

# **Statistical Inference For Data Science**

Saul Diaz-Infante

1/16/23

# Table of contents

<b>Preface</b>	<b>4</b>
<b>Introduction</b>	<b>5</b>
The tidyverse . . . . .	5
<b>1 R programming fundamentals for Data Science</b>	<b>9</b>
1.0.1 Entering Input: the assignment operator . . . . .	9
1.1 Intro Basics . . . . .	9
1.1.1 How it works . . . . .	10
1.1.2 Arithmetic with R . . . . .	10
1.1.3 Variable assignment . . . . .	11
1.1.4 Variable assignment (2) . . . . .	12
1.1.5 Variable assignment (3) . . . . .	12
1.1.6 Apples and oranges . . . . .	13
1.1.7 Basic data types in R . . . . .	14
1.1.8 Instructions 100 XP . . . . .	14
1.1.9 What's that data type? . . . . .	15
1.2 Vectors . . . . .	16
1.2.1 Create a vector . . . . .	16
1.2.2 Create a vector (2) . . . . .	16
1.3 Best Coding Practices for R . . . . .	17
1.3.1 What we mean when say "better coding practice" . . . . .	17
1.3.2 Folder Structure . . . . .	18
1.3.3 Code Structure . . . . .	18
1.3.4 Sections . . . . .	18
1.3.5 Structural Composition . . . . .	18
1.3.6 Indentation . . . . .	18
1.3.7 Styling . . . . .	18
1.3.8 Final Comments . . . . .	18
<b>2 Data visualization with ggplot2 and friends</b>	<b>19</b>

<b>II</b>	<b>The whole game of statistical Inference</b>	<b>20</b>
<b>3</b>	<b>Statistical Inference with resampling: Bootstrap and Jackknife.</b>	<b>22</b>
3.1	Likelihood inference. . . . .	22
3.2	Variance analysis. . . . .	22
3.3	ROC Curves . . . . .	22
<b>4</b>	<b>Linear Regression</b>	<b>23</b>
4.1	Linear Regression . . . . .	23
4.2	Multiple linear regression and generalized linear regression . . . . .	23
<b>5</b>	<b>Summary</b>	<b>24</b>
	<b>References</b>	<b>25</b>

# Preface

- Who I am. I am Saul Diaz Infante Velasco. I just starting as assistant professor at the Data Science graduate program of Universidad de Sonora at Hermosillo Mexico. My Background is related with numerical analysis and stochastic models. I'm are a enthusiastic of this treading topic called Data-Science, but perhaps at the moment I only have just intuition about what really it is. However, I have been programming almost 20 years an moved from old programming langues as FORTRAN, Pascal, Basic, Cobol, C, C++ to the new well established treading development workflows like R, Python and Julia. This is my firs attempt in R.
- What the book is about.
- When I writing this book.
- Why I write this book.
- Where I wrote this book.

# Introduction

The focus of this course is into the programming and basic techniques for inference that are usually applied in data science. We start by reviewing and enforcing programming skills. Then we will use the database of entomological data practice and build the required bases for more structured tools like bootstrap or Jackknife cuts.

Figure 1 further explores the impact of temperature on ozone level.

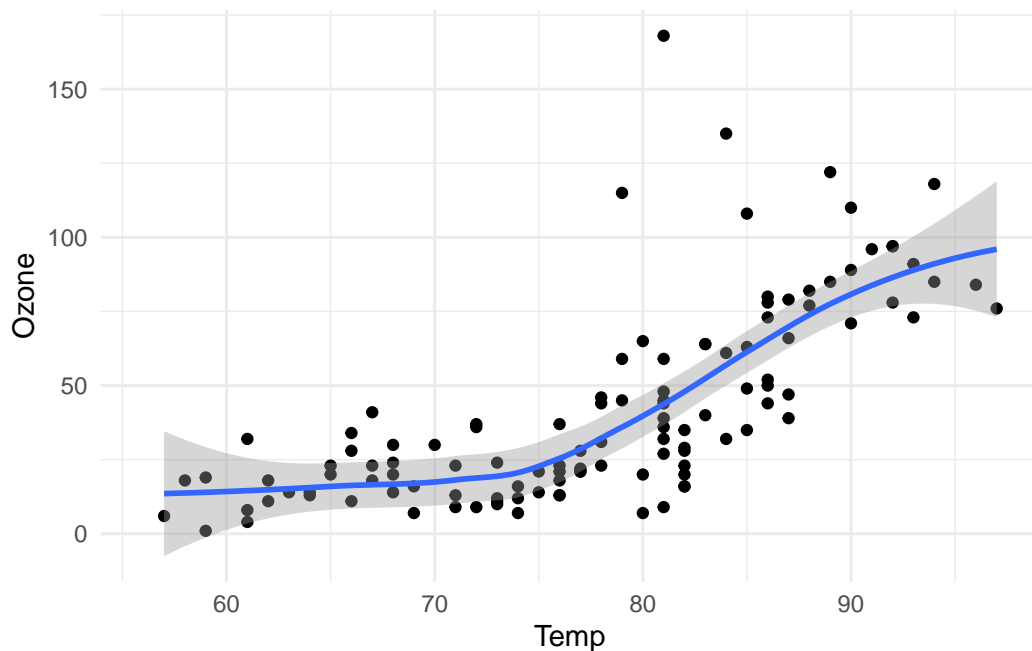


Figure 1: Temperature and ozone level.

## The tidyverse

We need to install a R package. The majority of the packages that we will use are part of the so-called tidyverse package. The packages in the tidyverse share a common philosophy of data and R programming, and are designed to work together naturally.

You can install the complete tidyverse with the line of code:

then we can use it by loading in the preamble section with

```
-- Attaching packages ----- tidyverse 1.3.2 --
v tibble  3.1.8      v dplyr   1.0.10
v tidyr   1.2.0      v stringr 1.5.0
v readr   2.1.2      v forcats 0.5.2
v purrr   0.3.4
-- Conflicts ----- tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()     masks stats::lag()
```

see <https://www.tidyverse.org/> documentation.

## Part I

We dedicate this part to overview the basics to program in R. The aim of this part is building the basis for Machine learning, namely **data visualization**, **data manipulation** and the **good coding practices** to type script of industrial production quality.



# 1 R programming fundamentals for Data Science

## 1.0.1 Entering Input: the assignment operator

The thing that we type on the R console prompt are expressions. The first expression we discuss here is the assignment operator, please watch the following video [https://www.youtube.com/watch?v=vGY5i\\_J2c-c&t=283s](https://www.youtube.com/watch?v=vGY5i_J2c-c&t=283s)

At the R console, any executable typed text that we put a side of the prompt are called expressions. We start by the `<-` symbol is the assignment operator.

```
[1] 0
```

```
[1] 0
```

```
[1] "what's up"
```

The `[1]` shown in the output indicates that `x` is a vector and `0` is the element at position with index `1`.

## 1.1 Intro Basics

Take your first steps with R. In this chapter, you will learn how to use the console as a calculator and how to assign variables. You will also get to know the basic data types in R. Let's get started. The below lines have been taken from the course Introduction to R of DataCamp. You can see the first chapter for free. I recommend this course to get started on R see <https://campus.datacamp.com/courses/free-introduction-to-r>

### 1.1.1 How it works

In the text editor you should type R code to solve the exercises. When you hit **ctrl + enter**, every line of code is interpreted and executed by R and you get a message whether or not your code was correct.

R makes use of the `#` sign to add comments, so that you and others can understand what the R code is about. Comments are not run as R code, so they will not influence your result. For example, Calculate  $3 + 4$  in the editor on the right is a comment.

You can also execute R commands straight in the console. This is a good way to experiment with R code.

#### 1.1.1.1 Instructions 100 XP

- In the text editor on the right there is already some sample code.
- Can you see which lines are actual R code and which are comments?
- Add a line of code that calculates the sum of 6 and 12, and hit the enter button

ex\_01.R

```
# Calculate 3 + 4
3 + 4
# Calculate 6 + 12
6 + 12
```

### 1.1.2 Arithmetic with R

In its most basic form, R can be used as a simple calculator. Consider the following arithmetic operators:

- Addition: `+`
- Subtraction: `-`
- Multiplication: `*`
- Division: `/`
- Exponentiation: `^`
- Modulo: `%%`

The last two might need some explaining:

- The `^` operator raises the number to its left to the power of the number to its right: for example  $3^2$  is 9.

- The modulo returns the remainder of the division of the number to the left by the number on its right, for example 5 modulo 3 or `5 %% 3` is 2.

### 1.1.2.1 Instructions 100 XP

- Type `2^5` in the editor to calculate 2 to the power 5.
- Type `28 %% 6` to calculate 28 modulo 6.
- Run the answer in the console and have a look at the R output .
- Note how the `#` symbol is used to add comments on the R code.

**ex\_02.R**

```
# An addition
5 + 5

# A subtraction
5 - 5

# A multiplication
3 * 5

# A division
(5 + 5) / 2


# Exponentiation
2 ^ 5

# Modulo
28 %% 6
```

### 1.1.3 Variable assignment

A basic concept in (statistical) programming is called a variable.

A variable allows you to store a value (e.g. 4) or an object (e.g. a function description) in R. You can then later use this variable's name to easily access the value or the object that is stored within this variable.

 You can assign a value 4 to a variable `my_var` with the command

```
my_var <- 4
```

### 1.1.3.1 Instructions 100 XP

Over to you: complete the code in the editor such that it assigns the value 42 to the variable `x` in the editor. Submit the answer. Notice that when you ask R to print `x`, the value 42 appears.

```
[1] 42
```

### 1.1.4 Variable assignment (2)

Suppose you have a fruit basket with five apples. As a data analyst in training, you want to store the number of apples in a variable with the name `my_apples`.

#### 1.1.4.1 Instructions 100 XP

- Type the following code in the editor: `my_apples <- 5`. This will assign the value 5 to `my_apples`.
- Type: `my_apples` below the second comment. This will print out the value of `my_apples`.
- Run your answer, and look at the output: you see that the number 5 is printed. So R now links the variable `my_apples` to the value 5.

`ex_04.R`

```
# Assign the value 5 to the variable my_apples
my_apples <- 5
# Print out the value of the variable my_apples
print(my_apples)
```

### 1.1.5 Variable assignment (3)

Every tasty fruit basket needs oranges, so you decide to add six oranges. As a data analyst, your reflex is to immediately create the variable `my_oranges` and assign the value 6 to it. Next, you want to calculate how many pieces of fruit you have in total. Since you have given meaningful names to these values,

**i** you can now code this in a clear way:

```
my_apples + my_oranges
```

### 1.1.5.1 Instructions 100 XP

- Assign to `my_oranges` the value 6.
- Add the variables `my_apples` and `my_oranges` and have R simply print the result.
- Assign the result of adding `my_apples` and `my_oranges` to a new variable `my_fruit`.

**ex\_05.R**

```
# Assign a value to the variables my_apples and my_oranges
my_apples <- 5
my_oranges <- 6

# Add these two variables together
my_apples + my_oranges

# Create the variable my_fruit
my_fruit <- my_apples + my_oranges
```

### 1.1.6 Apples and oranges

Common knowledge tells you not to add apples and oranges. But hey, that is what you just did, no :-)? The `my_apples` and `my_oranges` variables both contained a number in the previous exercise. The `+` operator works with numeric variables in R. If you really tried to add “apples” and “oranges”, and assigned a text value to the variable `my_oranges` (see the editor), you would be trying to assign the addition of a numeric and a character variable to the variable `my_fruit`. This is not possible.

#### 1.1.6.1 Instructions 100 XP

- Run the answer and read the error message. Make sure to understand why this did not work.
- Adjust the code so that R knows you have 6 oranges and thus a fruit basket with 11 pieces of fruit.

**ex\_06.R**

```
# Assign a value to the variable my_apples
my_apples <- 5
# Fix the assignment of my_oranges
my_oranges <- "six"
```

```
# Create the variable my_fruit and print it out
my_fruit <- my_apples + my_oranges
my_fruit
```

Response

ex\_\_06.R

```
# Assign a value to the variable my_apples
my_apples <- 5
# Fix the assignment of my_oranges
my_oranges <- 6
# Create the variable my_fruit and print it out
my_fruit <- my_apples + my_oranges
my_fruit
```

### 1.1.7 Basic data types in R

R works with numerous data types. Some of the most basic types to get started are:

- Decimal values like 4.5 are called numerics.
- Whole numbers like 4 are called integers. Integers are also numerics.
- Boolean values (TRUE or FALSE) are called logical.
- Text (or string) values are called characters.

Note how the quotation marks in the editor indicate that “some text” is a string.

### 1.1.8 Instructions 100 XP

Change the value of the:

- `my_numeric` variable to 42.
- `my_character` variable to "universe". Note that the quotation marks indicate that “universe” is a character.
- `my_logical` variable to FALSE.

**i** Note that R is case sensitive!

Thus despite the variables called `var`, `Var`, `vAr`, has the same fonetic characters, R understand each of these as different memory addresses.

ex\_\_07.R

```
# Change my_numeric to be 42
my_numeric <- 42.5

# Change my_character to be "universe"
my_character <- "some text"

# Change my_logical to be FALSE
my_logical <- TRUE
```

Response

ex\_07.R

```
# Change my_numeric to be 42
my_numeric <- 42

# Change my_character to be "universe"
my_character <- "universe"

# Change my_logical to be FALSE
my_logical <- FALSE
```

### 1.1.9 What's that data type?

Do you remember that when you added  $5 + \text{"six"}$ , you got an error due to a mismatch in data types? You can avoid such embarrassing situations by checking the data type of a variable beforehand. You can do this with the `class()` function, as the code in the editor shows.

#### 1.1.9.1 Instructions 100 XP

Complete the code in the editor and also print out the classes of `my_character` and `my_logical`.

ex\_08.R

```
# Declare variables of different types

my_numeric <- 42
my_character <- "universe"
my_logical <- FALSE
# Check class of my_numeric
```

```
class(my_numeric)

# Check class of my_character
class(my_character)

# Check class of my_logical
class(my_logical)
```

## 1.2 Vectors

### 1.2.1 Create a vector

Feeling lucky? You better, because this chapter takes you on a trip to the City of Sins, also known as Statisticians Paradise!

Thanks to R and your new data-analytical skills, you will learn how to uplift your performance at the tables and fire off your career as a professional gambler. This chapter will show how you can easily keep track of your betting progress and how you can do some simple analyses on past actions. Next stop, Vegas Baby... VEGAS!!

#### 1.2.1.1 Instructions 100 XP

- Do you still remember what you have learned in the first chapter? Assign the value "Go!" to the variable vegas. Remember: R is case sensitive!

ex\_\_08.R

```
# Define the variable vegas
vegas <- "Go!"
```

### 1.2.2 Create a vector (2)

Let us focus first!

On your way from rags to riches, you will make extensive use of vectors. Vectors are one-dimension arrays that can hold numeric data, character data, or logical data. In other words, a vector is a simple tool to store data. For example, you can store your daily gains and losses in the casinos.



In R, you create a vector with the combine function `c()`. You place the vector elements separated by a comma between the parentheses.

**i** For example:

```
numeric_vector <- c(1, 2, 3)
character_vector <- c("a", "b", "c")
```

Once you have created these vectors in R, you can use them to do calculations.

#### 1.2.2.1 Instructions 100 XP

Complete the code such that `boolean_vector` contains the three elements: `TRUE`, `FALSE` and `TRUE` (in that order).

ex\_09.R

```
numeric_vector <- c(1, 10, 49)
character_vector <- c("a", "b", "c")

# Complete the code for boolean_vector
boolean_vector <- c(TRUE, FALSE, TRUE)
```

### 1.2.3

## 1.3 Best Coding Practices for R

### 1.3.1 What we mean when say “better coding practice”

R programmers have a bad reputation writing bad code. Perhaps the main reason is that the people whose write much of the package are not programmers but scientific from other areas. Sometimes we overestimate crucial aspects from a programming standpoint. As R programmers we overcome to write the code for production. Mostly we write scripts and when we deploy it the same when we just wrap it in a function and perhaps a package. It is common to face poorly written code—**columns were referred by numbers, functions were dependent upon global environment variables, 50+ lines functions without arguments and with over-sized lines code 100 characters or more, not indentation, poor naming, conventions etc.,...**

We strongly encourage to use a style. Yea I know, there is not a unique way to do it, but the philosophy is to follow a consistent style. With respect to this regard made yourself a favor and read this great book for R

<https://bookdown.org/content/d1e53ac9-28ce-472f-bc2c-f499f18264a3/>

### **1.3.2 Folder Structure**

### **1.3.3 Code Structure**

### **1.3.4 Sections**

### **1.3.5 Structural Composition**

### **1.3.6 Indentation**

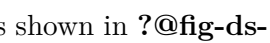
### **1.3.7 Styling**

### **1.3.8 Final Comments**

## **2 Data visualization with ggplot2 and friends**

## **Part II**

# **The whole game of statistical Inference**

Our goal in this part of the book is to give you a rapid overview of the main tools of data science: **importing**, **tidying**, **transforming**, and **visualizing data**, as shown in  **fig-ds-whole-game**. We want to show you the “whole game” of data science giving you just enough of all the major pieces so that you can tackle real, if simple, data sets. The later parts of the book, will hit each of these topics in more depth, increasing the range of data science challenges that you can tackle.

## **3 Statistical Inference with resampling: Bootstrap and Jackknife.**

**3.1 Likelihood inference.**

**3.2 Variance analysis.**

**3.3 ROC Curves**

## 4 Linear Regression

### 4.1 Linear Regression

### 4.2 Multiple linear regression and generalized linear regression

## 5 Summary

In summary, this book has no content whatsoever.

[1] 2



## References

- [1] T. Hastie, R. Tibshirani, J. Friedman, [The elements of statistical learning](#), Second, Springer, New York, 2009.
- [2] W.J. Krzanowski, D.J. Hand, [ROC curves for continuous data](#), CRC Press, Boca Raton, FL, 2009.
- [3] R. Martin, [A statistical inference course based on p-values](#), The American Statistician. 71 (2017) 128–136.
- [4] P. McCullagh, J.A. Nelder, [Generalized linear models](#), Chapman & Hall, London, 1989.
- [5] B. Ratner, [Statistical and machine-learning data mining:: Techniques for better predictive modeling and analysis of big data, third edition](#), CRC Press, 2017.
- [6] D.A. Sprott, Statistical inference in science, Springer-Verlag, New York, 2000.