Basics of R Programming

Yanchang Zhao
http://www.RDataMining.com

R and Data Mining Course
Canberra, Australia

10 December 2018
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 work?
Outline

Introduction to R

RStudio

Pipe Operations

Data Objects

Control Flow

Parallel Computing

Functions

Data Import and Export

Online Resources
What is R?

- R \(^1\) is a free software environment for statistical computing and graphics.
- R can be easily extended with 13,000+ packages available on CRAN\(^2\) (as of Dec 2018).
- Many other packages provided on Bioconductor\(^3\), R-Forge\(^4\), GitHub\(^5\), etc.
- R manuals on CRAN\(^6\)
  - An Introduction to R
  - The R Language Definition
  - R Data Import/Export
  - . . .

\(^1\)http://www.r-project.org/
\(^2\)http://cran.r-project.org/
\(^3\)http://www.bioconductor.org/
\(^4\)http://r-forge.r-project.org/
\(^5\)https://github.com/
\(^6\)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, :-(( 7).
- The CRAN Task Views 8 provide collections of packages for different tasks.
  - Machine learning & statistical learning
  - Cluster analysis & finite mixture models
  - Time series analysis
  - Multivariate statistics
  - Analysis of spatial data
  - ...

---

8 http://cran.r-project.org/web/views/
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)
prompt(a)
plot(a, type="b")
```

```
> a <- sample(10)
> print(a)
[1]  1  1  0  2  6  9  8  5  4  3  7
> plot(a, type="b")
```
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 in HTML, PDF and WORD formats
- Notebook: R notebook, Jupiter notebook
Outline

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?
Outline

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

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

```r
x <- 1:10
print(x)
```
```
[1] 1 2 3 4 5 6 7 8 9 10
```

## numeric vector, generated randomly from a uniform distribution

```r
y <- runif(5)
y
```
```
[1] 0.85400580 0.66021467 0.08613575 0.43215580 0.95526792
```

## character vector

```r
(z <- c("abc", "d", "ef", "g"))
```
```
[1] "abc" "d" "ef" "g"
```
## create a matrix with 4 rows, from a vector of 1:20

```r
m <- matrix(1:20, nrow=4, byrow=T)
```

```r
m
```

```r
## [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

```r
m - diag(nrow=4, ncol=5)
```

```r
## [1,] 0  2  3  4  5
## [2,] 6  6  8  9 10
## [3,] 11 12 12 14 15
## [4,] 16 17 18 18 20
```
```r
library(magrittr)
age <- c(45, 22, 61, 14, 37)
gender <- c("Female", "Male", "Male", "Female", "Male")
height <- c(1.68, 1.85, 1.80, 1.66, 1.72)
marr...
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"
Outline

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)
}
```

```
[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
Loop vs lapply

```r
## 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()
```

```r
## [1] 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379 1.79...
## [7] 1.9459101 2.0794415 2.1972246 2.3025851

## apply a function (log) to every element of x
tmp <- lapply(x, log)
y <- do.call("c", tmp) %>% print()
## [1] 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379 1.79...
## [7] 1.9459101 2.0794415 2.1972246 2.3025851
```
Outline

Introduction to R
RStudio
Pipe Operations
Data Objects
Control Flow
Parallel Computing
Functions
Data Import and Export
Online Resources
```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
ccluster <- makeCluster(n.cores)
## run jobs in parallel
tmp <- parLapply(cluster, x, log)
## 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)
```
Outline

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)
}
```

```r
## calculate the average of 1:10
average(1:10)
## [1] 5.5
```
Outline

Introduction to R

RStudio

Pipe Operations

Data Objects

Control Flow

Parallel Computing

Functions

Data Import and Export

Online Resources
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
var2 <- (1:5) / 10
var3 <- c("R", "and", "Data Mining", "Examples", "Case Studies")
df1 <- data.frame(var1, var2, var3)
names(df1) <- c("VarInt", "VarReal", "VarChar")

# save to a csv file
write.csv(df1, ".\data\dummyData.csv", row.names = FALSE)

# read from a csv file
df2 <- read.csv("./data/dummyData.csv")
print(df2)
```

```
## 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
```
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
d # the path of SAS on your computer
sashome <- "C:/Program Files/SAS/SASFoundation/9.4"
filepath <- ".//data"
d # filename should be no more than 8 characters, without extension
fileName <- "dumData"
d # read data from a SAS dataset
da <- 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"))
```

Another way: using function `read.xport()` to read a file in SAS Transport (XPORT) format
Outline

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

Online Resources
Online Resources

- Chapter 2: Data Import/Export, in book *R and Data Mining: Examples and Case Studies*
  

- RDataMining Reference Card
  

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

- RDataMining Group on LinkedIn (26,000+ members)
  
  http://group.rdatamining.com

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

Thanks!

Email: yanchang(at)RDataMining.com
Twitter: @RDataMining