

# Basics of R Programming

Yanchang Zhao

<http://www.RDataMining.com>

Tutorial on R and Data Mining

AusDM 2018

Charles Sturt University, Bathurst, Australia

29 November 2018

- ▶ Have you ever used R before?

- ▶ Have you ever used R before?
- ▶ Are you familiar with data mining and machine learning algorithms?

- ▶ Have you ever used R before?
- ▶ Are you familiar with data mining and machine learning algorithms?
- ▶ Have you used R for data mining and analytics in your work?

Introduction to R

RStudio

Data Objects

Control Flow

Parallel Computing

Functions

Pipe Operations

Data Import and Export

Online Resources

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

---

<sup>1</sup><http://www.r-project.org/>

<sup>2</sup><http://cran.r-project.org/>

<sup>3</sup><http://www.bioconductor.org/>

<sup>4</sup><http://r-forge.r-project.org/>

<sup>5</sup><https://github.com/>

<sup>6</sup><http://cran.r-project.org/manuals.html>

- ▶ R is widely used in both academia and industry.
- ▶ R was one of the top tools for data science and analytics, ranked #1 from 2011 to 2016 (but sadly overtaken by Python since 2017), according to the KDnuggets polls on *Top Analytics, Data Science software*<sup>7</sup>.
- ▶ *The CRAN Task Views*<sup>8</sup> 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
  - ▶ ...

---

<sup>7</sup> <https://www.kdnuggets.com/2018/05/poll-tools-analytics-data-science-machine-learning-results.html>

<sup>8</sup> <http://cran.r-project.org/web/views/>

Introduction to R

**RStudio**

Data Objects

Control Flow

Parallel Computing

Functions

Pipe Operations

Data Import and Export

Online Resources



- ▶ 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

---

<sup>9</sup><https://www.rstudio.com/products/rstudio/>

The screenshot shows the RStudio environment with the following components:

- Source Editor:** Contains three lines of R code:

```
1 a <- sample(10)
2 print(a)
3 plot(a, type="b")
```
- Environment:** Shows the 'Global Environment' with a variable 'a' of type 'int [1:10]' containing the values 1, 10, 2, 6, 9, 8, 5, 4, 3, 7.
- Console:** Displays the R version information, copyright notice, and the output of the code execution:

```
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")
>
```
- Plots:** A line plot with open circles showing the values of 'a' against an 'Index' from 1 to 10. The y-axis is labeled 'a' and ranges from 2 to 10. The x-axis is labeled 'Index' and ranges from 1 to 10. The data points are: (1, 1), (2, 10), (3, 2), (4, 6), (5, 9), (6, 8), (7, 5), (8, 4), (9, 3), (10, 7).

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

Introduction to R

RStudio

**Data Objects**

Control Flow

Parallel Computing

Functions

Pipe Operations

Data Import and Export

Online Resources

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

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

## numeric vector, generated randomly from a uniform distribution
y <- runif(5)
y
## [1] 0.4419456 0.4856832 0.7916919 0.1651451 0.3855163

## character vector
(z <- c("abc", "d", "ef", "g"))
## [1] "abc" "d" "ef" "g"
```

```
## create a matrix with 4 rows, from a vector of 1:20
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

## matrix subtraction
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
```

```
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)
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
```



```
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"
```

Introduction to R

RStudio

Data Objects

**Control Flow**

Parallel Computing

Functions

Pipe Operations

Data Import and Export

Online Resources

▶ `if ...else ...`

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

▶ `ifelse()`

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

- ▶ for, while, repeat
- ▶ break, next

```
for (i in 1:5) {  
  print(i ^ 2)  
}  
## [1] 1  
## [1] 4  
## [1] 9  
## [1] 16  
## [1] 25
```

- ▶ `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
## [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
```

Introduction to R

RStudio

Data Objects

Control Flow

**Parallel Computing**

Functions

Pipe Operations

Data Import and Export

Online Resources

```
## 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)
```



## Parallel Computing (cont.)

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

```
## 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)
```

Introduction to R

RStudio

Data Objects

Control Flow

Parallel Computing

**Functions**

Pipe Operations

Data Import and Export

Online Resources

Define your own function: calculate the arithmetic average of a numeric vector

```
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
```

Introduction to R

RStudio

Data Objects

Control Flow

Parallel Computing

Functions

**Pipe Operations**

Data Import and Export

Online Resources

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

```
library(magrittr) ## for pipe operations  
## traditional way  
b <- func3( func2( func1(a), p2) )  
## the above can be rewritten to  
b <- a %>% func1() %>% func2(p2) %>% func3()
```

Introduction to R

RStudio

Data Objects

Control Flow

Parallel Computing

Functions

Pipe Operations

**Data Import and Export**

Online Resources

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>

---

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

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

```
library(openxlsx)
xlsx.file <- "./data/dummyData.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

- ▶ 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)
```

- ▶ 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

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

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

Introduction to R

RStudio

Data Objects

Control Flow

Parallel Computing

Functions

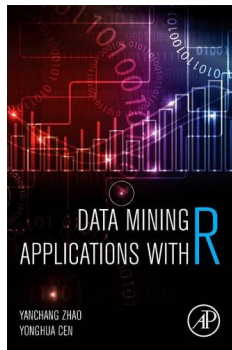
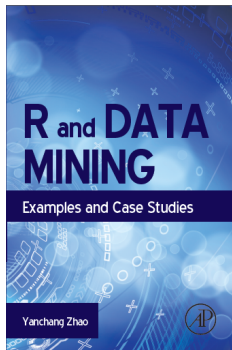
Pipe Operations

Data Import and Export

Online Resources



- ▶ Chapter 2: Data Import/Export, in book *R and Data Mining: Examples and Case Studies*  
<http://www.rdatamining.com/docs/RDataMining-book.pdf>
- ▶ RDataMining Reference Card  
<http://www.rdatamining.com/docs/RDataMining-reference-card.pdf>
- ▶ 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



Thanks!

Email: [yanchang\(at\)RDataMining.com](mailto:yanchang(at)RDataMining.com)

Twitter: @RDataMining