COURSE 07: Data Analysis with R Programming

Page No.
Date 07 | 03 | 23

Journey So Far

Instructor : Carsie
Research Manager within People Operations

> Use structured thinking to define a problem and ask the eight questions.
> Work with spreadsheets, databases, and tools like SQL to organize and transform data
> Clean your data to make sure it has integrity before your analyze it
> Create impactful data visualization to illustrate key points.
> Craft a compelling story to communicate insights to stakeholders.

Ask
Prepare
Process
Analyze
Share
Act

* Computer Programming:
Giving instructions to a computer to perform an action or set of actions.

R is useful for organizing, cleaning & Analyzing

This course :→
> Intro to programming languages
> Explore main features and functions
> Basic programming concepts in R
> How to ~~work with~~ data in R
> Clean, transform, visualize, report data in R

Page No.
Date

Fun with R

Palmer Penguins Dataset →
Size measurements for three penguin species that live on the palmer Archipelago in Antarctia
↓
Mostly used for exploration & Teaching

#Relax & watch the Power of R

---

* Getting Started with R:
Don't be afraid of errors!

Week 1
2/3

08/03/23

Programming Languages:
The words & symbols we use to write instructions for computers to follow.

Coding is writing instructions to the computer in the syntax of a specific programming language.

Ex. R, Python, JS, SAS, Scala, Julia

---

* Advantages of using PL for DA:
- Clarity the steps of your Analysis
- Save time
- Reproduce and Share work with colleagues, etc

Reproduce = Script

R: A PL frequently used for statistical Analysis, visualization and other data analysis.

R is based on S → 1970 John Chambers
for internal use at Bell Labs
1990s Ross Ooraca & Robert Gentleman
developed R
at University of Auckland, New Zel.

* Why use R:→ > Accessible
> Data-centric
> Open Source
> Community

Check out: R For Data Science Learning Community
R-Studio Community

* When to use R:→ > Reproducing your Analysis
> Processing lots of data
> Creating data visualizations
↓R is on Another Level.

In the Console:-
print("Hello, world!") → [1] Hello, world!
55+45 → [1] 100
69 == 96 → [1] FALSE

Week 1
3/3

Lubridate is an R package that makes it easier to work with dates and times.

Page No.
Date 09 | 03 | 23

Intro to RStudio

IDE : A software app that brings together all the tools you may want to use in a single place.

R console, R Editor, tools for managing data & creating visuals.

R is the Engine & RStudio is the Dashboard

* Packages are units of Reproducible R-code. Members of R-community create packages to keep track of R functions that they write and reuse.

* Lubridate is a part of tidyverse. The tidyverse is a collection of packages in R with a common design philosophy for data manipulation, exploration and visualization

* For a lot of analysis tidyverse is Essential

In the console type =>

install.packages("tidyverse")

Some processes can take a while, the Red icon in the Upper right of console indicates that.

Also

when the cursor appears in the console.

Takeaway :→ lubricate is part of tidyverse ; tidyverse → Essential

```
install.packages("tidyverse")
library(tidyverse)
library(lubridate)
```

R → Case sensetive

To load tidyverse package ⟹ > library(tidyverse)

> library(lubricate)

Done

R studio has 4 main windows called panes :-

4th is hidden by default so you'll see 3 at first.

↓

It's the R-script : To open:

file > New file > R script

Drag border to adjust sizes

View > panes > = for more customization

---

2 ways to code → Console
→ R-script [Source Editor]

Console > install.packages("palmerpenguins")

> library("palmerpenguins")

> summary(penguins)

Works together

Source Editor : View(penguins) → shows tabular form.

Environment Pane:

> All the data you currently have uploaded
> OR if you import using spreadsheet, it'll be here.
> Can view each object by clicking.

All the previous commands are available in History tab.
Double clicking the line copies it to console.

* Lower Right Pane: Files, Plots, Packages, Help, Viewer

Files: Access to file directory and shows content of current working folder.

Plots: Appear here.

Scatter plot

```
ggplot(data = penguins, aes(x = flipper_length_mm,
y = body_mass_g)) + geom_point(aes(color = species))
```

Help: ✓✓

Week 2
1/3

Page No.
Date 10 | 03 | 23

This week

> Fundamental of Programming using R in RStudio
> Coding in RStudio
> Syntax for performing calculations
> Pipes make a sequence of code easier to work with and read.
> R packages
> Reusable functions & more
> Tidyverse & packages (ggplot2 for viz.)

---

* Fundamentals :- Basic concepts of R :
  > Functions
  > Comments
  > Variables
  > Data Types
  > Vectors
  > Pipes

* Function (R):
A body of reusable code used to perform specific tasks in R.

* Argument (R)
Info that a function in R needs in order to run.

```
> print("Coding in R")
[1] "Coding in R"
> ?print()
```
→ Returns a page in help window

* Variable (R) :-
A representation of a value in R that can be stored for use later during programming.
Variables also called as objects.
↓
Should start with a letter and contains numbers and underscores.

other assignment operator work too, but stick to just one type in your code

Page No.
Date

* # Comment in R

first_variable ← "This is my variable"

When we hit Run, Environment Pane updates.

Second_var ← 12.5 [Run]

Logical, date, date_time, etc.

* Vector (R):
A group of data elements of same type stored in a sequence in R

> Can be made using Combine function
Just a c followed by ( )

c(a, b, c, d, ..., z)

> Vect_1 ← c(13, 48, 12.5, 17, 28.5)
> Vect_1
[1] 13.0 48.0 12.5 17.0 28.5

* Pipe (R) :
A tool in R for expressing a sequence of multiple operations, represented with "%>%"

Apply output of one function as to another function

This filters & sorts data :-

Ex. ToothGrowth %>% filter(dose == 0.5) %>% arrange(len)

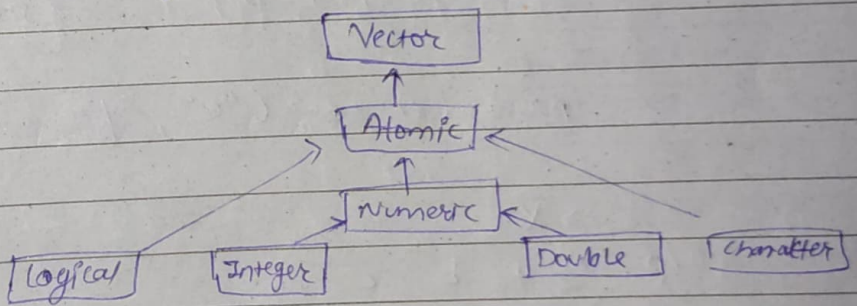
* There are 2 types of vectors :
- Atomic Vectors
- Lists

Vector is a group of data elements of same type stored in a sequence.

There are 6 primary types of Atomic vectors

4 main Types	Logical	True/False	TRUE
	Integer	Positive and negative whole values	3
	Double	Decimal values	120.75
	Character	String / character values	"coding"

Hierarchy :-



Creating vector : Using c() function

Note :- When creating integer vector append L after each.

```
nos <- c(1L, 5L, 7L)
```

Also,
```
c <- c(10.2, 7.9, 69.2.5)
c <- ("sora", "zoro", "choro")
c <- (TRUE, FALSE, TRUE)
```

Length(vector) → to determine length of vector.
↓
Var name

Page No.
Date

```
typeof(c("a", "b"))
> [1] "character"
```

```
cool ← c(1L, 3L)
typeof(cool)
> [1] "integer"
```

```
length(cool)
> [1] 2
```

---

* Boolean Checking:
  is.logical(), is.double(), is.integer(), is.character()
  ↓ is.vector()
  Returns TRUE/FALSE

---

* Naming Vector ← names(x)
  ↳ vector name

```
x ← (1, 3, 5)        {stored as double & not int}

names(x) ← ("a", "b", "c")

> x
o/p > a  b  c
> 1  2  3
```

Ex

```
remain ← c(11, 12, 11, 13)
suits  ← c("spades", "hearts", "diamonds", "clubs")
names(remain) ← suits
> remain
```

#	spades	hearts	diamonds	clubs
#	11	12	11	13

*Note:- Atomic vector contains elements of same type

If other-way around use lists.

Page No.
Date

Also

```
Remain <- c(spades = 11, hearts = 12,
            diamonds = 11, clubs = 13)
```

While simultaneously naming, it optional to keep the names in quotes.

```
Remain <- c("spades" = 11, "hearts" = 12,
            "diamonds" = 11, "clubs" = 13)
```

```
> str(Remain)
 Named num [1:4] 11 12 11 13
 - attr(*, "names") = chr [1:4] "spades" "hearts"
                                "diamonds" "clubs"
```

Single valued vectors

```
> my_apples <- 5
> my_oranges <- "six"
> is.vector(my_apples)   # TRUE
> is.vector(my_oranges)  # TRUE
> length(suits)  # 4
```

Vectors allows only homogeneous type so it performs any Coersion if necessary,

Converting different types to one ex. int to str etc

```
> class(suits)
# character.
```

Page No.
Date 13/03/23

* Lists:
Elements can be of any type:
dates, dataframe, vectors, matrices, etc.

To create use list() similar to c()

Ex. list("a", 1L, 1.5, TRUE)

Valid Ex. list(list(list(1, 3, 5)))

* Structure: str()
To determine size & type.

```
str(list("a", 1L, 1.5, TRUE)
> List of 4
> $ : chr "a"
> $ int 1
> $ num 1.5
> $ logi TRUE
```

→ It describes the variable called.

```
z ← list(list(list(1,3,5)))
str(z)
> List of 1
> $ List of 1
> .. $ List of 3
> .. .. num 1
> .. .. num 3
> .. .. num 5
```

Page No.
Date

Indentation suggest that its a list of list of list.

---

* Naming lists:

list ('Chicago' = 1, 'New York' = 2, 'Los Angeles' = 3)

$ Chicago
[1] 1
$ 'New York'
[1] 2
$ 'Los Angeles'
[1] 3

R - creates dates in standard : YYYY-MM-DD

Page No.
Date 14/03/23

Dates & times

> install.packages ("tidyverse")
> load (tidyverse)
> load (lubridate)

Types →
Date : "2016-08-16"
Time : "20:11:19 UTC"
Date Time : "2018-09-17 18:15:58 UTC"

today() → gives current date
now() → gives current date time

* 3-way we'll be creating date-time formats:
  > From a string
  > From an individual date
  > From an existing date/time objects

* Converting from string:
  ymd ("2021-01-20") → [1] "2021-01-20"
  mdy ("January 20th, 2021") → [1] "2021-01-20"
  dmy ("20-Jan-2021") → [1] "2021-01-20"

Converts into YYYY-MM-DD ONLY

These three functions can also take quoted numbers and convert them into dates.

ymd (20210120)
dmy (01202021)
d m Y (20012021)
>> [1] "2021-01-20"

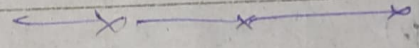
* Date-time to Date =>
as_date(now()) > [1] "2021-01-20"

* Date-time :-
ymd() & its variations creates dates. To create date time add underscore (_) and add one or more of letters h, m, s

~~ex. ymd_hms~~

ex. ymd_hms("2021-01-20 20:11:59")
> [1] "2021-01-20 20:11:59 UTC"
mdy_hm("01/20/2021 08:01")
> [1] "2021-01-20 08:01:00 UTC"
dmy_h("15/12/2022 20")
> [1] "2022-12-15 20:00:00 UTC"

& every combo



* Data Frames :-
data.frame(x = c(1, 2, 3), y = c(1.5, 5.5, 7.5))

	x	y
1	1	1.5
2	2	5.5
3	3	7.5

* files :-
dir.create("dest_folder") # To create
file.create("new_file.txt")
file.create("new.docx")
file.create("new.csv")
} self explanatory

if file create returns TRUE else FALSE
file.copy("new.csv", "dest_fo")
unlink("som.txt") # To delete

* matrices :

> objects in which elements are arranged in 2-D rectangular format
> Only homogenous element type
> Create using matrix()

Basic syntax :→

matrix(data, nrow, ncol, byrow, dimnames)
↓ or ↓ ↓
= = Bool

Ex. P ← matrix(c(3:14), nrow=4, byrow=TRUE)

print(P)

	[,1]	[,2]	[,3]
[1,]	3	4	5
[2,]	6	7	8
[3,]	9	10	11
[4,]	12	13	14

access element at (1,3) & (4,2)

print(P[1,3]) → 5
print(P[4,2]) → 13

Q ← matrix (c(14:25), nrow=4, byrow=TRUE)

print(P+Q)
print(P-Q)
print(P/Q)
print(P*Q)

Week 2
3/4

BODMAS()

Page No.
Date 14/03/23

* Logical Operators :→ & → AND
| → OR
! → NOT

> if () { }
> else () { }
> else if { }

Ex. datacamp_views ← c(100, 20, 5, 200, 60, 88, 190, 33, 290, 64)

youtube_views ← c(10, 20, 1000, 64, 100, 9, 19, 3, 90, 4)

(#) datacamp_views > youtube_views
1. TRUE
2. FALSE
3. FALSE
:
10. TRUE

views ← matrix(c(datacamp_views, youtube_views), nrow=2, byrow=TRUE)

views

op → (#)	100	20	5	200	60	88	190	33	290	64
	10	20	1000	64	100	9	19	3	90	4

views == 20

>	F	T	F	F	F	F	F	F	F	F
	F	T	F	F	F	F	F	F	F	F

WEEK 2
1/3

Page No.
Date 16/03/22

Packages → Units of reproducible R-code

Includes →
- Re-usable R functions
- Documentation about the functions
- Sample datasets
- Tests for checking your code

By default, R- includes a set of packages called Base R.

installed.packages
↓
shows a list of installed package

priority → base → installed & loaded
recommended → installed but not loaded

To load → library(name)

most commonly used source →

CRAN (Comprehensive R Archive Network):
An online archive with R packages, source code, manuals & R documentation.

Tidyverse (R):
A system of packages in R with a common design philosophy for data manipulation, exploration & visualization.

Using Tidyverse can help you work your way through pretty much the entire data analysis process.

Why ~~was~~ CRAN was created?

> Since packages are made by users, people need a reliable way to check and validate submitted code

> CRAN makes sure any R content open to public meets required quality standards.

GitHub also is large source of R packages.

→

> install.packages("tidyverse")

quotes aren't always necessary.

> library("tidyverse")

→ 8 other also imported & some CONFLICTS

Conflicts happen when packages have functions with same name as other functions

> Function related to last package loaded is used

8 core tidyverse packages:

> ggplot2
> tibble
> tidyr
> readr
> purrr
> dplyr
> stringr
> forcats

tidyverse_update(): To keep track of updates

tidyverse_update() → keep track of updates
update.packages() → update all packages
Install.package("name") → update single

* Vignett: Documentation that acts as a guide to an R package.

> shares details about problems that the package is designed to solve.

© browseVignettes("name")

4 most essential packages for DA →

(*)
- > ggplot2 → Data viz
- > dplyr → Data manipulation
- > tidyr → Cleaning
- > readr → Importing (read.csv, etc).

> tibble → Dataframes
> purrr → functions & vectors
> stringr → function related to strings
> forcats → some common problem with factors

Factors (R)

- store categorical data in R where data values are limited and usually based on a finite group like country or year.

Working with Pipes

Page No.
Date 17/03/23

* **Pipe:**
A tool in R for expressing a sequence of multiple operations, represented with "%>%"

↓

It takes the output of one statement and makes it the input of next statement.

↓

We can use pipe instead of typing out functions inside function. (Nesting)

Example:→
- Call up data (and then)
- Group the data (and then)
- Summarize the grouped data using a mean function.

* To use already installed dataset we use data()

```
data("ToothGrowth")
View(ToothGrowth)
```

↓

Say we need to filter and sort this data for analysis

→ Nesting commands / sequence of data frames.

filter

```
install.packages("dplyr")      dplyr
library(dplyr)
filtered_tg <- filter(ToothGrowth, dose==0.5)
View(filtered_tg)
```

Filter & Sort

View() To view

CTRL + SHIFT + M
Pipe

data("ToothGrowth")
View(ToothGrowth)

filtered_tg ← filter(ToothGrowth, dose == 0.5)
View(filtered_tg)

arrange(filtered_tg, len)
→ Output in console
Ascending sorted by len

So the tooth growth data is filtered (dose == 0.5) & sorted by len.

↓

Another way ↦

arrange(filter(ToothGrowth, dose == 0.5), len)

Pipe

filtered_toothgrowth <- ToothGrowth %>% %
filter(dose == 0.5) %>%
group_by(supp) %>%
summarize(mean_len = mean(len, na.rm = T), .group = "drop")

> If pipe operator is detected, then automatically indentation is corrected.
> Last line doesn't have a pipe

- Pipe op is added at the end
- Check code after adding pipe
- Reused to check what went wrong.

Connor : Marketing Analytics Manager!

> Overall readability is an important aspect.

> Using Comments!

> Documentation will explain in depth exactly what your code is doing, why it was built, what is the purpose for it and any limitations.

> Building it for scalibity as well as making it dynamic.
Q. will the code be used in future?
if yes, make it in such a way that it will be able to scale.

Dynamic : Not hard coding [YUP!]