R for Novice Programmers (1e)

Table of contents

Welcome

This is the website for "R for Novice Programmers." The goal of this book is to introduce non-programmers or those with very little programming experience to the benefits of the R and RStudio software. The main prerequisites for learners are basic knowledge of computer applications and experience working with files and folders. This book will primarily focus on the basic R concepts that are hardly emphasized, but that may prove difficult for learners new to programming.

This book was built with Quarto.

Introduction

Why did I write this book?

This book is primarily intended to cater to the needs of individuals who have a desire to learn the basics of programming. I focus on R and RStudio because their capabilities may be relevant to a wide variety of individuals and organizations seeking to perform basic statistical analysis and data visualization. Personally, my skills in R and RStudio were gained via classroom instruction, online tutorials and videos, as well as relevant blog posts. A significant disadvantage of some of these resources is the assumption of prior programming knowledge. To address this, I begin the book with instructions for downloading software, navigating the R and RStudio interfaces, and an overview of the basics of R to decrease the cognitive load on novices.

About the author

The author of this book is a Certified Carpentries instructor and a trainer with the Digital Research Academy. Additionally, the author holds a PhD in Biomedical Engineering and has completed postdoctoral fellowships in vascular biology and infectious diseases. Lastly, the author is passionate about using R and RStudio to generate data-driven visualizations to allow for a deeper understanding of public policy issues.

Syllabus

At the end of the book, the student should be able to perform the tasks listed in the syllabus below.

Table 1: Syllabus

Chapter	Title	Date Completed
	Introduction	
1	Overview of R and RStudio	
2	Download and Install R and RStudio	

Chapter	Title	Date Completed
3	Navigating the R and RStudio interfaces	
4	Managing your files and data	
5	Importing data and saving analysis outputs	
6	Basic arithmetic, arithmetic operators, and variables	
7	The primary types of operators in R	
8	Data Types	
9	Vectors	
10	Data Structures (Part I)	
11	Data Structures (Part II)	
12	Handling missing data	
	Conclusion	
	Appendix	

Sample chapter design

Each lesson will follow a pre-described format

i. Questions to be addressed

ii. Learning objectives

iii. Lesson content

iv. Practice exercises

v. Lesson summary

Feedback

Feedback can be provided using the following channels:

i. Email: willyokech@gmail.com

ii. Github pull request:

Summary

Overall, I believe that this book will increase both the knowledge and confidence levels of novice programmers and allow them to perform basic statistical analysis and simplify everyday computational tasks at home or in their workplaces. In the next chapter, we will perform a basic overview of the R and RStudio ecosystem.

1 Overview of R and RStudio

1.1 Questions

- What is R? How is it related to RStudio?
- Why is R considered a powerful language for statistical computing and data analysis?
- What are some of the common uses of R in various fields?
- What advantages does R offer over other programming languages for data science tasks?

1.2 Learning Objectives

- Learn about the historical background of R and RStudio.
- Understand the uses and primary advantages of R and RStudio.
- Explore the various applications of R across different industries.

1.3 Lesson Content

1.3.1 Why learn R and RStudio?

Both R and Studio are free, open-source software tools that are widely used for statistical analysis and data visualization. R is a programming language that enables the use of code to analyze data. The primary function of the R language is statistical analysis and this can be performed directly in the R console. To ease the analysis process and enhance usability, an integrated development environment (IDE), such as RStudio is recommended. The RStudio IDE is a user-friendly interface that allows the learner to manage multiple script files, use the command-line terminal, easily access file inputs and outputs, and review file/analysis history.

The R programming language software was developed by Ross Ihaka and Robert Gentleman in 1993 (published as open-source in 1995) when they were based at the University of Auckland. *Fun fact: R represents the first letter of the first names of the creators.* The software is utilized by individuals working for various organizations ranging from academic institutions

and healthcare organizations to financial services and information technology companies. In January 2024, the PopularitY of Programming Language (PYPL) Index, which is created by analyzing how often language tutorials are searched on Google, demonstrated that R was the 6th most popular programming language. However, in the same period, the TIOBE index (https://www.tiobe.com/tiobe-index/) indicated that R was the 23rd most popular language. This may result from different methodologies for developing the rankings. RStudio is an integrated development environment (IDE) for R that was developed by JJ Allaire. This software contains tools that make programming in R easier.

RStudio extends R's capabilities by making it easier to import data, write scripts, and generate visualizations and reports. The company RStudio (now Posit since 2022) was founded in 2009 with the main goal of "creating high quality open-source software for data scientists."

1.3.2 Uses of R and RStudio

i. The R and RStudio console can be used as a complex scientific calculator.

ii. The values of various data types can be assigned to variables using the symbol `<-` or `=`.

iii. Built-in functions can be used to manipulate variables.

iv. Built-in datasets can be accessed internally for analysis.

v. New datasets can be imported and new functions can be created for custom analysis.

vi. To aid in computational analysis, there exists a large package library (CRAN), as well as a lot of software in development to aid in computational analysis.

1.3.3 Primary advantages of R and RStudio

i. R and RStudio are free and open-source software programs which makes them accessible to anyone with a computer and internet connection. This accessibility is key in enabling learners from all socioeconomic levels and geographic regions to have a chance to work with statistical software,

ii. A large number of user communities exists for the R/RStudio software. These communities (listed in the Appendix) provide learning support and assist with technical challenges,

iii. Numerous freely available packages/extensions have been developed by the R and RStudio user communities to facilitate all forms of computational analysis, visualization, and publication. The (CRAN) has packages that contain datasets as well as allow one to perform statistical analysis and data visualization,

iv. R and RStudio allow for reproducible analysis where scripts and workflows can be shared with fellow users, and,

v. The R/RStudio software is cross-platform, which means that it can be used on Linux, Windows, and Mac operating systems.

1.3.4 Applications of R in different industries

i. Bioinformatics and Healthcare: epidemiological studies, clinical trial analysis, and genetic data analysis.

ii. Financial Modeling and Risk Analysis: risk management, algorithmic trading, trading strategies and analysis, time series analysis, and portfolio optimization.

iii. Retail and Marketing: customer analytics, sales forecasting, market research, web analytics, and customer segmentation.

iv. Social Sciences and Humanities: text analysis, surveys and opinion research, social trend analysis, and policy analysis.

v. Statistics and Data Analysis: hypothesis testing, data visualization, regression modelling, and statistical inference.

vi. Environmental Science and Climate Change: forecasting weather patterns, modelling climate change, monitoring pollution levels, and ecological modelling.

1.4 Exercises

As you embark on your R/RStudio learning journey, I have listed (below) a few questions for you to think about before we get started with the lessons.

i. Why do you want to learn R and RStudio?

ii. Do you currently use any other software tools for data analysis and visualization? What are the limitations of these tools?

iii. What are some key differences between R and other statistical programming languages like SAS or SPSS?

iv. What tasks do you hope to accomplish after completing this training?

v. Explore the various R/RStudio communities listed in the appendix and consider joining any one of them. What is the role of the R community in the development and support of R?

vi. Browse some of popular R packages (on CRAN or R-Universe) for different tasks like data visualization and statistical analysis. Pick one package that interests you and read about its capabilities.

1.5 Summary

Overall, I hope you enjoyed learning about the history of R and RStudio, and have seen the advantages of using these software tools. Additionally, we discussed the numerous applications of R in various industries. In the next chapter, we will look at how to download and install both R and RStudio on your local computer.

2 Download and Install R and RStudio

2.1 Questions

- How does one install R and RStudio on their personal computer?
- Can RStudio be used online via a cloud-based service?
- Is it possible to work with alternative code editors when using R?
- What steps are involved in installing RStudio?

2.2 Learning Objectives

- Install the R programming language on your local machine.
- Install RStudio as an integrated development environment (IDE) for R on your local machine.
- Create an RStudio Cloud Account on the Posit Website.
- Learn how to use alternative code editors.

2.3 Lesson Content

In progress

2.4 Exercises

i. Download and install R and RStudio (*If you have not already done so*).
ii. Describe the steps involved in installing R on a Windows operating system.
iii. Are there any specific considerations when installing R on a macOS or Linux system?
iv. Set up a free RStudio Cloud Account on the Posit Website.
v. Compare the local RStudio with the cloud-based RStudio and list the potential benefits/disadvantages of both.

vi. Verify your R installation by running a simple R script that prints "Hello, R! My name is (fill in the blank)" to the console.
vii. Can you customize the appearance or behavior of RStudio according to your preferences?

2.5 Summary

In progress

3 Navigating the R and RStudio interfaces

3.1 Questions

- What are the main components of the R interface?
- How is the RStudio IDE organized, and what are its key features?
- How does one navigate the R and RStudio interfaces?
- What are the functions of the various RStudio panes?

3.2 Learning Objectives

- Identify and understand the main components of the R interface.
- Navigate the RStudio IDE and comprehend its organizational structure.
- Customize your RStudio environment to suit your preferences.
- Understand the roles of the console and menu options in the R software.
- Review the various panes and menu options in the RStudio software.
- Learn the various keyboard shortcuts that can improve speed and efficiency.

3.3 Lesson Content

In progress

3.4 Exercises

i. Describe the main components of the RStudio interface.
ii. Open RStudio and explore the different panels of the interface, then describe in your own words the purpose of the Environment, History, Files, and Plots panes.
iii. What is the purpose of the Console pane in RStudio?
iv. Access the help documentation using the Help pane and find information on specific R functions/packages.
v. Customize your RStudio environment by changing the theme and adjusting the appearance settings.
vi. Practice using keyboard shortcuts for common tasks in RStudio.
vii. How can you customize the layout of panes in RStudio?

3.5 Summary

In progress

4 Managing your files and data

4.1 Questions

- Why is good file and data management important in R?
- What is the concept of a project in R, and how does it help in organizing your work?
- What is the purpose of creating R project folders and scripts?
- What is the significance of a working directory in R, and how does it impact your workflow?
- How do you create, save, and run R scripts for efficient code management?

4.2 Learning Objectives

- Understand the importance of properly organized files and datasets for efficient R projects.
- Understand the concept of R projects and their role in organizing your work.
- Write and run R code in scripts for better code organization and reproducibility.
- Master the concept of working directories and navigate your project files effectively.

4.3 Lesson Content

In progress

4.4 Exercises

i. Create a new RStudio project called "novice_programmer_guide".

ii. Within the project folder, create a new R Script called "introduction.R" and save it.

iii. Set your working directory in the console to the root of this new project, and practice loading data from a file within it.

iv. Explain the significance of using relative and absolute file paths in R, and experiment with using relative paths to access different data files within your project directory.

v. Explain the role of the functions, file.path(), list.files(), and dir.create(), in R.

vi. Describe the purpose of the .Rproj file in an R project.

vii. How can you clear the R workspace? Explore functions like rm() and rm(list=ls()) to remove unwanted objects.

4.5 Summary

In progress

5 Importing data and saving analysis outputs

5.1 Questions

- Can users access and use internal datasets for analysis in R? How do I load the datasets?
- How does one import external datasets into R? What functions and packages are commonly used for this purpose?
- How do I handle different data formats such as CSV, Excel, or text files?
- Once the user completes their data analysis, how can their output(s) be saved?
- How can I create new data objects or modify existing ones within R?

5.2 Learning Objectives

- Learn how to load and access internal R datasets.
- Import external datasets into your R project folder.
- Demonstrate how to save analysis outputs from R such as tables and datasets.
- Understand methods for importing data stored internally in R and from external sources.
- Utilize built-in functions and packages to read various data formats like CSV, Excel, and text files.

5.3 Lesson Content

In progress

5.4 Exercises

i. Import datasets from different formats like CSV, Excel, and text files into your R environment using the base R functions such as read.*().

ii. Save your file using a new name with write.*(), save(), or .RData?

iii. How can you check the structure and summary statistics of an imported dataset in R?

iv. What options do you have when importing data from non-local sources like URLs or databases? Explore functions like url() and dbConnect() for remote data access.

v. How can you specify import options for different file formats? Learn about arguments like sep, header, and na.strings for customized data reading.

vi. What is the difference between overwriting a dataset and appending new data to it in R?

5.5 Summary

In progress

6 Basic arithmetic, arithmetic operators, and variables

6.1 Questions

- How can we perform basic arithmetic operations in R?
- What arithmetic operators can be used in R?
- How do I assign values to variables?
- Are there naming conventions/variable style guides for creating new variables?

6.2 Learning Objectives

- Learn how to use R to perform basic arithmetic.
- Declare and manipulate variables to store and retrieve data in R.
- Master basic arithmetic operations like addition, subtraction, multiplication, and division in R.
- Understand the concept of variables and assign numeric values in your R code.
- Use appropriate naming conventions for variables.
- Differentiate between different variable types (numeric, integer, character) and choose appropriate ones.

6.3 Lesson Content

In progress

6.4 Exercises

i. In R, calculate 3 + 5 and then 4 * 6.
ii. Assign the value 10 to a variable called x. Then calculate x ^ 2. Next, calculate the expression (3 + x) * (x - 1) using the x from before.
iii. Declare a variable to store your age and assign a value to it. Print the variable value.
iv. Create variables for the length and width of a rectangle and calculate its area.
v. Explain the rules for naming variables in R. Provide examples of valid and invalid variable names.
vi. What is operator precedence in R? How does it work for arithmetic operators?
vii. What data types can be used for arithmetic operations in R?
viii. What is the difference between <- and = for assignment in R?

6.5 Summary

In progress

7 The primary types of operators in R

7.1 Questions

- What are the primary operator types in R?
- How can one use the various operator types in data analysis?
- How do arithmetic, comparison, and logical operators work in R?
- When should I use assignment operators versus comparison operators?

7.2 Learning Objectives

- Identify the primary operator types in R.
- Learn how to use various operators for data manipulation.
- Utilize arithmetic operators for efficient numerical calculations in your R scripts.
- Apply comparison operators to evaluate conditions and filter data based on logical comparisons.
- Employ logical operators to combine multiple conditions and control the flow of your R code.

7.3 Lesson Content

In progress

7.4 Exercises

- List the basic operator types in R and provide an example for each.
- What operators allow you to subset vectors, lists, and data frames in R?
- Explain how the modulus operator (%/%) differs from the regular division operator (/).
- What are the membership operators %in% and !%in% used for in R?
- Assign 10 to x using <-. Then compare if x == 10.
- Calculate 3^4 using arithmetic operators.
- Compare 3 > 5. Is the result TRUE or FALSE?
- When would you use the sequence operator (:) in R? Give an example.

7.5 Summary

In progress

8 Data Types

8.1 Questions

- What are the primary data types in R?
- Can one convert between different data types?
- How does R handle different types of data?
- How can I determine the type of a variable in R?
- What are some helpful functions for identifying and manipulating data types?
- How do data types impact my analysis and coding in R?

8.2 Learning Objectives

- Identify and differentiate between common data types in R.
- Learn how to convert between different data types.
- Use functions like class(), typeof(), and str() to inspect types.
- Convert between data types using the as.* functions.

8.3 Lesson Content

In progress

8.4 Exercises

- What are the basic data types in R? Give examples of each.
- How can you check the data type of a variable in R?
- Use the class(), typeof(), and str() functions to check the data type of objects.
- Describe the process of coercion in R.
- Provide an example of implicit and explicit coercion between data types.
- How can you coerce or convert between different data types in R?
- What does it mean for an object in R to have attributes? What attributes are commonly used?

8.5 Summary

In progress

9 Vectors

9.1 Questions

- What are vectors and why are they essential in R?
- What are the different types of vectors and their key characteristics?
- How does one create a vector?
- What operations can be performed on a vector?
- How do I create and access different elements within a vector?
- Can I combine and manipulate vectors to achieve specific tasks?
- What are some common functions for working with vectors?

9.2 Learning Objectives

- Define and create vectors in R.
- Understand how to perform basic operations on vectors.
- Understand the concept of vectors as fundamental data structures in R.
- Master the creation and manipulation of vectors using various techniques.
- Identify and differentiate between different types of vectors based on their elements.
- Utilize functions and operators to combine, subset, and transform vectors with ease.
- Apply your understanding of vectors to enhance your R coding and data analysis efficiency.

9.3 Lesson Content

In progress

9.4 Exercises

i. Create a numeric vector with values 1 through 5 using c() and extract the middle element from the vector using the described subsetting methods.
ii. Generate a sequence of even numbers from 2 to 10 using the seq() function and calculate the total sum and average of this new vector.
iii. Create a vector of student ages (between 16 and 20, n = 10), calculate their average and standard deviation, and identify students younger than 18.
iv. Use logical indexing to subset elements from a vector based on a condition.
v. Perform element-wise addition and multiplication on two newly created numeric vectors of the same length.
vi. Concatenate two newly created character vectors to create a longer vector.
vii. Create a named vector with elements representing days of the week.
viii. Use vector functions to find the length, sum, and mean of a numeric vector.
ix. Use functions like `sort()` or `rev()` to arrange elements in ascending, descending, or reversed order.
x. Use functions like `which()` or `match()` to locate elements based on conditions or matching values.
xi. How can you append elements to an existing vector?
xii. When binding vectors together, what is the difference between cbind() and rbind()?
xiii. Explore other vector capabilities like recycling, naming elements, and using vector lengths in your R code.

9.5 Summary

In progress

10 Data Structures (Part I)

10.1 Questions

- List the data structures that can be used in R?
- How do we define the various data structures?
- What operations can be performed on the different data structures?
- How do these data structures differ from one another in terms of storage and functionality?

10.2 Learning Objectives

- Learn about data structures in R, specifically vectors, lists, and dataframes.
- Define and manipulate vectors, lists, and dataframes.
- Perform common operations on each data structure.
- Identify and differentiate between the diverse data structures offered by R.
- Choose the appropriate data structure based on the type and organization of your data.
- Understand the strengths and limitations of each structure for efficient analysis.

10.3 Lesson Content

In progress

10.4 Exercises

i. Explain the characteristics and use cases of lists in R.

ii. Create a list containing a numeric vector, character vector, and logical vector.

iii. How can you add elements to a list in R?

iv. Explain what a data frame is and why it is widely used in R.

v. Write code to create a data frame with columns for "Name," "Age," and "Gender" for three individuals.

vi. How can you combine two data frames horizontally in R?

vii. Discuss various methods for subsetting elements in data structures.

viii. How do you check the structure of a data frame? What function is used?

ix. How do you access a specific column of a data frame?

x. What are rownames and colnames in a data frame? How are they useful?

xi. How do you add a new column to an existing data frame?

10.5 Summary

In progress

11 Data Structures (Part II)

11.1 Questions

- List the data structures that can be used in R?
- How do we define the various data structures?
- What operations can be performed on the different data structures?
- How do these data structures differ from one another in terms of storage and functionality?

11.2 Learning Objectives

- Learn about data structures in R, specifically matrices, arrays, and factors.
- Define and manipulate matrices, arrays, and factors.
- Perform common operations on each data structure.
- Identify and differentiate between the diverse data structures offered by R.
- Choose the appropriate data structure based on the type and organization of your data.
- Understand the strengths and limitations of each structure for efficient analysis.

11.3 Lesson Content

In progress

11.4 Exercises

- How do you create a matrix in R? Give an example.
- What are the column and row dimensions of a matrix called in R?
- How do you access elements in a matrix?
- Generate a 3x3 identity matrix using matrix().
- Write code to create a 3x3 matrix with sequential numeric values.
- Convert a character vector to a factor with 3 levels.
- Describe the purpose of factors in R.
- How do you create a factor variable with specified levels?
- What is an array, and how does it differ from a matrix in R?
- Provide an example of creating a three-dimensional array.

11.5 Summary

In progress

12 Questions

Handling missing data {#sec-missing}

- What is missing data?
- Why is missing data a common challenge in data analysis?
- How can we identify and handle missing data effectively in R?
- What are the consequences of ignoring or mishandling missing data?
- What are the best practices for imputing missing values?
- When should I choose deletion, imputation, or alternative approaches?

12.1 Learning Objectives

- Define the main types of missing data.
- Identify missing data in R and assess its impact on analyses.
- Work with datasets that contain missing data.
- Understand the importance of addressing missing data in data analysis.
- Apply best practices for handling missing data in your R projects.
- Apply functions like is.na() and complete.cases() to identify NA values.

12.2 Lesson Content

In progress

12.3 Exercises

i. Identify missing values in a dataset using the `is.na()` and `complete.cases()` functions.

ii. Practice removing missing data using the `na.omit()` function.

iii. How can you handle missing values when calculating summary statistics like mean or median?

iv. If you take the mean of a vector with `NA` values, what will the result be?

v. How are missing values represented in R?

vi. What is `NA` in R? How is it different from `NULL`?

vii. What are some common reasons you may get `NA` values in a dataset?

viii. How can you check if a value is `NA` in R?

12.4 Summary

In progress

Conclusion

We have finally come to the end of the "R for Novice Programmers" book. Congratulations on completing this learning journey!

The overall goal of this book was to introduce non-programmers or those with very little programming experience to R and RStudio and their use in basic statistical programming. It is hoped that the learners gained new skills and discovered how this software could be used to simplify everyday tasks in their workplaces. The learner should now be able to understand all the topics listed below based on the previously described learning objectives.

Table 12.1: Syllabus

Chapter	Title	Date Completed
	Introduction	
1	Overview of R and RStudio	
2	Download and Install R and RStudio	
3	Navigating the R and RStudio interfaces	
4	Managing your files and data	
5	Importing data and saving analysis outputs	
6	Basic arithmetic, arithmetic operators, and variables	
7	The primary types of operators in R	
8	Data Types	
9	Vectors	
10	Data Structures (Part I)	
11	Data Structures (Part II)	
12	Handling missing data	
	Conclusion	
	Appendix	

Now that the learner has mastered the basics, it is time to transition to specific topics such as data wrangling/manipulation, data visualization, and programming. Therefore, I would encourage the learner to review some of the links I have provided in the Appendix and look out for more books in the future which will cover:

i. Data visualization

ii. Tidyverse

iii. Programming: Functions, Loops, and Control Statements

iv. Data Science

v. Statistics

vi. Machine Learning and Artificial Intelligence

vii. Publishing with R Markdown and/or Quarto

viii. R Shiny

Thanks for your participation and for completing this book!

Appendix

To further your R learning journey, I would recommend a review of the following freely available resources. This list will constantly be updated as new material becomes available.

1. Online repositories with R/RStudio-related links

- Big Book of R
- Free R Reading Material
- Awesome R 1
- Awesome R 2
- Awesome R 3
- Awesome R 4
- resouRces: Database of Resources to Learn & Teach R
- R-universe
- R Community Public Library

2. R Communities

- Posit (formerly RStudio) Community
- Stack Overflow R
- Stack Overflow RStudio
- Cross Validated
- R-bloggers
- R for Data Science Slack Group
- R User Community
- R-Ladies Global
- satRday
- rOpenSci
- **Twitter (now X) – #rstats and #rstudio**
- **Reddit – r/rstats and r/RStudio**

3. Conferences and Meetups

- R Foundation Conferences (useR! and DSC)
- R Conference
- Posit Conference
- R User Group Meetups

4. Cheatsheets

- R Views Cheatsheets
- Posit Cheatsheets
- 30 Essential Data Science, Machine Learning & Deep Learning Cheat Sheets

5. Tutorials

- Swirl: Learn or Teach R, in R
- RStudio Education Learning
- RStudio Education Teaching
- R Workflow
- R Screencasts
- Quick-R
- RTutor: Interactive R Problem Sets

6. Newsletters and Blogs

- R Weekly
- Data Elixir
- Revolution Analytics Blog

7. Data Visualization

- The R Graph Gallery
- R Charts
- ggplot2 extensions
- Telling Stories with Data
- R Graphics Cookbook
- Fundamentals of Data Visualization

- ggplot2: Elegant Graphics for Data Analysis
- Data Visualization: A practical introduction
- Modern Data Visualization with R

8. Data Science

- R for Data Science
- Data 8: The Foundations of Data Science
- Introduction to Data Science
- Readings in Applied Data Science
- Elements of Data Science
- The Art of Data Science
- Data Science Cheatsheet
- Introduction to Data Science (for not-yet scientists)
- data.org
- Data Basic
- Data Science in a Box
- Data Literacy

9. Statistics

- An Introduction to Statistical Learning with R or Python
- Modern Statistics with R
- Introduction to Modern Statistics
- Statistical Thinking
- Introduction to Modern Statistics
- Biostatistics for Biomedical Research
- STA 212: Statistical Models
- STA 312: Linear Models
- STA 363: Statistical Learning
- Introduction to Probability for Data Science
- Statistics 431: Advanced Statistical Computing with R
- Statistics and R
- Tidy Statistics in R - Datamethods
- Statistical Computing

10. Datasets

- Awesome Public Datasets

- Dataverse Project
- Data Commons
- Open Africa Data
- Data Africa
- Our World in Data
- California Data Sources
- DataSF
- NYC Open Data
- The Humanitarian Data Exchange
- Global Data Barometer
- Microsoft Research Open Data
- Kaggle Datasets
- Google Research Datasets
- NHS R-community Datasets
- IHME
- Best Public Datasets for Health Data Science Projects
- World Bank Open Data

11. Tidyverse

- A very short introduction to Tidyverse
- Tidyverse Workshop Series
- Tidyverse Skills for Data Science
- Eight R Tidyverse tips for everyday data engineering
- Tidyverse Tips
- An Introduction to R through the tidyverse
- Modern R with the tidyverse
- C'est quoi, le tidyverse?
- Transitioning into the tidyverse (part 1)
- Transitioning into the tidyverse (part 2)

12. Programming: Functions, Loops, and Control Statements

- Control Structures
- Apply Functions
- Defining your own functions
- Functional programming 1
- Functional programming 2
- Statistical Programming Paradigms and Workflows
- Hands-On Programming with R

- purr tutorial
- Unlocking the Power of Functional Programming in R (Part 1)
- Unlocking the Power of Functional Programming in R (Part 2)

13. Machine Learning and Artificial Intelligence

- Hands-On Machine Learning with R
- Create machine learning models: An R version
- Interpretable Machine Learning
- Supervised Machine Learning for Text Analysis in R
- Posit AI Blog
- Tidy Modeling with R
- Tidymodels
- A Gentle Introduction to tidymodels
- The Case for tidymodels
- ISLR tidymodels labs
- Introduction to Machine Learning with the Tidyverse
- tidypredict

14. Publishing

- Quarto
- R Markdown
- Awesome Quarto

15. R Shiny

- Shiny Learning Resources
- Shiny Tutorials
- Mastering Shiny
- Building Web Applications WITH SHINY
- a gRadual intRoduction to Shiny
- Engineering Production-Grade Shiny Apps
- ShinyUI Editor
- Shiny Examples
- An Intro to Shiny
- Shiny Gallery
- shinydashboard
- Application layout guide