Введение в язык программирования R

Тяпочкин Константин

Почему R?

- Open source
- Язык созданный статистиками для статистиков
- Большое количество стандартных методов статистического анализа в базовом пакете
- Более 4000 пакетов в репозиториях CRAN
- Огромное число других пакетов (например, bioconductor для биоинформатики)

История

- S основал John Chambers в 1976 году
- В 1988 году ядро было переписано на C
- В это же время появился S-PLUS — коммерческая версия
- В 1991 году начинается разработка языка R
- В 2000 году версия 1.0.0
- Сегодня актуальная версия 2.15.2

Полезные ссылки

- www.r-project.org
- www.rstudio.com
- www.rseek.org
- www.r-analytics.blogspot.ru/p/blog-page_06.html

Получение справки

- ?sum — страница справки
- help(sum) — то же
- ??'sum' — поиск по всем страницам
- help.search('sum') — то же
- Если ввести просто имя функции без скобок, R выведет ее исходный код
- Функция str кратко выводит информацию об объекте

Объекты

- Атомарные типы:
  - character
  - numeric
  - integer
  - complex
  - Logical
- Векторы
- Списки

Числа

- Обычно типа numeric
- Чтобы получить тип integer добавляем суффикс L
- Число Inf
- NaN

Присвоение

- x<-1; print(x)

  [1] 1
- x=1 (в основном используется при задании аргументов функций)
- x<-'sdf'

Векторы

- 1:10

  [1] 1 2 3 4 5 6 7 8 9 10
- c(1,4,15)

  [1] 1 4 15
- seq(0,10,2)

  [1] 0 2 4 6 8 10

Векторы

- c(0.5, 0.6) ## numeric
- c(TRUE, FALSE) ## logical
- c(T, F) ## logical
- c("a", "b", "c") ## character
- 9:29 ## integer
- c(1+0i, 2+4i) ## complex
- vector("numeric", length = 10)

Векторы

- Большинство функций работает с векторами.
- sum(c(1,2,5))

  [1] 8
- mean(0:50)

  [1] 25

Векторы

- Нумерация начинается с единицы!
- x<-3:10
- x[2:4]

  [1] 4 5 6
- x[c(T,F,T,F,F,T,F)]

  [1] 3 5 8 10
- x[-(1:2)]

  [1] 5 6 7 8 9 10

Матрицы

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

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

- dim(m)

```
[1] 2 3
```

Матрицы

- x <- 1:3; y <- 10:12
- cbind(x,y)

```
     x  y
[1,] 1 10
[2,] 2 11
[3,] 3 12
```

- rbind(x,y)

Матрицы

- m<-matrix(1:6, nrow = 2, ncol = 3)
- m[2,3]

  [1] 6
- m[2,]

  [1] 2 4 6
- m[1:2,1:2]

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

Списки

- x <- list(123, "a")
- x

  [[1]]

  [1] 1

  [[2]]

  [1] "a"
- X[[1]]

  [1] 123

Факторы

- x <- factor(c("yes", "yes", "no", "yes", "no"))
- x

  [1] yes yes no yes no

  Levels: no yes
- table(x)

  x

  no yes

  2 3

Data frames

- x <- data.frame(foo = 1:4, bar = c(T, T, F, F))
- x

  foo bar

  1 1 TRUE

  2 2 TRUE

  3 3 FALSE

  4 4 FALSE
- nrow(x)

  [1] 4
- ncol(x)

  [1] 2

Имена элементов

- x <- 1:3
- names(x)

  NULL
- names(x) <- c("foo", "bar", "norf")
- x

  foo bar norf

  1 2 3
- names(x)

  [1] "foo" "bar" "norf"
- x["foo"]

  foo

  1

Имена элементов

- x <- list(a = 1, b = 2)
- x

  $a

  [1] 1

  $b

  [1] 2
- x$a

  [1] 1

If

- if (x < 3) {
  y<-10
  } else {
  Y<-0
  }

- y<-if (x<3) {
  10
  } else {
  0
  }

For

- x <- c("a", "b", "c", "d")
- for(i in 1:4) {
  print(x[i])
  }
- for(i in seq_along(x)) {
  print(x[i])
  }
- for(letter in x) {
  print(letter)
  }
- for(i in 1:4) print(x[i])

- По возможности стоит избегать работы с циклами.
- Более правильным будет использование векторизованных функций и функций второго порядка (например apply)

Функции

- f <- function(<arguments>) {

  ## Do something interesting

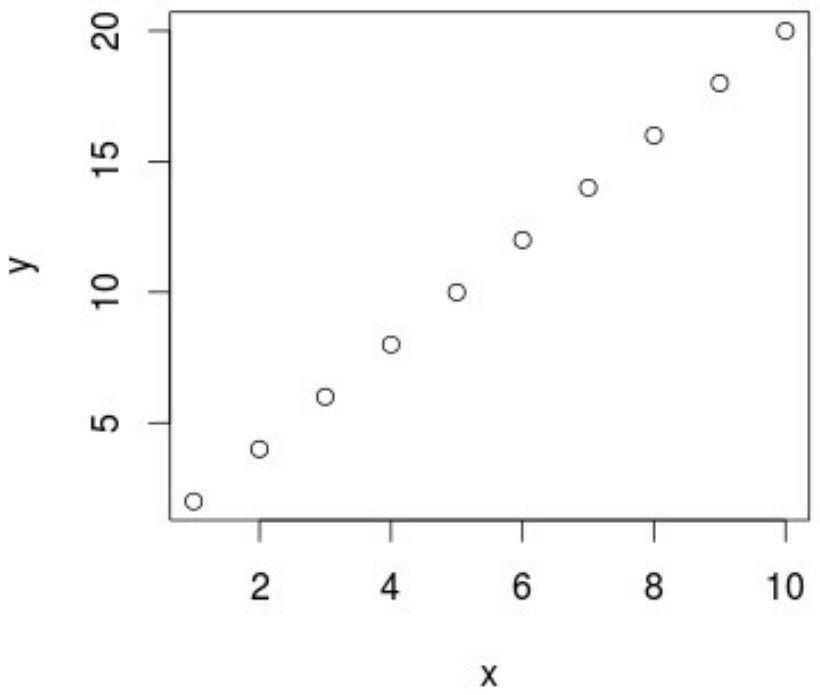
  }
- Значение можно возвращать либо через return(value) либо просто последним выражением функции
- f <- function(a, b = 1, c = 2) {return(a+b+c)}

Работа с данными

- read.table()
- read.csv()
- write.table()
- write.csv()

Графики

- x<-1:10
- y<-seq(2,20,2)
- plot(x,y)



Случайные величины

- set.seed() - выставление ГСЧ
- Функции для нормльного распределения
  - qnorm()
  - pnorm()
  - qnorm()
  - rnorm()

Пример: проверка нормальности

- set.seed(1)
- x<-rnorm(100)
- qqnorm(x)
- qqline(x)
- shapiro.test(x)

  Shapiro-Wilk normality test

  data: x

  W = 0.9956, p-value = 0.9876