# R Programming Basics and Data management in R

Pablo E. Gutierrez-Fonseca

Fall 2024

# What is R Programming?

- R is a free and open source software environment and programming language for statistics.
- R was created by Ross Ihaka and Robert Gentleman at the University of Auckland (in New Zealand), and is based on the S language that was created by John Chambers at Bell Laboratories.


University of Vermont

# What is R Programming?

- When you download and install R, you get a collection of basic packages (and libraries) that can be used to implement several common data manipulations, graphical displays, and statistical models.


University of Vermont

# Strengths and Weaknesses

- **Strengths:**
  -Free and Open Source.
  -Strong User Community.
  -Highly extensible, flexible.
  -Implementation of high-end statistical methods.
  -Flexible graphics and intelligent defaults.
- **Weakness:**
  -Steep learning curve.
  -Slow for large datasets.

University of Vermont

# Why R?

- **Computing power**: R can handle much larger datasets than traditional programs, which is especially important for long-term ecological data, community science data, and public health data.
- **Flexibility & convenience**: Traditionally, scientists would need separate programs for data cleaning, statistical procedures, and data visualization.
  Learning R means you can do everything in a single program, and customize to fit the goals of your project.
- **Reproducibility**: Saving your code as an R script makes it easy for other scientists to see what you did, and repeat your methods.

University of Vermont

# R Overview

University of Vermont

# How to download?

- Google it using R or CRAN (Comprehensive R Archive Network)
  – http://www.r-project.org

University of Vermont

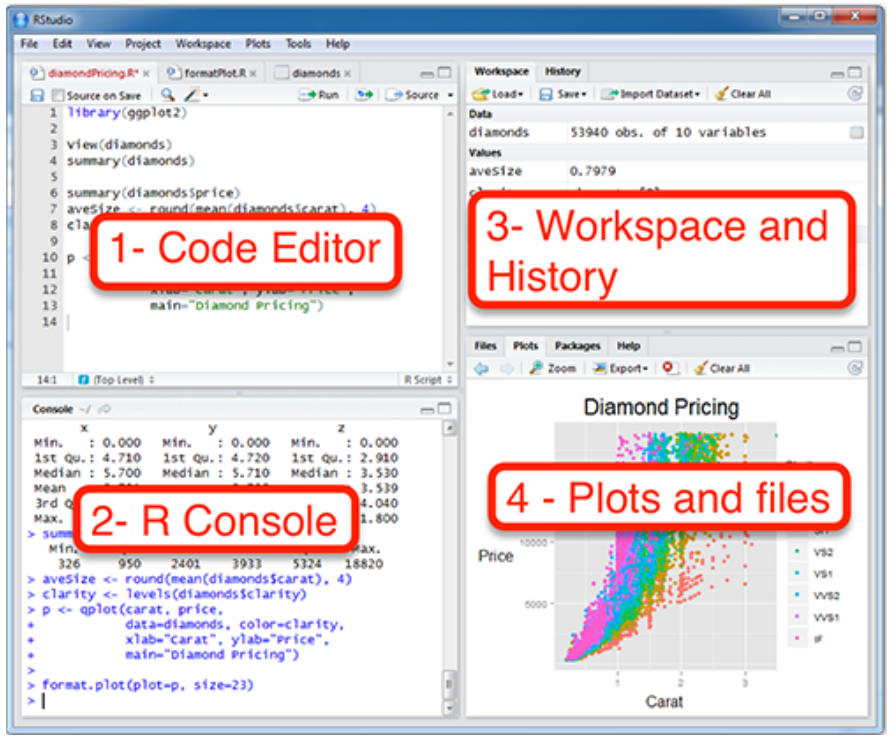
# How to download?

- Having installed R, the next thing we will want to do is install **RStudio**, a popular and useful interface for writing scripts and using R.
- If you google **RStudio** you will get to this window: https://www.rstudio.com/products/rstudio/download/
- Rstudio is an ***integrated development environment (IDE)*** for R that allows users to interact more easily with R by integrating different aspects of scripting, from code completion to debugging.

University of Vermont

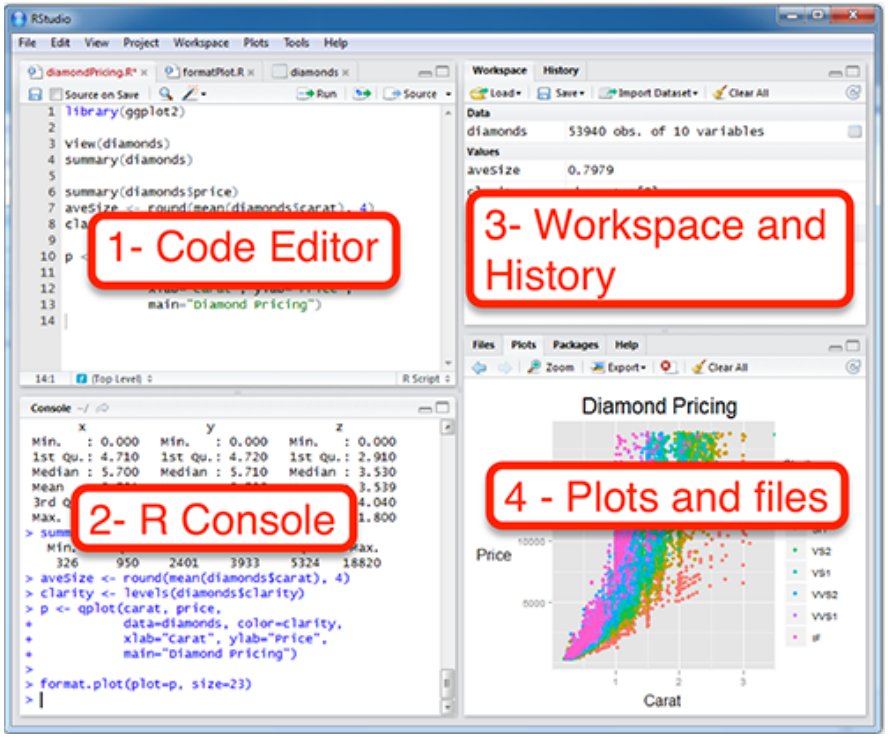
# Typical Rstudio session

- **Code Editor**: Contains code to tell R what to do. Save scripts for future use



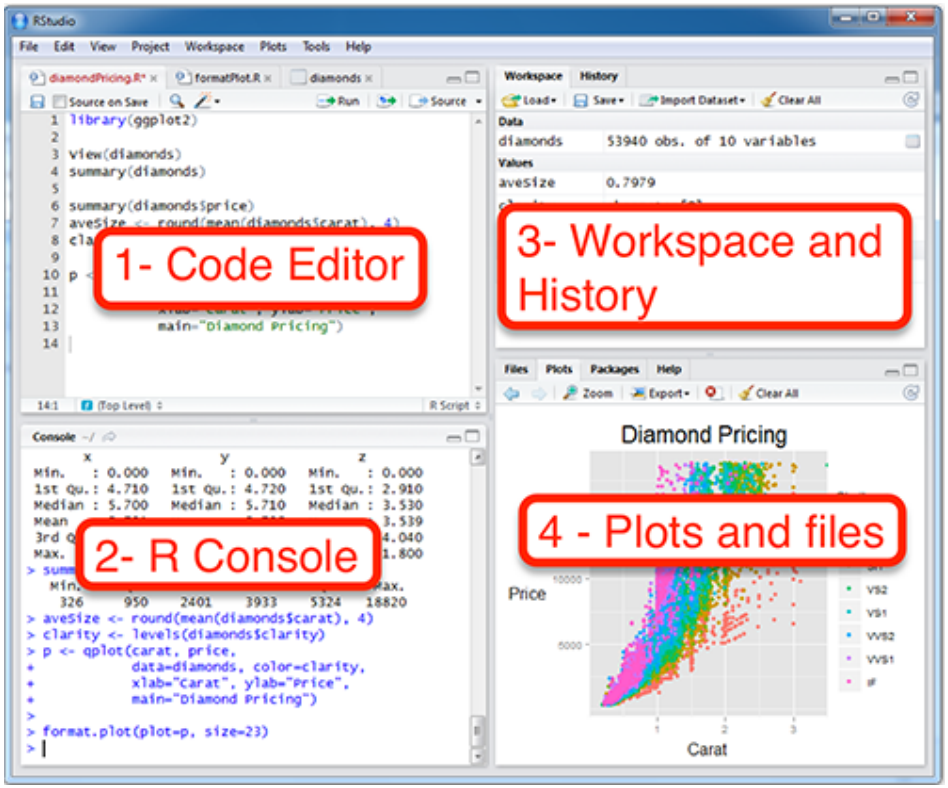
# Typical Rstudio session

- **Console**: output & temporary input - usually unsaved



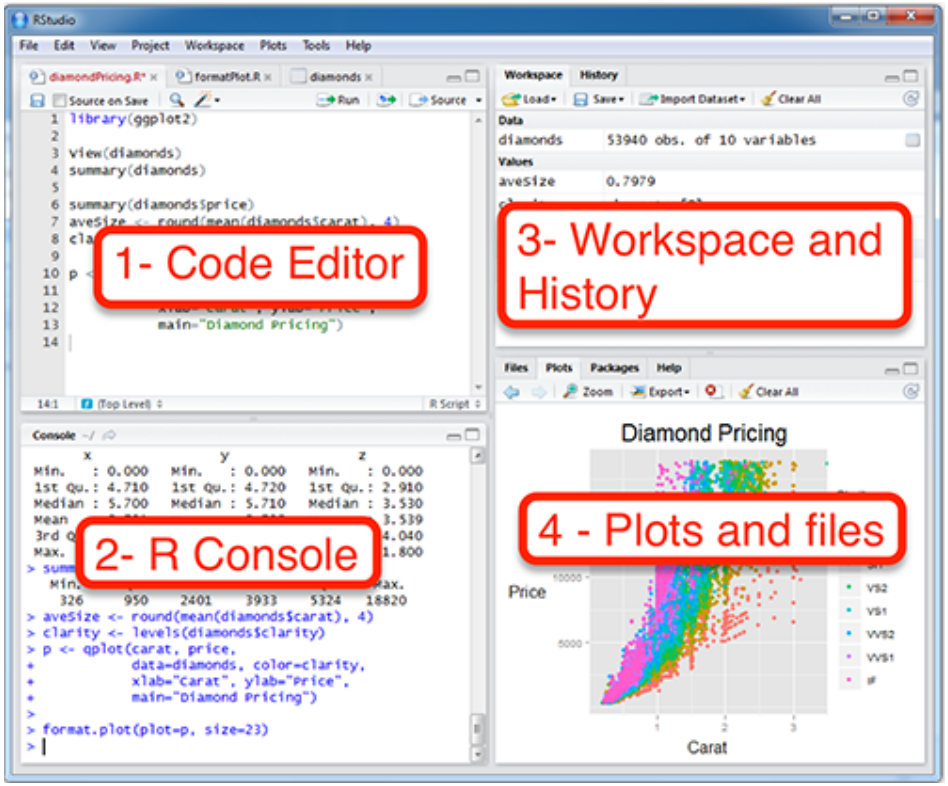
# Typical Rstudio session

- **Workspace**: Stores objects created during the session. Save with save.image().
- **History**: Records commands used. Save with savehistory().



# Typical Rstudio session

- **Plots**: Displays graphs. Save plots with ggsave().
- **Files**: Shows files in the working directory.



# Data structures

# Data structures

- R is an object oriented programming language where an **object** is a generic term to describe something in R.
- There are many types of R-objects.
  Vectors
  Factors
  Lists
  Matrices
  Data Frames

University of Vermont

# Vectors in R

- Vector is a sequence of elements of the same type (numeric, character, logical).

```
a <- c(1,2,3,4,5)
b <- c(6,7,8,9,10)
```

- Command *c* creates a vector that is assigned to object a and b

University of Vermont

# Vectors in R

- Two vectors of the same lenght can be addedd (a+a), multiply (a*a)

```
c <- (a+b)
c
```

```
## [1]  7  9 11 13 15
```

```
d <- (a*b)
d
```

```
## [1]  6 14 24 36 50
```

University of Vermont

# Vectors in R

- Element in a vector can be sorter (*sort()*)

```
e <- c(3,2,20,25,5)
sort(e)
```

```
## [1]  2  3  5 20 25
```

# Factors in R

- Categorical data structure with predefined levels.

```
f <- factor(c("low", 'medium', "high", 'very high'))
```

University of Vermont

# Data

- R has three general classes for data:
  1. **list**: collection of objects of different lengths or classes
  2. **matrix**: vectors of the same length and same class
  3. **data.frame**: vectors of the same length and different classes

University of Vermont

# Lists

- A list is an R-object which can contain many different types of elements inside it like vectors, functions and even another list inside it.

```r
# Creating a list in R with various types of elements
my_list <- list(
  # A numeric vector
  Numbers = c(1, 2, 3, 4, 5),
  # A character vector
  Words = c("apple", "banana", "cherry"))
```

University of Vermont

# Lists

- A list is an R-object which can contain many different types of elements inside it like vectors, functions and even another list inside it.

```
  NestedList = list(
    Logical = c(TRUE, FALSE, TRUE),
    Mixed = c(3.14, "pi", FALSE))
NestedList
```

```
## $Logical
## [1]  TRUE FALSE  TRUE
##
## $Mixed
## [1] "3.14"  "pi"    "FALSE"
```

University of Vermont

# Matrices

- A matrix is a two-dimensional rectangular data set.
- It can be created using a vector input to the matrix function.

```
# Create a 3x3 matrix in R
my_matrix <- matrix(
  data = 1:9,         # Data to fill the matrix
  nrow = 3,           # Number of rows
  ncol = 3,           # Number of columns
  byrow = TRUE)        # Fill the matrix by rows
my_matrix
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

University of Vermont

# Data Frames

- Data frames are tabular data objects.
- **Unlike a matrix in data frame each column can contain different modes of data.**
- The first column can be numeric while the second column can be character and third column - Can be logical.
- It is a list of vectors of equal length.
- Data Frames are created using the ***data.frame()*** function.
- When we execute the above code, it produces the following result:

University of Vermont

# Table

- Summary

| | **Linear** | **Rectangular** |
|---|---|---|
| **All Same Types** | Vectors | Matrix |
| **Mixed** | List | Data Frame |

University of Vermont

# Reading data

# Reading data

- Command for reading in text files is:

```
read.table("suomi.txt", header=T, sep="\t")
```

- This examples has one command with three arguments:
  - file name (in quotes)
  - header that tells whether columns have titles
  - sep that tells that the file is tab-delimited.

University of Vermont

# Reading data

- **CSV File**
- Usage: Reads comma-separated values, often used for structured data.

```
data <- read.csv("file.csv")
```

- **Text File**
- Usage: Reads tabular data from a text file; sep can be adjusted for different delimiters.

```
data <- read.table("file.txt", header = TRUE, sep = "\t")
```

- **Excel File**
- Usage: Reads data from Excel files; you can specify the sheet number or name.

```
library(readxl)
data <- read_excel("file.xlsx", sheet = 1)
```

University of Vermont

# Reading data

- Print the current working directory

```
getwd()
```

- Change to my directory

```
setwd(mydirectory)
setwd("c:/docs/mydir")
```

University of Vermont

# Reading data

- Subscripts are given inside square brackets after the object's name:
- df[,1]
  - Gets the **first column** from the object dat
- df[,1]
  - Gets the **first row** from the object dat
- df[1,1]
  - Gets the **first row** and it's **first column** from the object dat

University of Vermont

# Reading data

- Subscripts can be used for, e.g., extracting a subset of the data:
  - df[which(df$year>1900),]
    - ★ Now, this takes a bit of pondering to work out.
    - ★ First we have the object df, and we are accessing a part of it, because it's name is followed by the square brackets
    - ★ Then we have one command (which) that makes an evaluation whether the column year in the object df has a value higher than 1900.
    - ★ Last the subscript ends with a comma, that tells us that we are accessing rows.
    - ★ So this command takes all the rows that have a year higher 1900 from the object dat that is a data frame.

University of Vermont

# Assigning Values in R

- In R, you can assign values to objects using the syntax object <- value
- An arrow (<-) formed by a smaller than character and a hyphen without a space!
- The equal character (=).

University of Vermont

# R Warning!

- Naming objects:
  - R is a case sensitive language.
  - FOO, Foo, and foo are three different objects
  - Object names can't start with a number
  - Never use special characters, such as å, ä, or ö in object names.

University of Vermont

# Using Functions in R

- R is a function based language where a ***function*** takes in some input and produces some output.
  - Vegas rules: what happens in a function, stays in a function.

University of Vermont

# Data Visualization in R

- Most powerful approach to statistical graphs, based on the **Grammar of Graphics**.
- A graphics language, composed of layers, **geoms** (points, lines, regions), each with graphical **aesthetics** (color, size, shape)

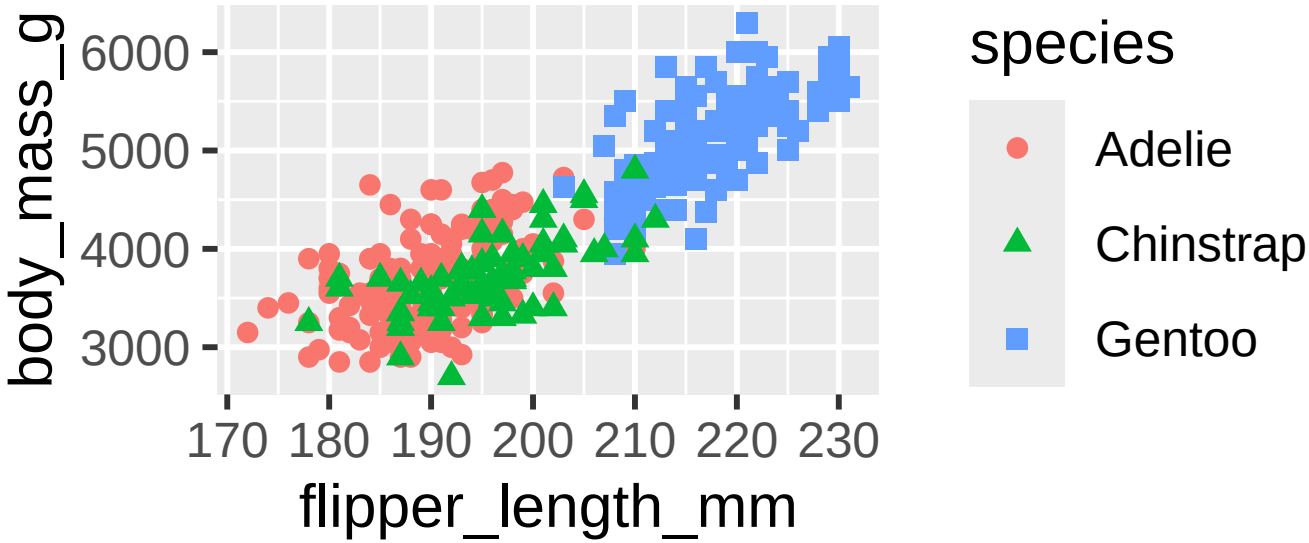
```
library(ggplot2)
library(palmerpenguins)
```

University of Vermont

# An Introduction to ggplot2

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

```
## Warning: Removed 2 rows containing missing values or values
## (`geom_point()`).
```




University of Vermont

# An Introduction to ggplot2

```
ggplot(data = <your data set>, aes(x = <x axis variable>,
                                   y = <y axis variable>)) +
    geom_point(aes(size = <size variable>,
                   color = <color variable>,
                   shape = <shape variable>))
```

University of Vermont

# ## Hypothesis Testing Summary

| Scenario | Null Hypothesis is True | Null Hypothesis is False |
|---|---|---|
| **Reject Null Hypothesis** | **Type I Error**: Incorrectly rejecting the null hypothesis when it is true. | **Correct Decision**: Correctly rejecting the null hypothesis when it is false. |
| **Fail to Reject Null Hypothesis** | **Correct Decision**: Correctly not rejecting the null hypothesis when it is true. | **Type II Error**: Incorrectly failing to reject the null hypothesis when it is false. |

**Explanation**: - **Type I Error**: False positive. We conclude there is an effect or difference when there is none. - **Type II Error**: False negative. We fail to detect an effect or difference when one exists. - **Correct Decisions**: Accurately concluding the presence or absence of an effect or difference based on the truth of the null hypothesis.

University of Vermont