Basics of R Programming

Yanchang Zhao

http://www.RDataMining.com

Short Course on R and Data Mining
University of Canberra

7 October 2016

Quiz

- Have you used R in your study or work?

Quiz

- Have you used R in your study or work?
- Are you doing or have you done any data mining study, research or applications?

Quiz

- Have you used R in your study or work?
- Are you doing or have you done any data mining study, research or applications?
- Have you used R for data mining and analytics in research or projects?

Outline

Introduction to R

RStudio

Data Objects

Control Flow

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 9,200+ packages available on CRAN[2] (as of Oct 2016).
- 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 was ranked #1 in the KDnuggets 2016 poll on *Top Analytics, Data Science software*[7] (actually R has been #1 in a row from 2011 to 2016!).
- *The CRAN Task Views* [8] provide collections of packages for different tasks.
  - Machine learning & atatistical learning
  - Cluster analysis & finite mixture models
  - Time series analysis
  - Multivariate statistics
  - Analysis of spatial data
  - . . .

[7] `http://kdnuggets.com/2016/06/r-python-top-analytics-data-mining-data-science-software.html`

[8] `http://cran.r-project.org/web/views/`

Outline

Introduction to R

RStudio

Data Objects

Control Flow

Data Import and Export

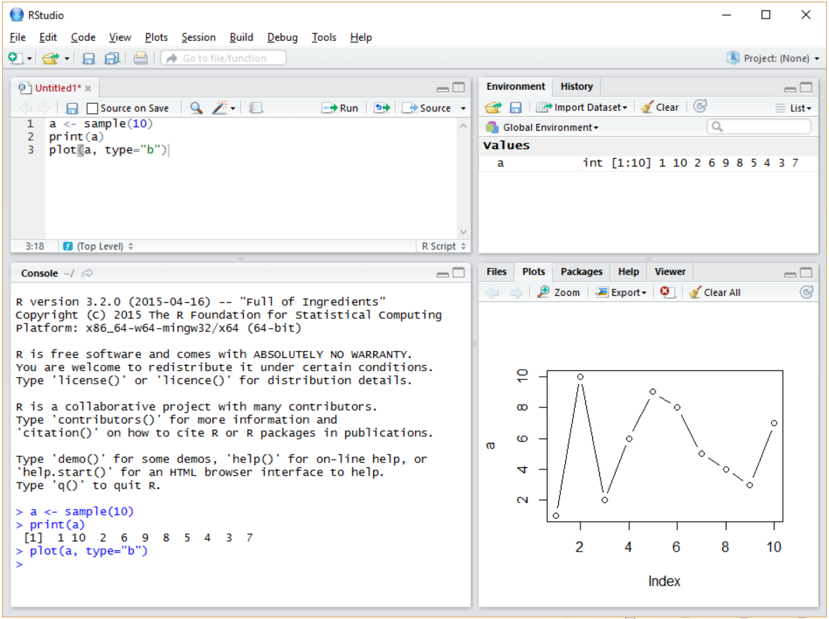
Online Resources

RStudio[9]

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

[9] https://www.rstudio.com/products/rstudio/

RStudio



```
a <- sample(10)
print(a)
plot(a, type="b")
```

values

a   int [1:10] 1 10 2 6 9 8 5 4 3 7

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

> a <- sample(10)
> print(a)
 [1]  1 10  2  6  9  8  5  4  3  7
> plot(a, type="b")
>
```

Outline

Introduction to R

RStudio

Data Objects

Control Flow

Data Import and Export

Online Resources

Data Types and Structures

- Data types
  - Integer
  - Numeric
  - Character
  - Factor
  - Logical
- Data structures
  - Vector
  - Matrix
  - Data frame
  - List

Vector

```r
## integer vector
x <- 1:10
# class(x)
print(x)

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

## numeric vector
y <- runif(5)
y

## [1] 0.6098228 0.6492410 0.8384724 0.1725274 0.3899108

## character vector
(z <- c("abc", "d", "ef", "g"))

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

Matrix

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

##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    6    7    8    9   10
## [3,]   11   12   13   14   15
## [4,]   16   17   18   19   20

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

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

Data Frame

```
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)
married <- c(T, F, T, F, F)
(df <- data.frame(age, gender, height, married))
```

```
##   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
```

List

```
x <- 1:10
y <- c("abc", "d", "ef", "g")
(ls <- list(x, y))

## [[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

Data Objects

Control Flow

Data Import and Export

Online Resources

Control Flow

- if ... else ...
- ifelse():

```r
score <- 1:5
ifelse(score >= 3, "pass", "fail")

## [1] "fail" "fail" "pass" "pass" "pass"
```

- for, while, repeat
- break, next

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

```
## for loop
x <- 1:10
y <- rep(NA, 10)
for (i in 1:length(x)) {
    y[i] <- log(x[i])
}
y

##  [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))

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

Parallel Computing

```r
## on Linux or Mac machines
library(parallel)
(n.cores <- detectCores() - 1)
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)
```

Outline

Introduction to R

RStudio

Data Objects

Control Flow

Data Import and Export

Online Resources

Data Import and Export [10]

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:

- `http://cran.r-project.org/doc/manuals/R-data.pdf`

[10] Chapter 2: Data Import and Export, in book *R and Data Mining: Examples and Case Studies*. `http://www.rdatamining.com/docs/RDataMining.pdf`

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

```
# 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/dummmyData.csv", row.names = FALSE)
# read from a csv file
df2 <- read.csv("./data/dummmyData.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
```

Import from and Export to EXCEL Files

Package *xlsx*: read, write, format Excel 2007 and Excel 97/2000/XP/2003 files

```
library(xlsx)
xlsx.file <- "./data/dummmyData.xlsx"
write.xlsx(df2, xlsx.file, sheetName = "sheet1", row.names = F)
df3 <- read.xlsx(xlsx.file, sheetName = "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.

```
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.

```
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/9.2"
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/9.2"
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

Data Objects

Control Flow

Data Import and Export

Online Resources

Online Resources

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

  `http://www.rdatamining.com/docs/RDataMining.pdf`

- R Reference Card for Data Mining

  `http://www.rdatamining.com/docs/R-refcard-data-mining.pdf`

- Free online courses and documents

  `http://www.rdatamining.com/resources/`

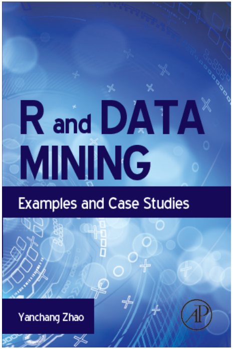
- RDataMining Group on LinkedIn (22,000+ members)

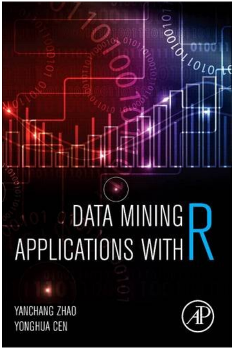
  `http://group.rdatamining.com`

- Twitter (2,700+ followers)

  @RDataMining

The End





Thanks!

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