Introduction to R - Part 2 Data Science Skills Day 2022

Anjali Silva, PhD

Summer Undergraduate Data Science Research Program University of Toronto 03 June 2022









1/64

Intro

Course Description

- Introduction to R Part 2
 Data Science Skills Day
 - The vast amount of data produced by evolving information technology requires tools and skills. Among the many tools, R is a free, open-source language for data sciences. R is a programming language that can aid in the process of data analysis. This course is a beginner level, introductory course for R for data analysis. We will learn about R, RStudio (the environment use to work in R), including installation, and apply R for beginner-level data modeling and visualization. By the end of the course, you'll have a introduction to the flexibility of R, different functionalities, and understand how to apply it for basic data exploration.
 - Friday 10:00 am 4 pm EST; online synchronous.



Material

- Instructor Slides:
 - https://github.com/anjalisilva/DSI_IntroductionToR
 - SlideIntroR2022.pdf
 - SlideIntroR2022_part2.pdf

Data Frames & Booleans

Data Frames

• Data frames are used to store tabular data in R.

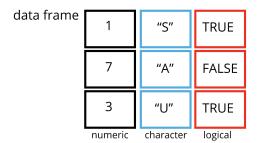


Figure: Data frames can store different classes of objects in each column. Figure from: https://datacarpentry.org/r-socialsci/02-starting-with-data/index.html

Data Frames

- Data frames are used to store tabular data in R.
- Data frames can store different classes of objects in each column.

```
dataFrameExample <- data.frame(
numbers = 1:4,
sex = c("M", "M", "F", "F"))

dataFrameExample
class(dataFrameExample) # "data.frame"
dim(dataFrameExample) # 4 2
names(dataFrameExample) # "numbers" "sex"</pre>
```

Data Frames

Data frames can be converted to a matrix using data.matrix().

```
dataMatrix <- data.matrix(dataFrameExample)
class(dataMatrix) # "matrix"
dim(dataMatrix) # 4 2</pre>
```

Question 14:

Generate the following information into a data frame.

1	numbers	sex	age	height
2	1	M	30	72
3	2	M	31	70
4	3	F	40	65
5	4	F	35	62.4

Question 15:

After generating the data frame, you realize the height is recorded in inches but should be changed to centimeters. How would you do this? Note, 1 inch = 2.54 cm.

1	numbers	sex	age	height
2	1	M	30	72
3	2	M	31	70
4	3	F	40	65
5	4	F	35	62.4

Booleans

• TRUE/FALSE or T/F are called "boolean" values.

```
testValue <- FALSE
typeof(testValue)
is.logical(testValue)</pre>
```

- ! testValue
- "!" is the NOT operator. It returns the opposite of the argument.

Boolean Operators

Operators > or <:

$$a < -c(1:5)$$

а

• Equivalence test:

Boolean Operators

• & symbol is the "AND" operator:

symbol is the "OR" operator:

Any questions?

Question 16:

Given the below numeric vector 'numericVec', how can a user check if values are greater than 5?

numericVec \leftarrow c(1.1, 3, 5.3, 2)

Question 17:

Given the below numeric vector 'numericVec', how can a user retrieve the value/s that is/are greater than 5, from the vector?

numericVec <-c(1.1, 3, 5.3, 2)



Data Import/Export

Data Import/Export

• To see the list of pre-loaded data in R:

```
data(package = "datasets")
```

AirPassengers # Example dataset head(AirPassengers) # see first few entries tail(AirPassengers) # see last few entries

Data Slicing

• Let us look at another pre-loaded datasets:

```
women # another dataset (last)
?women
dim(women) # 15  2
class(women) # "data.frame"
head(women) # height and weight information
women$height > 60 # slicing
women[women$height > 60, ] # slicing
```

Data Reading/Writing

Files can be written using functions like write.csv(), write.table():

```
getwd() # file will be saved here
write.csv(x = women, file = "women.csv")
# saving women dataset in current working directory
```

Data Reading/Writing

 Txt files can be read using read.table("location of the file") or read.csv()

womenNew[c(1:5),] # to view first 5 rows
womenNew[, 1] # to view first column

```
womenNew <- read.csv(file = "women.csv", row.names = 1)
womenNew
head(womenNew) # to view first part of object
tail(womenNew) # to view last part of object
dim(womenNew) # 15 2</pre>
```

• "+" is used for addition:

• "-" is used for subtraction:

• "/" is used for division:

• "*" is used for multiplication:

• "%*%" is used for matrix multiplication:

```
a <- matrix(1:6, nrow = 2, ncol = 3) # 2 x 3 matrix
a
b <- matrix(7:12, nrow = 3, ncol = 2) # 3 x 2 matrix
b
c <- a %*% b
c # 2 x 2 matrix</pre>
```

Any questions?

Functions

Writing Functions

- Functions combine a sequence of expressions that are executed to achieve a goal.
- Can be reused without rewriting the sequence of expressions.
- Take input, function arguments and generate output, return value.
- When writing functions, ask yourself:
 - What will the user want to modify in this function?
 - This will help determine function arguments.

Function Interface

Functions are defined using 'function' assigned to a variable name.

```
firstFunction <- function(argumentOne, argumentTwo) {
  cat("\n First argument is", argumentOne, "\n")
   cat("\n Second argument is", argumentTwo, "\n")
  argumentThree <- argumentOne + argumentTwo
  cat("\n argumentOne + argumentTwo is",
  argumentThree, "\n")

return(argumentThree)
  # always end with return statement (best practice)
}</pre>
```

Function Interface

- Function names cannot begin with a number.
- To run the function:

```
firstFunction(argumentOne = 2, argumentTwo = 3)
```

• To "call" or "invoke" the function, type function name:

firstFunction

Function Interface

- Default values play a vital role in R functions and influence user's behaviour.
- Default values should be assigned using "=", and not "←".

```
secondFunction <- function(argOne = 1, argTwo = 3) {
  cat("\n First argument is", argOne, "\n")
  cat("\n Second argument is", argTwo, "\n")
  argThree <- argOne + argTwo
  cat("\n argOne + argTwo is", argThree, "\n")
  return(argThree)
}</pre>
```

Local and Global Variables

- Where variables are defined matters.
- Variables defined within functions are only accessible from within the function.
- Variables declared within a function are called "local".
- Variables declared outside of functions are called "global".

Variables

 If variables are defined outside of functions, globally, they hold that value.

```
argOne <- "Hello"
anotherFunction <- function() {
  argOne <- 10
  return(argOne)
  }
anotherFunction()
argOne # What would this return?</pre>
```

Variables

Need to pass data into functions:

```
women
# Incorrect
dataAnalysis <- function() {</pre>
    heightData <- women$height
   return(mean(heightData))
dataAnalysis() # will work only for women$height
# Recommended
dataAnalysis <- function(inputData) {</pre>
    meanOfHeight <- mean(inputData)</pre>
    return(meanOfHeight)
dataAnalysis(inputData = women$height)
# will work for any dataset with height information
```

Any questions?

Non-base Functions

- Functions have been written by other authors.
- Non-base functions in R can be utilized by downloading R packages.
- To see all the currently loaded packages:

```
search()
```

Non-base Functions

- Popular repositories for Packages:
 - The Comprehensive R Archive Network (CRAN)
 - Link: https://cran.r-project.org/web/packages/
 - Bioconductor
 - Link: https://www.bioconductor.org/packages/release/bioc/
 - GitHub
- Depending on the source of Package, downloading instructions may differ.

tro Data Frames Data Import/Export Arithmetic **Functions** Graphics Next Steps

Example

- User wants to do a cluster analysis of the women built-in dataset.
- Assume the user is interested in performing model-based clustering using Gaussian finite mixture models.
 - Do an R search: ??clustering
 - Google R packages for model-based clustering.
 - Read blogs on clustering.

Example, continue...

- Get to know about R package 'mclust'.
- Visit https:
 //cran.r-project.org/web/packages/mclust/index.html.
- Click on Reference manual to see all the functions within the package.
- Download package to R session:

```
# Install package from CRAN, case matters!
install.packages("mclust")
library("mclust") # to load and attach
```

Example, continue...

More details on 'mclust' package (only after downloading):

```
vignette("mclust") # vignette for 'mclust'
?'mclust-package' # get information on package
?mclust # get information on package
ls("package:mclust") # list all functions in package
```

Example, continue...

Work with 'mclust' package:

```
# Running mclust
MclustResults <- mclust::Mclust(data = women)
str(MclustResults) # provide the structure; list of 15
names(MclustResults)
MclustResults$G # There are four clusters in the dataset
# Citing the package
citation("mclust")</pre>
```

Bioconductor

- Bioconductor packages are available from bioconductor.org.
- URL: https://www.bioconductor.org/install/.

```
if (! requireNamespace("BiocManager", quietly = TRUE))
install.packages("BiocManager")
```

 Bioconductor packages are listed here: https://www.bioconductor.org/packages/release/bioc/.

Title

Bioconductor



Search:

Home Install Help Developers About

Home » Bioconductor 3.11 » 3.11 Software Packages

Bioconductor Software Packages

**-1-*-1---

Bioconductor version: Release (3.11)

Package	Maintainer	Title
<u>a4</u>	Tobias Verbeke, Laure Cougnaud	Automated Affymetrix Array Analysis Umbrella Package
a4Base	Tobias Verbeke, Laure Cougnaud	Automated Affymetrix Array Analysis Base Package
a4Classif	Tobias Verbeke, Laure Cougnaud	Automated Affymetrix Array Analysis Classification Package
a4Core	Tobias Verbeke, Laure Cougnaud	Automated Affymetrix Array Analysis Core Package
a4Preproc	Tobias Verbeke, Laure Cougnaud	Automated Affymetrix Array Analysis Preprocessing Package
a4Reporting	Tobias Verbeke, Laure Cougnaud	Automated Affymetrix Array Analysis Reporting Package
ABAEnrichment	Steffi Grote	Gene expression enrichment in human brain regions
<u>ABarray</u>	Yongming Andrew Sun	Microarray QA and statistical data analysis for Applied Biosystems Genome Survey Microrarray (AB1700) gene expression data.
abseqR	JiaHong Fong	Reporting and data analysis functionalities for Rep-Seq datasets of antibody libraries

Packages » Bioconductor's : release:

Bioconductor's stable, semi-annual

- Analysis software packages.
- Annotation packages.
 Illustrative experiment data packages.
- Workflow packages.
- · Latest release announcement.

Bioconductor is also available via Docker and Amazon Machine Images.

Development Version »

Bioconductor packages under development:

- Analysis software packages.
- Annotation packages
 Illustrative experiment data packages
 - . _

Developer Resources:

GIT Log

Example

- User is interested in Bioconductor package GenomicFeatures.
- To load package:

```
BiocManager::install("GenomicFeatures")
library("GenomicFeatures")
# list all functions in the package
ls("package:GenomicFeatures")
```

• More about installation:

```
https://www.bioconductor.org/install/.
```

Bioconductor



GitHub

To down from GitHub, need to use 'devtools' pacakge.

```
library("devtools")
devtools::install_github("<GitHubUsername>/<PackageName>",
  build_vignettes = TRUE)
```

- Example: https://github.com/anjalisilva/mixGaussian
- To download this GitHub R package:

```
library("devtools")
devtools::install_github("anjalisilva/mixGaussian",
build_vignettes = TRUE)
library("mixGaussian")
ls("package:mixGaussian") # to see all functions
```

Any questions?

Question 18:

You are interested in R packages for analyzing Covid-19 data for a project you are doing. So you search Google for such R packages.

You come across a package called 'covid19.analytics' which has a GitHub R package

(https://github.com/mponce0/covid19.analytics) that was later developed into a CRAN package (https://cran.r-project.org/web/packages/covid19.analytics/index.html). How would you download both the GitHub and CRAN package?

R: Graphics

Applying Base Functions

• Package 'graphics' is a standard or base package.

Applying Base Functions

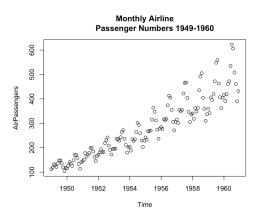


Figure: Plotting AirPassengers dataset as a scatter plot.



Applying Base Functions

Package 'graphics' is a base package.

tro Data Frames Data Import/Export Arithmetic Functions **Graphics** Next Step

Applying Base Functions

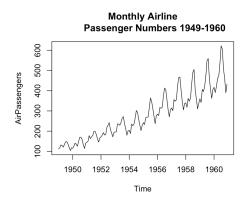


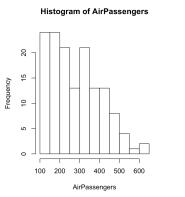
Figure: Plotting AirPassengers dataset as a line plot.

Applying Base Functions

Package 'graphics' is a base package.

```
library(graphics) # to load and attach
?AirPassengers
# Monthly Airline Passenger Numbers 1949-1960
# To have multiple plots in one overall plot using par(mfrow = c(1, 2))
hist(AirPassengers) # histogram
boxplot(AirPassengers) # boxplot
```

Applying Base Functions



Boxplot of AirPassengers

Figure: Output from plotting AirPassengers dataset.

Applying Base Functions

Package 'graphics' is a base package.

```
library(graphics) # to load and attach
?iris
?pairs

pairs(iris[, c(1:4)], col = iris$Species,
    main = "Scatter plot of iris dataset")
```

Applying Base Functions

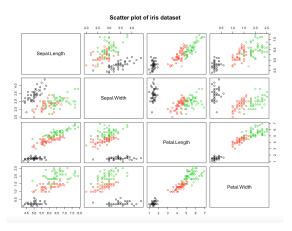


Figure: Output from plotting iris dataset as a pairs plot.



Applying Base Functions

Package 'graphics' is a base package.

Applying Base Functions

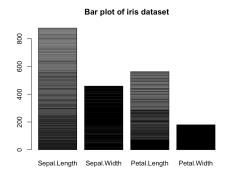


Figure: Output from plotting iris dataset as a bar plot.

R Packages for Plotting

- More resources:
 - R Graphics Cookbook: https://r-graphics.org/.
 - sthda: http://www.sthda.com/english/wiki/ggplot2-essentials.

Any questions?

Next Steps

Resources

- The R Journal: https://journal.r-project.org/
- R for Data Science textbook (free): https://r4ds.had.co.nz/.
- Bioconductor Workshops: https://www.bioconductor.org/help/events/
- Coursera offers courses via both paid + free options
 - https://www.coursera.org/courses?query=bioinformatics

Resources



R packages for data science

The tidyverse is an opinionated **collection of R packages** designed for data science. All packages share an underlying design philosophy, grammar, and data structures

Install the complete tidyverse with:

install.packages("tidyverse")

Figure: Tidyverse R Packages designed for data science. Figure from https://www.tidyverse.org/.



Question 19:

You are interested in R packages for analyzing Covid-19 data for a project you are doing. So you search Google for such R packages.

You come across a package called 'covid19.analytics' which has a GitHub R package

(https://github.com/mponce0/covid19.analytics) that was later developed into a CRAN package (https://cran.r-project.org/web/packages/covid19.analytics/index.html). How would you download both the GitHub and CRAN package?