Basics of R Programming

Yanchang Zhao
http://www.RDataMining.com

R and Data Mining Course
Beijing University of Posts and Telecommunications,
Beijing, China

July 2019
Quiz

▶ Have you used R before?
Quiz

- Have you used R before?

- Are you familiar with data mining and machine learning techniques and algorithms?
Quiz

» Have you used R before?

» Are you familiar with data mining and machine learning techniques and algorithms?

» Have you used R for data mining and analytics in your study/research/work?
Contents

Introduction to R

RStudio

Pipe Operations

Data Objects

Control Flow

Parallel Computing

Functions

Data Import and Export

Online Resources
What is R?

- R * is a free software environment for statistical computing and graphics.
- R can be easily extended with 14,000+ packages available on CRAN† (as of July 2019).
- Many other packages provided on Bioconductor‡, R-Forge§, GitHub¶, etc.
- R manuals on CRAN∥
  - An Introduction to R
  - The R Language Definition
  - R Data Import/Export
  - ...

---

*http://www.r-project.org/
†http://cran.r-project.org/
‡http://www.bioconductor.org/
§http://r-forge.r-project.org/
¶https://github.com/
∥http://cran.r-project.org/manuals.html
Why R?

▶ R is widely used in both academia and industry.
▶ R is one of the most popular tools for data science and analytics, ranked #1 from 2011 to 2016, but sadly overtaken by Python since 2017, :-( **.
▶ *The CRAN Task Views* †† provide collections of packages for different tasks.
  ▶ Machine learning & atistical learning
  ▶ Cluster analysis & finite mixture models
  ▶ Time series analysis
  ▶ Multivariate statistics
  ▶ Analysis of spatial data
  ▶ ...

** The KDnuggets polls on *Top Analytics, Data Science software*
†† http://cran.r-project.org/web/views/
Contents

Introduction to R

RStudio

Pipe Operations

Data Objects

Control Flow

Parallel Computing

Functions

Data Import and Export

Online Resources
RStudio

- An integrated development environment (IDE) for R
- Runs on various operating systems like Windows, Mac OS X and Linux
- Suggestion: always using an RStudio project, with subfolders
  - code: source code
  - data: raw data, cleaned data
  - figures: charts and graphs
  - docs: documents and reports
  - models: analytics models

‡‡https://www.rstudio.com/products/rstudio/
```r
a <- sample(10)
print(a)
plot(a, type="b")
```

R version 3.2.0 (2015-04-16) -- "Full of Ingredients"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY. You are welcome to redistribute it under certain conditions. Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors. Type 'contributors()' for more information and 'citation()' on how to cite R or R packages in publications.

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

- Run current line or selection: Ctrl + enter
- Comment / uncomment selection: Ctrl + Shift + C
- Clear console: Ctrl + L
- Reindent selection: Ctrl + I
Writing Reports and Papers

- Sweave + LaTeX: for academic publications
- beamer + LaTeX: for presentations
- knitr + R Markdown: generating reports and slides in HTML, PDF and WORD formats
- Notebooks: R notebook, Jupiter notebook
Contents

Introduction to R
RStudio
Pipe Operations
Data Objects
Control Flow
Parallel Computing
Functions
Data Import and Export
Online Resources
Pipe Operations

- Load library magrittr for pipe operations
- Avoid nested function calls
- Make code easy to understand
- Supported by dplyr and ggplot2

```r
library(magrittr)  ## for pipe operations
## traditional way
b <- fun3(fun2(fun1(a), b), d)
## the above can be rewritten to
b <- a %>% fun1() %>% fun2(b) %>% fun3(d)
```
Pipe Operations

- Load library magrittr for pipe operations
- Avoid nested function calls
- Make code easy to understand
- Supported by dplyr and ggplot2

```r
library(magrittr)  ## for pipe operations
## traditional way
b <- fun3(fun2(fun1(a), b), d)
## the above can be rewritten to
b <- a %>% fun1() %>% fun2(b) %>% fun3(d)
```

Quiz: Why not use 'c' in above example?
Contents

Introduction to R
RStudio
Pipe Operations
Data Objects
Control Flow
Parallel Computing
Functions
Data Import and Export
Online Resources
Data Types and Structures

- Data types
  - Integer
  - Numeric
  - Character
  - Factor
  - Logical
  - Date

- Data structures
  - Vector
  - Matrix
  - Data frame
  - List
## integer vector

```r
x <- 1:10
pin(x)
```

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

## numeric vector, generated randomly from a uniform distribution

```r
y <- runif(5)
y
```

```r
## [1] 0.95724678 0.02629283 0.49250477 0.07112317 0.93636358
```

## character vector

```r
(z <- c("abc", "d", "ef", "g"))
```

```r
## [1] "abc" "d" "ef" "g"
```
Matrix

```r
## create a matrix with 4 rows, from a vector of 1:20
m <- matrix(1:20, nrow = 4, byrow = T)
m
## [1,] 1  2  3  4  5
## [2,] 6  7  8  9 10
## [3,] 11 12 13 14 15
## [4,] 16 17 18 19 20

## matrix subtraction
m - diag(nrow = 4, ncol = 5)
## [1,] 0  2  3  4  5
## [2,] 6  6  8  9 10
## [3,] 11 12 12 14 15
## [4,] 16 17 18 18 20
```
library(magrittr)
age <- c(45, 22, 61, 14, 37)
gender <- c("Female", "Male", "Male", "Female", "Male")
height <- c(1.68, 1.85, 1.8, 1.66, 1.72)
marrried <- c(T, F, T, F, F)
df <- data.frame(age, gender, height, married) %>% print()

## age gender height married
## 1 45 Female 1.68 TRUE
## 2 22 Male 1.85 FALSE
## 3 61 Male 1.80 TRUE
## 4 14 Female 1.66 FALSE
## 5 37 Male 1.72 FALSE

str(df)

#'data.frame': 5 obs. of 4 variables:
#$ age : num 45 22 61 14 37
#$ gender : Factor w/ 2 levels "Female","Male": 1 2 2 1 2
#$ height : num 1.68 1.85 1.8 1.66 1.72
#$ married: logi TRUE FALSE TRUE FALSE FALSE
Data Slicing

df$age
## [1] 45 22 61 14 37

df[, 1]
## [1] 45 22 61 14 37

df[1, ]
##   age gender height married
## 1  45 Female 1.68   TRUE

df[1, 1]
## [1] 45

df$gender[1]
## [1] Female
## Levels: Female Male
Data Subsetting and Sorting

```r
df %>% subset(gender == "Male")
## age gender height married
## 2 22 Male 1.85 FALSE
## 3 61 Male 1.80 TRUE
## 5 37 Male 1.72 FALSE

idx <- order(df$age) %>% print()
## [1] 4 2 5 1 3

df[idx, ]
## age gender height married
## 4 14 Female 1.66 FALSE
## 2 22 Male 1.85 FALSE
## 5 37 Male 1.72 FALSE
## 1 45 Female 1.68 TRUE
## 3 61 Male 1.80 TRUE
```
x <- 1:10
y <- c("abc", "d", "ef", "g")
ls <- list(x, y) %>% print()

## [[1]]
## [1]  1  2  3  4  5  6  7  8  9 10
##
## [[2]]
## [1] "abc" "d" "ef" "g"

## retrieve an element in a list
ls[[2]]

## [1] "abc" "d" "ef" "g"

ls[[2]][1]

## [1] "abc"
Character

```r
x <- c("apple", "orange", "pear", "banana")
## search for a pattern
grep(pattern = "an", x)
## [1] 2 4

## search for a pattern and return matched elements
grep(pattern = "an", x, value = T)
## [1] "orange" "banana"

## replace a pattern
gsub(pattern = "an", replacement = "**", x)
## [1] "apple" "or**ge" "pear" "b****a"
```
library(lubridate)
x <- ymd("2019-07-08")
class(x)
# [1] "Date"

year(x)
# [1] 2019

# month(x)
day(x)
# [1] 8

weekdays(x)
# [1] "Monday"

Date parsing functions: ymd(), ydm(), mdy(), myd(), dmy(), dym(), yq() in package lubridate
Contents

Introduction to R
RStudio
Pipe Operations
Data Objects
Control Flow
Parallel Computing
Functions
Data Import and Export
Online Resources
Conditional Control

- **if ...else ...**

```r
score <- 4
if (score >= 3) {
    print("pass")
} else {
    print("fail")
}
## [1] "pass"
```

- **ifelse()**

```r
score <- 1:5
ifelse(score >= 3, "pass", "fail")
## [1] "fail" "fail" "pass" "pass" "pass"
```
Loop Control

▶ for, while, repeat

▶ break, next

```r
for (i in 1:5) {
  print(i^2)
}
```

```r
## [1] 1
## [1] 4
## [1] 9
## [1] 16
## [1] 25
```
Apply Functions

- `apply()`: apply a function to margins of an array or matrix
- `lapply()`: apply a function to every item in a list or vector and return a list
- `sapply()`: similar to `lapply`, but return a vector or matrix
- `vapply()`: similar to `sapply`, but as a pre-specified type of return value
## for loop
x <- 1:10
y <- rep(NA, 10)
for (i in 1:length(x)) {
    y[i] <- log(x[i])
}
y
## apply a function (log) to every element of x
tmp <- lapply(x, log)
y <- do.call("c", tmp) %>% print()
## same as above
sapply(x, log)
Contents

Introduction to R
RStudio
Pipe Operations
Data Objects
Control Flow
Parallel Computing
Functions
Data Import and Export
Online Resources
Parallel Computing

```r
## on Linux or Mac machines
library(parallel)
n.cores <- detectCores() - 1 %>% print()
tmp <- mclapply(x, log, mc.cores=n.cores)
y <- do.call("c", tmp)

## on Windows machines
library(parallel)
## set up cluster
cluster <- makeCluster(n.cores)
## run jobs in parallel
tmp <- parLapply(cluster, x, log)
## stop cluster
stopCluster(cluster)
# collect results
y <- do.call("c", tmp)
```
Parallel Computing (cont.)

On Windows machines, libraries and global variables used by a function to run in parallel have to be explicitely exported to all nodes.

```r
## on Windows machines
library(parallel)
## set up cluster
cluster <- makeCluster(n.cores)
## load required libraries, if any, on all nodes
tmp <- clusterEvalQ(cluster, library(igraph))
## export required variables, if any, to all nodes
clusterExport(cluster, "myvar")
## run jobs in parallel
tmp <- parLapply(cluster, x, myfunc)
## stop cluster
stopCluster(cluster)
# collect results
y <- do.call("c", tmp)
```
On Windows machines, libraries and global variables used by a function to run in parallel have to be explicitly exported to all nodes.

```r
## on Windows machines
library(parallel)
## set up cluster
cluster <- makeCluster(n.cores)
## load required libraries, if any, on all nodes
tmp <- clusterEvalQ(cluster, library(igraph))
## export required variables, if any, to all nodes
clusterExport(cluster, "myvar")
## run jobs in parallel
tmp <- parLapply(cluster, x, myfunc)
## stop cluster
stopCluster(cluster)
# collect results
y <- do.call("c", tmp)
```
Contents

Introduction to R
RStudio
Pipe Operations
Data Objects
Control Flow
Parallel Computing
Functions
Data Import and Export
Online Resources
Define your own function: calculate the arithmetic average of a numeric vector

```r
average <- function(x) {
  y <- sum(x)
  n <- length(x)
  z <- y/n
  return(z)
}
## calculate the average of 1:10
average(1:10)
## [1] 5.5
```
Data Import and Export

Read data from and write data to

- R native formats (incl. Rdata and RDS)
- CSV files
- EXCEL files
- ODBC databases
- SAS databases

R Data Import/Export:


Save and Load R Objects

- `save()`: save R objects into a `.Rdata` file
- `load()`: read R objects from a `.Rdata` file
- `rm()`: remove objects from R

```r
a <- 1:10
save(a, file = "../data/dumData.Rdata")
rm(a)
a

## Error in eval(expr, envir, enclos): object 'a' not found

load("../data/dumData.Rdata")
a
## [1]  1  2  3  4  5  6  7  8  9 10
```
Save and Load R Objects - More Functions

- **save.image()**: save current workspace to a file
  It saves everything!
- **readRDS()**: read a single R object from a .rds file
- **saveRDS()**: save a single R object to a file
- **Advantage of readRDS() and saveRDS()**: You can restore the data under a different object name.
- **Advantage of load() and save()**: You can save multiple R objects to one file.
Import from and Export to .CSV Files

- `write.csv()`: write an R object to a .CSV file
- `read.csv()`: read an R object from a .CSV file

```r
# create a data frame
var1 <- 1:5
df1 <- data.frame(var1, var2, var3)
```

```r
# save to a csv file
write.csv(df1, "../data/dummmyData.csv", row.names = FALSE)
```

```r
# read from a csv file
df2 <- read.csv("../data/dummmyData.csv")
print(df2)
```
Package *openxlsx*: read, write and edit XLSX files

```r
library(openxlsx)
xlsx.file <- "../data/dummmyData.xlsx"
write.xlsx(df2, xlsx.file, sheetName = "sheet1", row.names = F)
df3 <- read.xlsx(xlsx.file, sheet = "sheet1")
df3
## VarInt VarReal VarChar
## 1 1 0.1 R
## 2 2 0.2 and
## 3 3 0.3 Data Mining
## 4 4 0.4 Examples
## 5 5 0.5 Case Studies
```
### Read from Databases

- **Package `RODBC`**: provides connection to ODBC databases.
- **Function `odbcConnect()`**: sets up a connection to database
- **`sqlQuery()`**: sends an SQL query to the database
- **`odbcClose()`**: closes the connection.

```r
library(RODBC)
db <- odbcConnect(dsn = "servername", uid = "userid",
                  pwd = "******")
sql <- "SELECT * FROM lib.table WHERE ..."
# or read query from file
sql <- readChar("myQuery.sql", nchars=99999)
myData <- sqlQuery(db, sql, errors=TRUE)
odbcClose(db)
```
Read from Databases

- Package *RODBC*: provides connection to ODBC databases.
- Function `odbcConnect()`: sets up a connection to database
- `sqlQuery()`: sends an SQL query to the database
- `odbcClose()` closes the connection.

```r
library(RODBC)
db <- odbcConnect(dsn = "servername", uid = "userid",
pwd = "******")
sql <- "SELECT * FROM lib.table WHERE ..."
# or read query from file
sql <- readChar("myQuery.sql", nchars=99999)
myData <- sqlQuery(db, sql, errors=TRUE)
odbcClose(db)
```

Functions `sqlFetch()`, `sqlSave()` and `sqlUpdate()`: read, write or update a table in an ODBC database
Import Data from SAS

Package *foreign* provides function `read.ssd()` for importing SAS datasets (.sas7bdat files) into R.

```r
library(foreign)  # for importing SAS data
# the path of SAS on your computer
sashome <- "C:/Program Files/SAS/SASFoundation/9.4"
filepath <- "./data"
# filename should be no more than 8 characters, without extension
fileName <- "dumData"
# read data from a SAS dataset
a <- read.ssd(file.path(filepath), fileName,
              sascmd=file.path(sashome, "sas.exe"))
```
Import Data from SAS

Package *foreign* provides function `read.ssd()` for importing SAS datasets (.sas7bdat files) into R.

```r
library(foreign) # for importing SAS data
# the path of SAS on your computer
sashome <- "C:/Program Files/SAS/SASFoundation/9.4"
filepath <- "./data"
# filename should be no more than 8 characters, without extension
fileName <- "dumData"
# read data from a SAS dataset
a <- read.ssd(file.path(filepath), fileName,
               sascmd=file.path(sashome, "sas.exe"))
```

Alternatives:

- function `read.xport()`: read a file in SAS Transport (XPORT) format
- RStudio : Environment Panel : Import Dataset from SPSS/SAS/Stata
Contents

Introduction to R
RStudio
Pipe Operations
Data Objects
Control Flow
Parallel Computing
Functions
Data Import and Export

Online Resources
Online Resources

- Book titled *R and Data Mining: Examples and Case Studies*

- R Reference Card for Data Mining

- Free online courses and documents
  http://www.rdatamining.com/resources/

- RDataMining Group on LinkedIn (27,000+ members)
  http://group.rdatamining.com

- Twitter (3,300+ followers)
  @RDataMining
The End

Thanks!

Email: yanchang(at)RDataMining.com
Twitter: @RDataMining
How to Cite This Work

▶ Citation

▶ BibTex
@BOOK{Zhao2012R,
  title = {R and Data Mining: Examples and Case Studies},
  publisher = {Academic Press, Elsevier},
  year = {2012},
  author = {Yanchang Zhao},
  pages = {256},
  month = {December},
  isbn = {978-0-123-96963-7},
  keywords = {R, data mining},
  url = {http://www.rdatamining.com/docs/RDataMining-book.pdf}
}