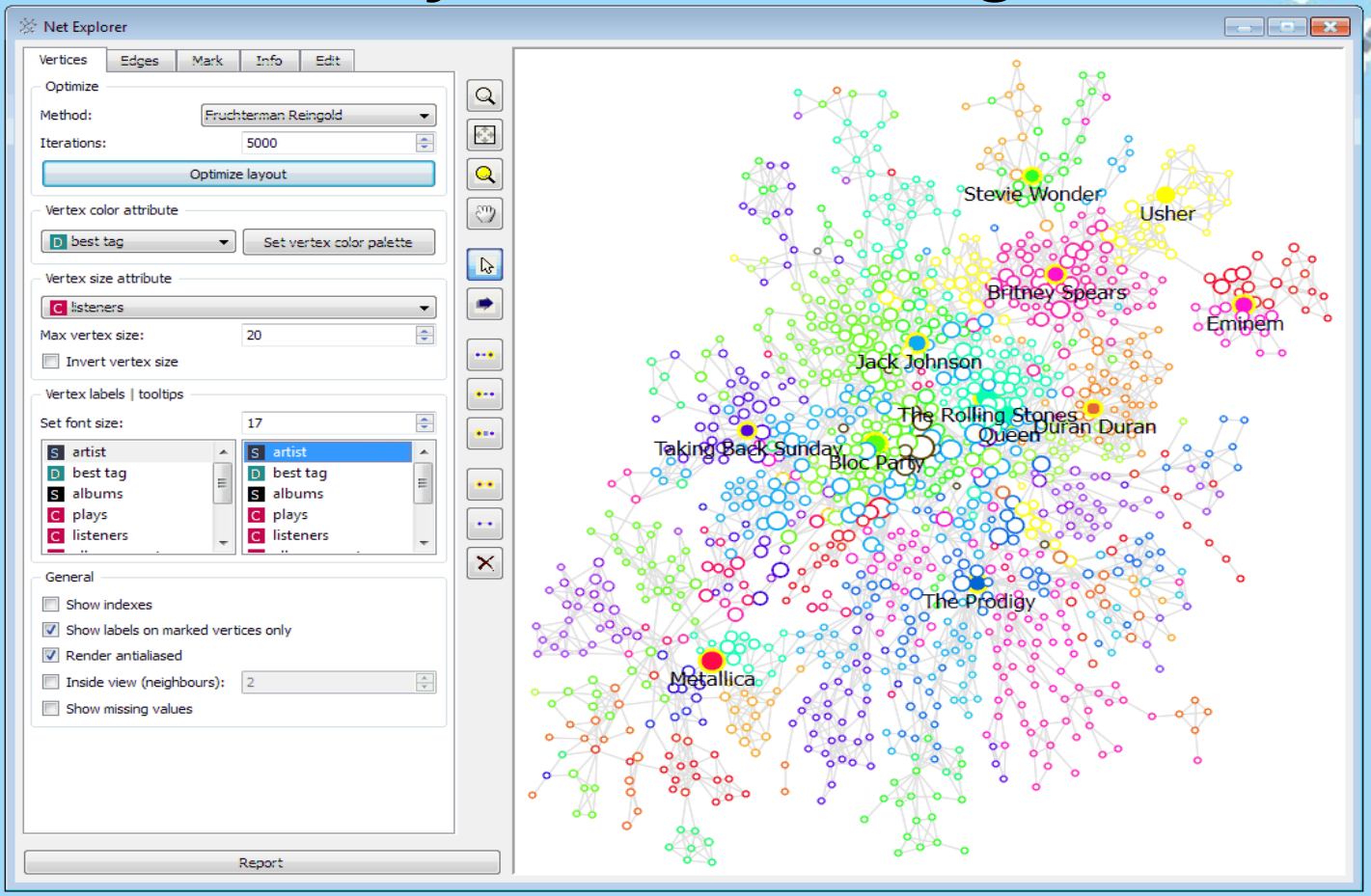
или здесь: http://www.emacs.uniyar.ac.ru/

Python

- ✓ Компания Google использует Python в своей поисковой системе и оплачивает труд создателя Python — Гвидо ван Россума
- ✓ Такие компании, как Intel, Cisco, Hewlett-Packard, Seagate, Qualcomm и IBM, используют Python для тестирования аппаратного обеспечения
- ✓ Служба коллективного использования видеоматериалов YouTube в значительной степени реализована на Python
- ✓ NASA использует Python для шифрования и анализа разведданных
- ✓ Компании JPMorgan Chase, UBS, Getco и Citadel применяют Python для прогнозирования финансового рынка
- ✓ Популярная программа BitTorrent для обмена файлами в пиринговых сетях написана на языке Python
- ✓ Популярный веб-фреймворк App Engine от компании Google использует Python в качестве прикладного языка программирования
- ✓ NASA, Los Alamos, JPL и Fermilab используют Python для научных вычислений.

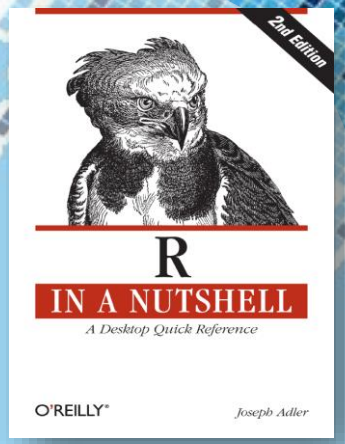
Основные библиотеки, которые применяются для анализа данных при помощи Python: **NumPy**, **SciPy**, **Matplotlib** и **Pandas**.

Python & Orange



Orange – это open source система для анализа и визуализации данных, написанная на языке Python. Она представляет широкие возможности для анализа данных в ней непосредственно, а также для ее использования в создании производных программ для анализа данных. Система состоит из основного функционального ядра и графического интерфейса Canvas к нему.

Нам понадобится



- Книга «R in a Nutshell» Джозефа Адлера
- R версии 3.3.2 (установить перед RStudio)

http://cran.r-project.org/bin/windows/base/

- Среда исполнения RStudio версии 0.98.1091- http://www.rstudio.com/products/rstudio/download/

- Марк Лутц. Изучаем Python, 4-е издание.
- Python -3.4.2. msi http://pythonworld.ru/osnovy/skachat-python.html
- Orange http://orange.biolab.si/download/

3 Структура R, конструкции

Структура языка

- Программы на R состоят из *выражений,*
- *Выражения* состоят из *объектов* и *функций,*
- *Объекты* (векторы, списки, функции),
- Имена переменных называются *символами.*
- *Среда* – это набор символов.
- *Функции* – это объекты, которые принимают на вход объекты (*аргументы*) и выдают объекты.

Структура языка

```
> x<-1; if (2<x) "go" else "stop"
  [1] "stop"
> c('e','x','p','a','s','o','f','t')
  [1] "e" "x" "p" "a" "s" "o" "f" "t"
> function (x) { x+1 }
  function (x) { x+1 }
```

- Когда вы присваиваете переменной объект, вы на самом деле присваиваете объект этому символу в рамках текущей среды (контекста)
- Единичка в квадратных скобках означает номер элемента вектора. Отсюда сразу два логических вывода:

1)R результат любой операции с числами трактует как вектор единичной длины. **Скаляров в R вообще нет**;

2)Элементы векторов нумеруются с единицы, а не с нуля, как во многих языках программирования.

Функции

- Каждый оператор может быть записан в виде *функции*
- Функциональная запись

```
>x<-c("Аз", "Буки", "Веди", "Глаголь")
>x[2]<-"Есмь"
>`[<-`(x,2,"Есмь") #Эквив. предыд. команде
>sign<-"Ъ"
>`<-`(sign,"Ъ") # Эквив. предыд. команде
> sign
[1] "Ъ"
>`+`(1799,2014)
[1] 3715
>1799+2014
[1] 3715
```

Функциональная запись присваивания и сложения соответственно. Даже if может быть записан в функциональном виде и обрабатываться как функция

Поведение объектов

- Объекты в основном неизменяемые (почти всегда выполняется копирование объекта)
- Объекты копируются в операторах присваивания
- Функции имеют свой контекст

```
> x<-3; y<-x # y=3
> x<-2
> y
[1] 3
> s<-"Яблоко"
> f <- function (x) { x = "Банан" }
> z<-f(s); z
[1] "Банан"
> s
[1] "Яблоко"
```

Объекты

- Всё – объекты
  - Символы
  - Имена функций (символы)
  - Функции (тела функций)
- Символы не привязаны к типам
- Связка символ-объект сохраняется в объекте «среда» (environment) или во всей программе.

```
> f<-2
> f(2)
Ошибка: не могу найти функцию "f"
>f <- function (x) { x+1 }
>f
function (x) { x+1 }
>f(2)
[1] 3
```

Выражения

- Выражение – это запись вида { exp1; exp2; exp3 }
- В качестве возвращаемого значения используется последнее выражение в фигурных скобках.

```
>{ f<-2; x<-c(1,2); f(2) }
Ошибка: не могу найти функцию "f"
>f <- function (x) { x=x+1; x+1 }
>f(10)
[1] 12
```

Основные функции: *c*

- c (combine) – позволяет создать вектор объектов

В новый вектор объединяются 2 других.

- Доступ к элементам вектора через [] (`[` - 2-х местный оператор)
- Изменение вектора через []<- (`[<-` - 3 ный оператор)

```
>c(1,2,3,5)
[1] 1 2 3 5
>x<-c(c(1,2,3),c(5,7)); x
[1] 1 2 3 5 7
>x[2]
[1] 2
>x(2)
Ошибка: не могу найти функцию "x“
>`[`(x,2)
[1] 2
>x[2]<-1; x
[1] 1 1 3 5 7
```

Специальные значения

- NA – пробелы (“Not available”)
  - Когда выходим за пределы массивов
  - При расширении массивов промежутки заполняются NA
- Inf-Inf – числа за пределами
- NaN – не число (“Not a number”)
  - Неопределённость, деление на 0
- NULL – объект, показывающий отсутствующий аргумент

```
> x<-c("М", "и", "р"); x[4]
[1] NA
>length(x)<-7; x
[1] "М" "и" "р" NA  NA  NA  NA
>factorial(1000)
[1] Inf
Предупреждение
In factorial(1000) : значение в
'gammafn' -- за пределами
>0/0; Inf-Inf
[1] NaN
[1] NaN
```

Приведение типов

- Всегда пытается привести:
  - к той *функции*, которая соответствует типам аргументов
  - *объект* к более общему типу

```
> x <- c(1:5)
> x
[1] 1 2 3 4 5
> typeof(x)
[1] "integer"
> class(x)
[1] "numeric"
```

```
> x[2] <- "Нуль"
> x
[1] "1" "Нуль"
"3" "4" "5"
> typeof(x)
[1] "character"
> class(x)
[1] "character"
```

Приведение типов: правила

- Логические значения приводятся к числам: `TRUE` в 1, `FALSE` в 0.
- Значения приводятся к наиболее простому типу, необходимому для представления всей информации.
- Порядок приведения: `logical` > `integer` > `numeric` > `complex` > `character` > `list` (списки – это наиболее общий вид).
- Объекты `raw` (загруженные байтовые массивы) не приводятся к другим типам.
- Свойства объектов теряются при приведении от одного типа к другому.

4 Синтаксис R

Синтаксис

1. Константы (векторы: числовые, буквенные; символы)
2. Операторы (приоритет операций, присвоение)
3. Выражения
4. Управляющие структуры (условный оператор; цикл)
5. Структуры данных (индексы: вектор чисел, вектор логических значений, имена)

Векторы: числовые (numeric)

- Числа, с плавающей точкой и шестнадцатеричные
- По умолчанию числовые векторы имеют тип `double`
- `a:b` – диапазон `integer`

- Комплексные числа `a+bi`
- Предельная точность чисел `2^1023`
- Предельный размер `2^1024`

```
> c(1.4142135, 2^1023, 0x010)
[1] 1.414214e+00 8.988466e+307
1.600000e+01
>typeof(1) # double
[1] "double"
>typeof(1:1) # integer
[1] "integer"
>0-1i * -1+1i
[1] 0+2i
>2^1023-1 == 2^1023
[1] TRUE
```

Векторы: буквенные (character)

- Кавычки (одиночная ', двойная ") эквиватентны.
- c – combine – тоже вектор

```
>"слово"; 'тоже слово'; "'однако'"
[1] "слово"
[1] "тоже слово"
[1] "'однако'"
>"\"то же\""=='"то же"'
[1] TRUE
>'\'кавычки\''=="'кавычки'"
[1] TRUE
>c("Ночь","улица","фонарь","аптека")
[1] "Ночь"   "улица"  "фонарь"
"аптека"
```

Символы

- Символы - это объекты
- Начинаются с буквы
- Могут содержать точки (. –это элемент имени), другие буквы, подчерк (_), числа
- В обратных кавычках `` могут содержать любые символы
- Любой символ может быть переопределён, кроме

```
if, else, repeat, while,
function, for, in, next, break,
TRUE, FALSE, NULL, Inf, NaN, NA,
NA_integer_, NA_real_,
NA_complex_, NA_character_,
..., ..1, ..2, ..3, ..4, ..5,
..6, ..7, ..8, ..9
```

```
> x_<-1; x1<-2; x1.1<-3; X1<-4
> sum(x_,x1,x1.1,X1)
[1] 10
>`<-`
.Primitive("<-")
>`2+2==5`<-TRUE
>`2+2==5`
[1] TRUE
>`TRUE`<-FALSE; `TRUE`
[1] FALSE
>c<-"Это не комбайн, а комбинат"
>c; c("Это","как","посмотреть!")
[1] "Это не комбайн, а комбинат"
[1] "Это" "как" "посмотреть!"
```

Назначили новый символ (переменную)

Операторы

- Оператор – это унарная или бинарная функция
- Особая запись:
  - `*унарный оператор*`x
  - x `*бинарный оператор*` y
- Можно определить *%свой_оператор%*
- <-, [, +, - - бинарные операторы
- ?, -, + - унарные операторы
- Выполнение функции - тоже оператор

```
> x <- -c(TRUE,FALSE); x
[1] -1 0
>c(1 + 2, 3 * 4, 5 %% 6, 3 ^ 7, 9 %/% 4)
[1]    3   12    5 2187    2
>`%++%`<-function(a,b) { a + a + b + b }
>1 %++% 2
[1] 6
>`[`(x,1)
[1] -1
>sum(1:50) # sum-1ый аргумент, 1:50-2ой
[1] 1275
```

%%-Остаток от деления
%/% - Целая часть

Функция определила оператор ++

Приоритет операций

- Вызов функции, группировка выражений
- Индексы и обращения
- Арифметические
- Сравнение
- Формулы
- Присвоение
- Помощь

```
> ?base::Syntax
```

:: :::	access variables in a namespace
$ @	component / slot extraction
[[[	indexing
^	exponentiation (right to left)
- +	unary minus and plus
:	sequence operator
%any%	special operators
* /	multiply, divide
+ -	(binary) add, subtract
< > <= >= == !=	ordering and comparison
!	negation
& &&	and
| ||	or
~	as in formulae
-> ->>	rightwards assignment
<- <<-	assignment (right to left)
=	assignment (right to left)
?	help (unary and binary)

Контроль созданных объектов

Чтобы убедиться, что переменная была создана (R сохраняет их в активной памяти – в рабочем окружении) используйте ***ls()*** функцию.

```
> a=b=c=1
> ls ( )
[1] " a "  " b "  "c"
```

Если хотите удалить объект из памяти, используйте функцию ***rm()***. Будьте осторожны – эта команда удаляет переменную навсегда.

```
> rm(c)
> ls ( )
[1] " a "  " b "
```

Если хотите удалить все переменные, воспользуйтесь командой:

```
> rm( list = ls ( ))
```

Присваивание

- Присваивается *символу объект*
- Присвоить можно операторами `<-`, `->`, `=`
- Присваивание с модификацией объекта: `f(x,…)<-`
- Возможно переопределение присваивания:
  - `` `<-` ``
  - `` `f<-` `` - тоже, что и `x <- f(x,y)`

```
> x <- "Учиться"
> "Учиться" -> y
> z = "Учиться"
> s<-c(x,y,z); s
[1] "Учиться" "Учиться" "Учиться"
> s[2]<-"Работать"; s
[1] "Учиться"  "Работать" "Учиться"
> length(s)<-6; s # изменение длины вектора
[1] "Учиться" "Работать" "Учиться" NA NA NA
> f<-function (x) { x-1 }
> `f<-` <- function (x, value) { x-value }
> f(length(s)); f(length(s))<-3; s
[1] 5
[1] "Учиться" "Работать" "Учиться"
```

Группировка выражений

- Перевод строки
- ; - разделитель выражений
- Скобки:
  - Круглые (*expr*)
  - Фигурные { *e_first*; …; *e_last* }
    - *Как функция `{`*
    - *Возвращает только e_last*
    - *Вне функций не создаёт новый контекст*

```
> x <- 1
> y <- 2
> z <- 3
> x <- y; z <- x; f<-function(x){x}
> (43) == f(43)
[1] TRUE
> {x; y; z}
[1] 2 # выводится только z
```

Специальные операторы: условный

- Условный оператор
  - Не векторный (проверяется только первое условие вектора)
  - `ifelse` - векторный

```
> if (1:50>2) "Да" else "Нет"
[1] "Нет"
Предупреждение
In if (1:50 > 2) "Да" else "Нет" :
  длина условия > 1, будет использован только первый элемент
>ifelse(c(T,T,F),c(1:3),c(4:6))
[1] 1 2 6
```

Специальные операторы: выбор

- `switch` оператор
  - не векторный
  - первый аргумент буквенный
  - безымянный – по умолчанию
  - после = следует *выражение*

```
> switch("Раз","Раз"=1,"Два"={1+1},Inf)
[1] 1
> switch("Два","Раз"=1,"Два"={1+1},Inf)
[1] 2
> switch("Нуль","Раз"=1,"Два"={1+1},Inf)
[1] Inf
```

Специальные операторы: циклы

- `repeat` *expression*
- `while (`*condition*`)` *expression*
- `for (`*var* `in` *list*`)` *expression*

- выход – break
- следующая итерация - next

```
> p<-c("З","А","Ч","Е","Т"); i<-1
```

```
> repeat {
    print(
      c(i,p[i]))
    if(i==3)
      break
    else i<-i+1
  }
> i
[1] "1" "З"
[1] "2" "А"
[1] "3" "Ч"
[1] 3
```

```
> while(i <= 3) {
    print(
      c(i,p[i]))
    i<-i+1
  }
> i

[1] "1" "З"
[1] "2" "А"
[1] "3" "Ч"
[1] 3
```

```
> for(i in 1:3)
  print(
    c(i,p[i]))
> i



[1] "1" "З"
[1] "2" "А"
[1] "3" "Ч"
[1] 3
```

x<- readline("Введите номер. ") # ввод с клавиатуры

Доступ к структурам данных

Запись	Объекты	Описание
`x[i]`	Векторы, списки	Возвращает объект с индексом ***i*** из ***x***, ***i*** может быть ***числовым*** вектором, ***буквенным*** вектором (в т.ч. ***имён объектов***) или ***логическим*** вектором. Не допускает частичного совпадения. В списках возвращает список. В векторах --- вектор.
`x[[i]]`	Векторы, списки	Возвращает единственный элемент `x`, соответствующий `i`. `i` может быть как числовым, так и буквенным вектором длины 1. Допускает частичное соответствие (с опцией `exact=FALSE`).
`x$name`	Объект	Возвращает объект с именем ***name*** из объекта ***x***.
`x@name`	Объект	Возвращает объект, сохранённый в объекте ***x*** в слоте с именем ***name***.

5 Типы данных R

1. Примитивные типы
2. Векторы, списки, матрицы, массивы
3. Таблицы "объект-свойство"

Примитивные типы

- Базовые векторы
- Составные объекты
- Специальные объекты
- Объекты языка R
- Функции
- Внутренние объекты
- Объекты байт-кода

```
> c('куб','г','у')
  [1] "куб" "г" "у"
> typeof(`if`)
[1] "special"
> typeof(quote(if(1>2) "что-то не так"))
  [1] "language"
```

Типы данных

- Язык R принадлежит к семейству так называемых высокоуровневых объектно-ориентированных языков программирования. Для простоты можно называть объектами все, что мы создаем в ходе работы с R. Их выделяют два основных типа:
- **Объекты, предназначенные для хранения данных** («*data objects*») – это векторы, матрицы и массивы, списки, факторы, таблицы данных;
- **Функции** («*function objects*») – это поименованные программы, предназначенные для выполнения определенных действий над другими объектами.

Типы данных

Объектный тип	Описание	Примеры
integer	Создаётся из последовательностей. Может быть получен с помощью приведения типов `Integer()`.	`1:1; integer(1)`
double	Числа с плавающей запятой. 8 байт. По умолчанию числовые значения представляются в этом виде. Приведение с помощью `double()`.	`1; 2.1; double(1)`
complex	Комплексные числа. Записываются как `a+bi`, "плюс" нельзя опускать, даже если `a` или `b` равны `0`.	`0+0i; 1+0i; 0.1+0.9i`
character	Строка из символов.	`"Строка из символов"`
logical	Булевы значения `TRUE` и `FALSE`.	`TRUE; FALSE`
raw	Вектор из байтов типа `raw`. Как правило представляет объекты загруженные извне.	`raw(1)` `charToRaw("Сырой")`

Векторы

- Состоят из однотипных объектов (атомарные векторы), но могут быть и «generic» - общие
- `recursive=FALSE` – по умолчанию вектор приводится к списку

`recursive=TRUE` – распаковывает списки в вектор

`length<-` изменяет размер вектора

```
> str(x<-c(1:1, 2.1, "3"))
 chr [1:3] "1" "2.1" "3"
>x<-c(1:1, 2, 3, c(4,5))
[1] 1 2 3 4 5
>c(1,list(5)) # вектор приводится к списку
[[1]]
[1] 1

[[2]]
[1] 5 # список
# вектор не приводится к списку:
>c(1,list(5), recursive=TRUE)
[1] 1 1 # вектор
>length(x)<-2; x
[1] "1" "2.1"
```

Векторы

Имена

Вы можете именовать вектор 3-мя способами:

- При создании: `x <- c(a = 1, b = 2, c = 3).`
- Изменяя существующий вектор: `x <- 1:3; names(x) <- c("a", "b", "c").`

  или: `x <- 1:3; names(x)[[1]] <- c("a").`

```
> x
   a <NA> <NA>
   1    2    3
> names(x)
[1] "a" NA  NA
```

- Создавая модифицированную копию вектора:

```
> x <- setNames(1:3, c("a", "b", "c")).
```

```
> x <- setNames(1:3, c("a", "b", "c"))
> x
a b c
1 2 3
```

Имена не обязаны быть уникальными. Однако при замене содержимого элементов вектора (subsetting) удобнее использовать уникальные имена.

Векторы

Выборка элементов из вектора:

```
> xA [1] 100 105 110 115 120 125 130 135 140 145 150 155 160
[14] 165 170 175 180 185 190 195 200
```

```
> which(xA>160);
[1] 14 15 16 17 18 19 20 21
> a<-xA[which(xA>160)];
> a
[1] 165 170 175 180 185 190 195 200
> a<-which(xA>160);
> a
[1] 14 15 16 17 18 19 20 21
```

Списки

```
> n = c(2, 3, 5)
> s = c("aa", "bb", "cc", "dd")
> b = c(TRUE, FALSE, TRUE, FALSE)
> x = list(n, s, b, 3)
> x
[[1]]
[1] 2 3 5
[[2]]
[1] "aa" "bb" "cc" "dd"
[[3]]
[1] TRUE FALSE TRUE FALSE
[[4]]
[1] 3
```

```
> x[[1]] #это вектор – элемент списка
[1] 2 3 5
> x[2]
[[1]] #это вектор
[1] "aa" "bb" "cc" "dd" "ee"
> x[c(1, 2)] #это вектор
[[1]]
[1] 2 3 5
[[2]]
[1] "aa" "bb" "cc" "dd" "ee"
```

Массивы

```
> a <-array(c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12), dim=c(3, 4))
> a
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

строки, столбцы

…

```
> a[10]
[1] 10
> a[1, 2]
[1] 4
```

```
> a[, 1] #первый столбец
[1] 1 2 3
> a[3, ] #третья строка
[1] 3 6 9 12
```

Массивы

```
>x <- array(1:9, dim = c(3, 3))
>i <- array(c(2, 1, 1, 2), c(2, 2))
>x[i] <- 0
>x
     [,1] [,2] [,3]
[1,]    1    0    7
[2,]    0    5    8
[3,]    3    6    9
```

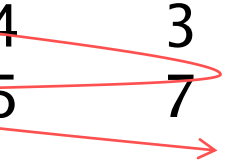
dim можно опустить

К массиву можно обратиться по индексу, который тоже является массивом

В массив ***x*** по индексу ***i*** присваиваем 0, т.е. элементы x с индексами 2, 1 и 1,2 станут равны 0

Матрицы

```
>A = matrix(c(2, 4, 3, 1, 5, 7), nrow=2, ncol=3,
 byrow = TRUE)
>A
     [,1] [,2] [,3]
[1,]    2    4    3
[2,]    1    5    7
```



По умолчанию выстраивает по колонкам

```
> m<-matrix(data=(101:112),nrow=3, ncol=4)
> m
     [,1] [,2] [,3] [,4]
[1,]  101  104  107  110
[2,]  102  105  108  111
[3,]  103  106  109  112
> m[m>105]
[1] 106 107 108 109 110 111 112
> m[1:2,]
     [,1] [,2] [,3] [,4]
[1,]  101  104  107  110
[2,]  102  105  108  111
```

Таблица "объект-свойство " data frame

```
>n = c(2, 3, 5)
>s = c("aa", "bb", "cc")
>b = c(TRUE, FALSE, TRUE)
>df = data.frame(n, s, b)
>df
  n  s     b
1 2  aa  TRUE
2 3  bb FALSE
3 5  cc  TRUE
```

В R есть некоторые предопределенные data.frame, как показано на следующем слайде.

Таблица "объект-свойство"

```
> head(mtcars)
                   mpg cyl disp …
Mazda RX4         21.0   6  160
Mazda RX4 Wag     21.0   6  160
Datsun 710        22.8   4  108
Hornet 4 Drive    21.4   6  258
Hornet Sportabout 18.7   8  360
Valiant           18.1   6  225
…
```

Работа со временем

```
> t <- Sys.time()
> timeStamp <- strftime(t,"%Y-%m-%d %H:%M:%S")
> timestamp
> [1]  "2014-01-23 14:47:43"
> typeof(timeStamp)
> [1] "character"
```

Команда *strftime* используется для извлечения времени и конвертации в строку.

http://www.cyclismo.org/tutorial/R/time.html

Работа со временем

```
> t1<- strptime(Sys.time(),"%Y-%m-%d %H:%M:%S")
> t1 [1] "2017-02-23 13:05:58 MSK"
> t2<- strptime(Sys.time(),"%Y-%m-%d %H:%M:%S")
> t2 [1] "2017-02-23 13:06:41 MSK"
> t2-t1
> Time difference of 43 secs
> t3<- strptime(Sys.time(),"%Y-%m-%d %H:%M:%S")
➢ t3-t1
➢ Time difference of 1.416667 mins
> as.double(t2-t1); as.double(t3-t1)
>[1] 43
>[1] 1.416667
```

Команда *strptime()* command используется для перевода строки в форму с которой R может производить вычисления.

Таблица "объект-свойство"

Объект	Возможные типы данных	Использование в объекте нескольких типов данных
vector	числовой, символьный, комплексный, логический	Нет
factor	числовой, символьный	Нет
array	числовой, символьный, комплексный, логический	Нет
matrix	числовой, символьный, комплексный, логический	Нет
data.frame	числовой, символьный, комплексный, логический	Да
Ts	числовой, символьный, комплексный, логический	Да
List	числовой, символьный, комплексный, логический, функция, выражение, формула	Да

Выводы

- Во-первых, для R есть замечательная среда разработки — RStudio. Как и сам R доступна для всех трёх основных платформ, плюс есть серверная версия для работы через браузер. Преимущества — есть редактор файлов с подсветкой синтаксиса, список созданных объектов, графики и документация открываются в отдельной области, ну и в целом поприятней голой консоли.
- Во-вторых, R во многом похож на Scheme, только с JavaScript-подобным синтаксисом и без макросов. Зато с такой же системой environment-ов, с такой же нежёсткой типизацией, символами (symbols) и выражениями (expressions) и т.д. Последние 2 фичи, кстати, позволяют решать многие из задач, решаемых в Лиспах за счёт макросов.

Выводы

- В-третьих, сразу стоит понимать, что R — язык с историей, т.е. многие его «странности» объясняются именно историческими причинами. Например, в R есть сразу 2 системы объектно-ориентированного программирования — S3 и S4 (про их отличия был вопрос на StackOverflow). При этом нельзя сказать, что одна система является устаревшей — отнюдь, обе используются довольно часто. Историческими же причинами объясняется и отсутствие единой конвенции кода: изначально в R было принято писать всё маленькими буквами и разделять слова точками ("." в R — вполне законный символ в имени, так же, как "_" в Си, например). Но потом пришла S3, в которой точка использовалась при диспетчеризации методов объектов, и имена функций с точками стали не очень очевидными.

Выводы

- В-четвёртых, типизация в R может поначалу сломать мозг. В R у каждого объекта есть как минимум 2 атрибута типа — mode и class (их можно узнать с помощью одноимённых функций — mode() и class()). Грубо говоря, class — это тип самого объекта (например, matrix или table), а mode — это тип примитивов, хранимых в объекте (например, numeric или character). При этом numeric, integer, character и т.д. — это не число, целое и чар, как это принято в других языках, а вектор этих значений. И даже `5` — это не просто число, а вектор из одно элемента. В общем, что касается типов, в R всё очень необычно :)

Выводы

- В-пятых, в R обращение к данным производится посредством индексации. Например, к пятому элементу (привет, Мила Йовович) вектора `x` можно обратиться как `x[5]`. Но у вектора также может быть атрибут `names`, и тогда к тому же элементу можно обратиться, например, `x[«fifth_element»]`. Кроме оператора `[ ]` есть оператор `[[ ]]` (в основном используется для поиска по спискам — lists) и оператор `$`, который работает только для обращения через имя элемента, но и то не для всех типов. Так что не удивляйтесь, а внимательно читайте документацию.
- Кроме того, обратите внимание, что у любого объекта кроме основного значения может быть любое количество атрибутов. Например, class и mode — это просто атрибуты объекта (которые, кстати, можно перезаписывать). Часто можно встретить и такие атрибуты как dim, names/dimnames и др.

Литература

1. "J. Adler, R in a Nutshell, Second ed., 2012, 722 p.
2. Тренажер для освоения основ языка R: http://tryr.codeschool.com/ от O'Reilly
3. Язык R: из учебной лаборатории — в мир больших данных. Леонид Черняк. osp.ru/os/2012/04/13015768/
4. **R Tutorial.** http://www.cyclismo.org/tutorial/R

Вопросы к зачету

1. Дать краткую сравнительную характеристику инструментария для анализа данных.
2. Охарактеризовать конструкции языка R
3. Перечислить типы языка R, привести примеры.