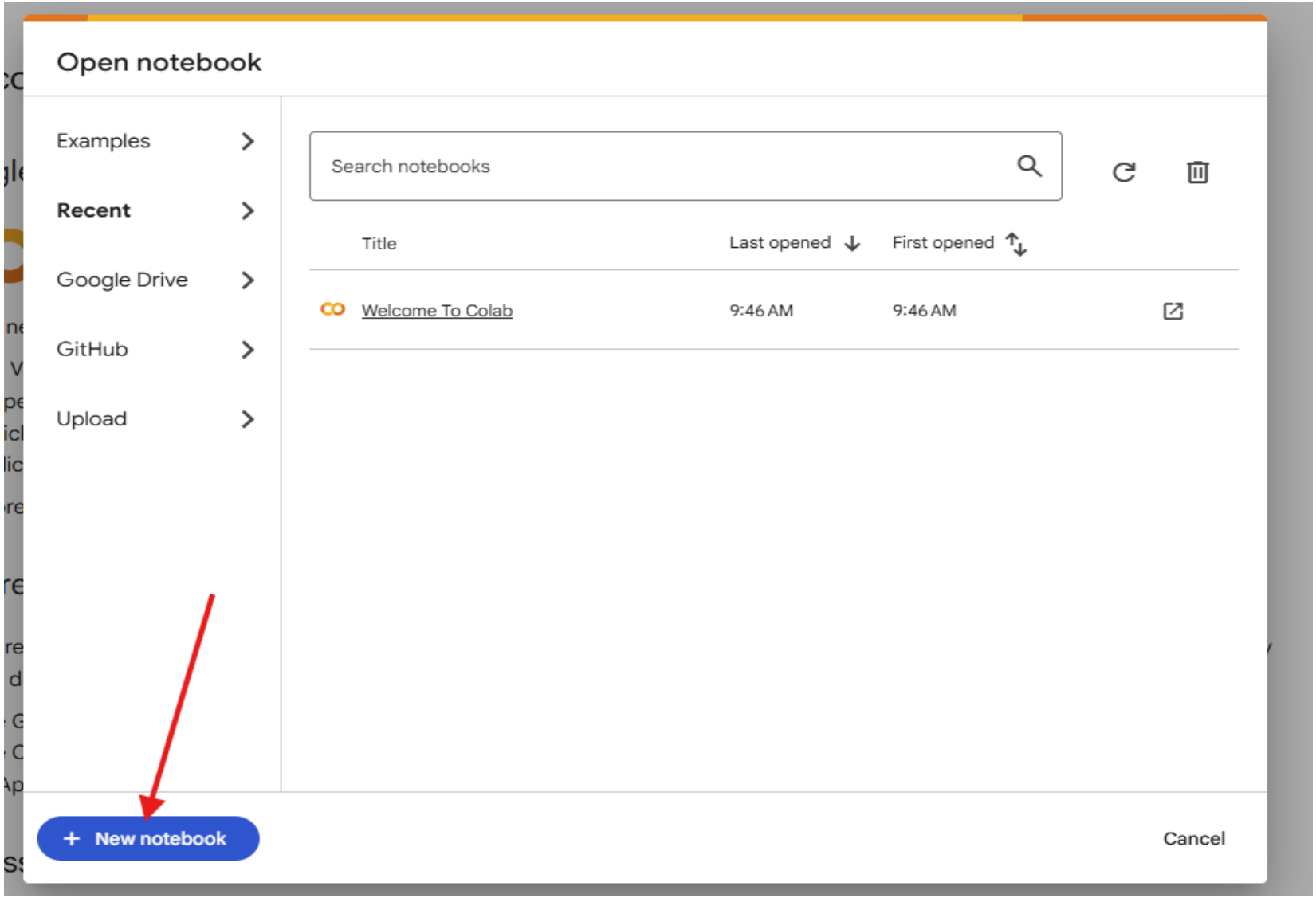
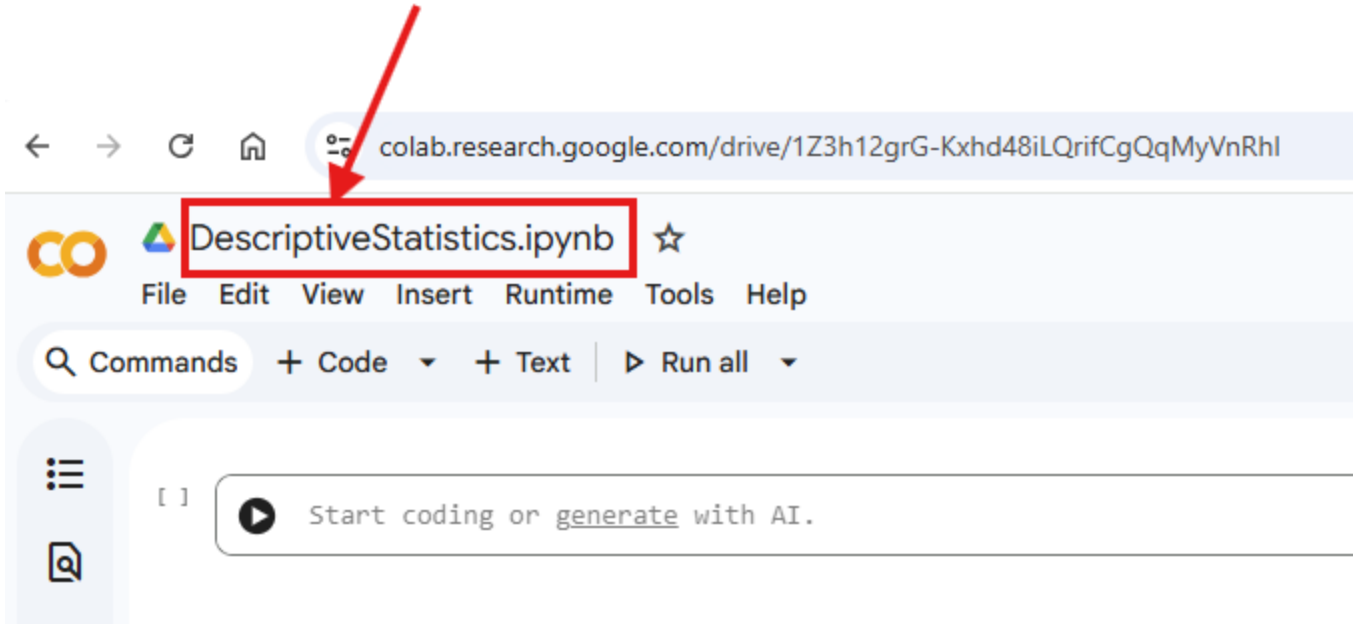
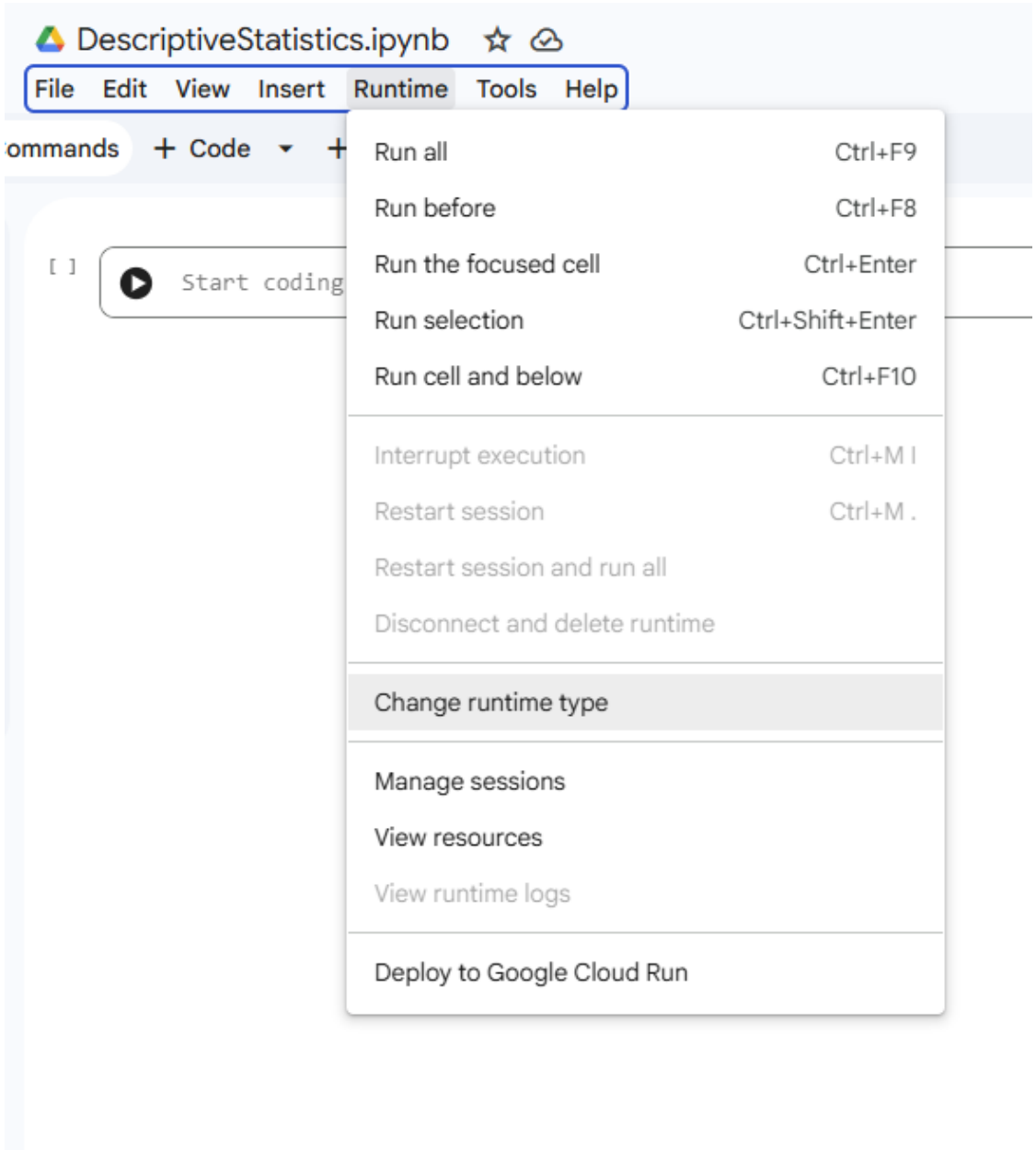
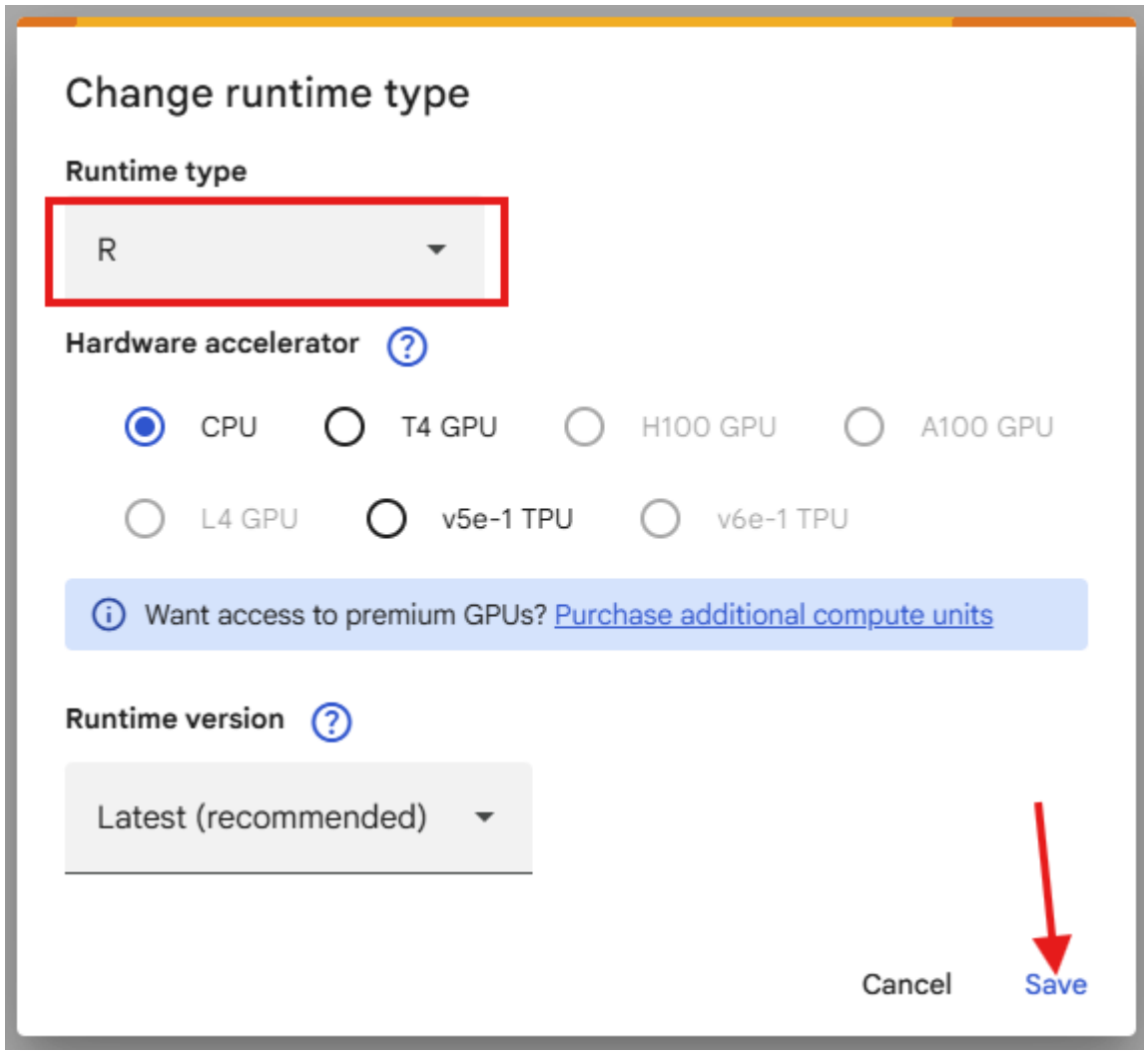
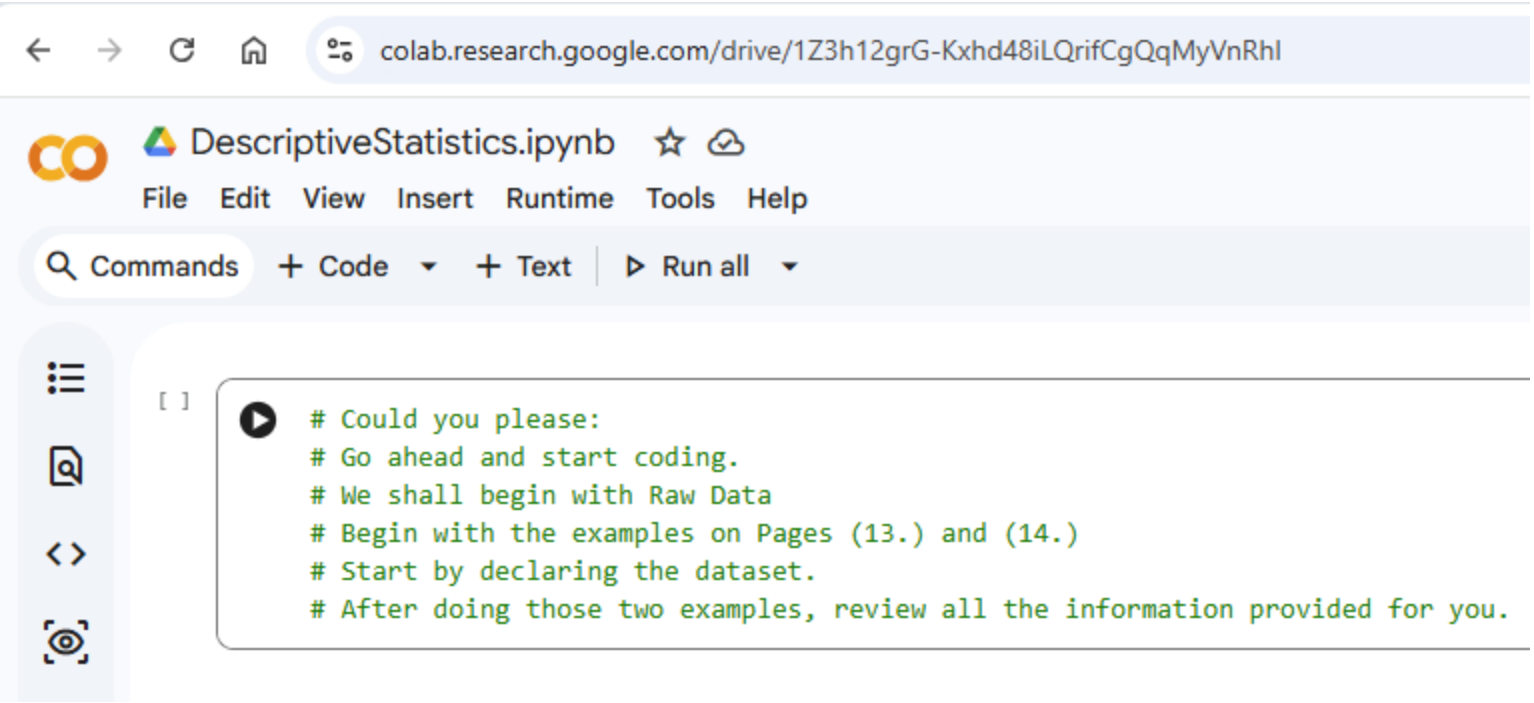
https://colab.research.google.com/




colab.research.google.com/drive/1Z3h12grG-Kxhd48iLQrifCgQqMyVnRhl
DescriptiveStatistics.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
[ ]
Start coding or generate with AI.


DescriptiveStatistics.ipynb
File Edit View Insert Runtime Tools Help
ommands + Code +
[ ]
Start coding
Run all Ctrl+F9
Run before Ctrl+F8
Run the focused cell Ctrl+Enter
Run selection Ctrl+Shift+Enter
Run cell and below Ctrl+F10
Interrupt execution Ctrl+M I
Restart session Ctrl+M .
Restart session and run all
Disconnect and delete runtime
Change runtime type
Manage sessions
View resources
View runtime logs
Deploy to Google Cloud Run


Change runtime type
Runtime type
R
Hardware accelerator
CPU
T4 GPU
H100 GPU
A100 GPU
L4 GPU
v5e-1 TPU
v6e-1 TPU
Want access to premium GPUs? Purchase additional compute units
Runtime version
Latest (recommended)
Cancel
Save



Statistical functions in R are used for statistical computations/analysis of a dataset.

Some statistical functions are already defined in R (these are **built-in functions** also known as **system-defined functions**)

For all other functions not built-in, we have to define them. These are known as **user-defined functions.**

As at today: 07/05/2023;

R programming language has these descriptive statistics represented by these built-in functions.

Let us review them.

For a data frame (dataset in the form of a recursive vector)

Assume the data frame is: `dataset`

Descriptive Statistics	Type	Built-in R Function	Code
Mean	Measure of Center	mean	`mean(dataset)`
Median	Measure of Center	median	`median(dataset)`
Sample Variance	Measure of Spread	var	`var(dataset)`
Sample Standard Deviation	Measure of Spread	sd	`sd(dataset)`
Five-Number Summary	Measure of Location	quantile	`quantile(dataset)`
Minimum	Measure of Location	min	`min(dataset)`
Maximum	Measure of Location	max	`max(dataset)`
Percentile (Example: 70th percentile)	Measure of Location	quantile	`quantile(dataset, c(0.7))`
Percentiles (Example: 34th and 70th percentiles)	Measure of Location	quantile	`quantile(dataset, c(0.34, 0.7))`

This means that:

(1.) We cannot use the built-in function names as names for any user-defined variable or user-defined function.

(2.) We have to define our own functions (user-defined functions) for any function that is not built-in.

User-Defined Functions

The syntax for writing a function is

```
functionName <- function(parameters)
{
    code/expression
}
```

Let us write functions for the rest of the descriptive statistics measures.

Measure of Center: **Mode**

```
# Function to calculate the mode of a dataset
# It computes the mode for unimodal and multimodal data
Mode <- function(x)
{
    resultMode <- names(table(x))[table(x) == max(table(x))]
    cat(paste(resultMode, collapse = ", "))
}

# Call the function
Mode(dataset)
```

Measure of Center: **Midrange**

```
# Function to calculate the midrange of a dataset
Midrange <- function(x)
{
    (min(x) + max(x)) / 2
}

# Call the function
Midrange(dataset)
```

Measure of Spread: **Range**

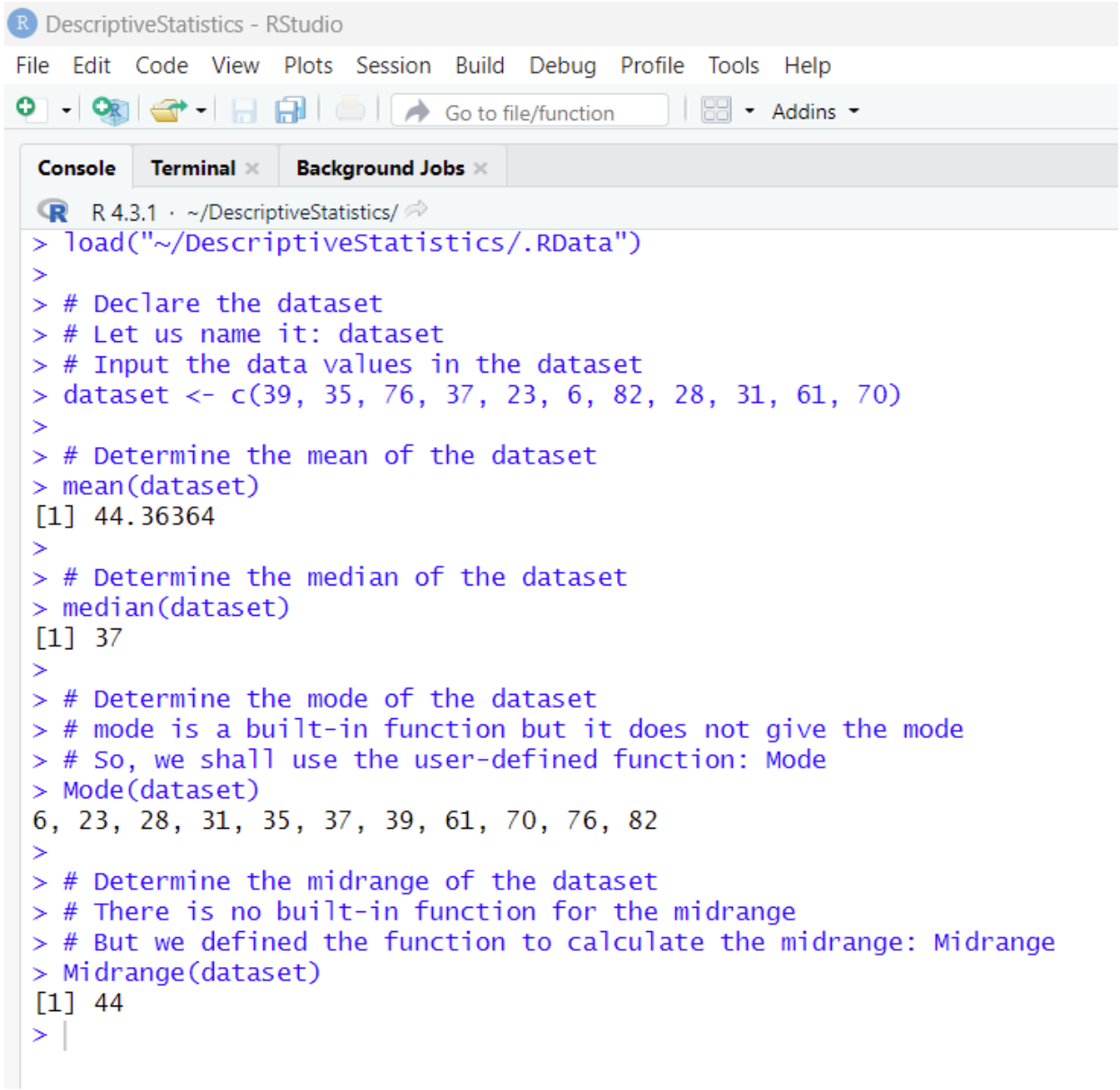
```
# Function to calculate the range of a dataset
Range <- function(x)
{
    max(x) - min(x)
}

# Call the function
Range(dataset)
```

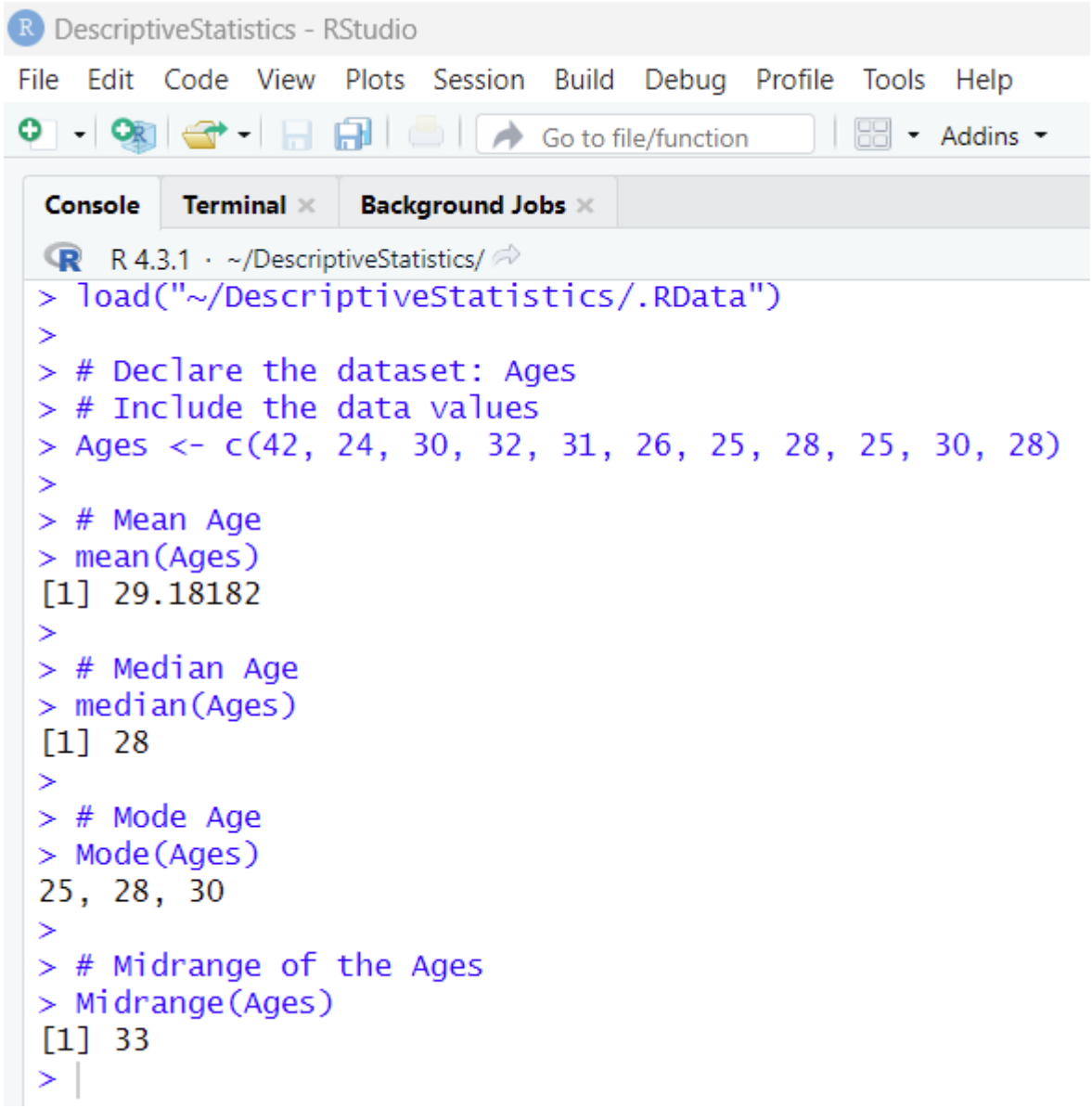
Measure of Spread: **Population Variance**

```
# Function to calculate the population variance of a dataset
PopulationVariance <- function(x)
{
    var(x) * (length(x) - 1) / length(x)
}

# Call the function
PopulationVariance(dataset)
```

Measure of Spread: **Population Standard Deviation**

```
# Function to calculate the population standard deviation of a dataset
PopulationStandardDeviation <- function(x)
{
    sd(x) * sqrt((length(x) - 1)/length(x))
}

# Call the function
PopulationStandardDeviation(dataset)
```

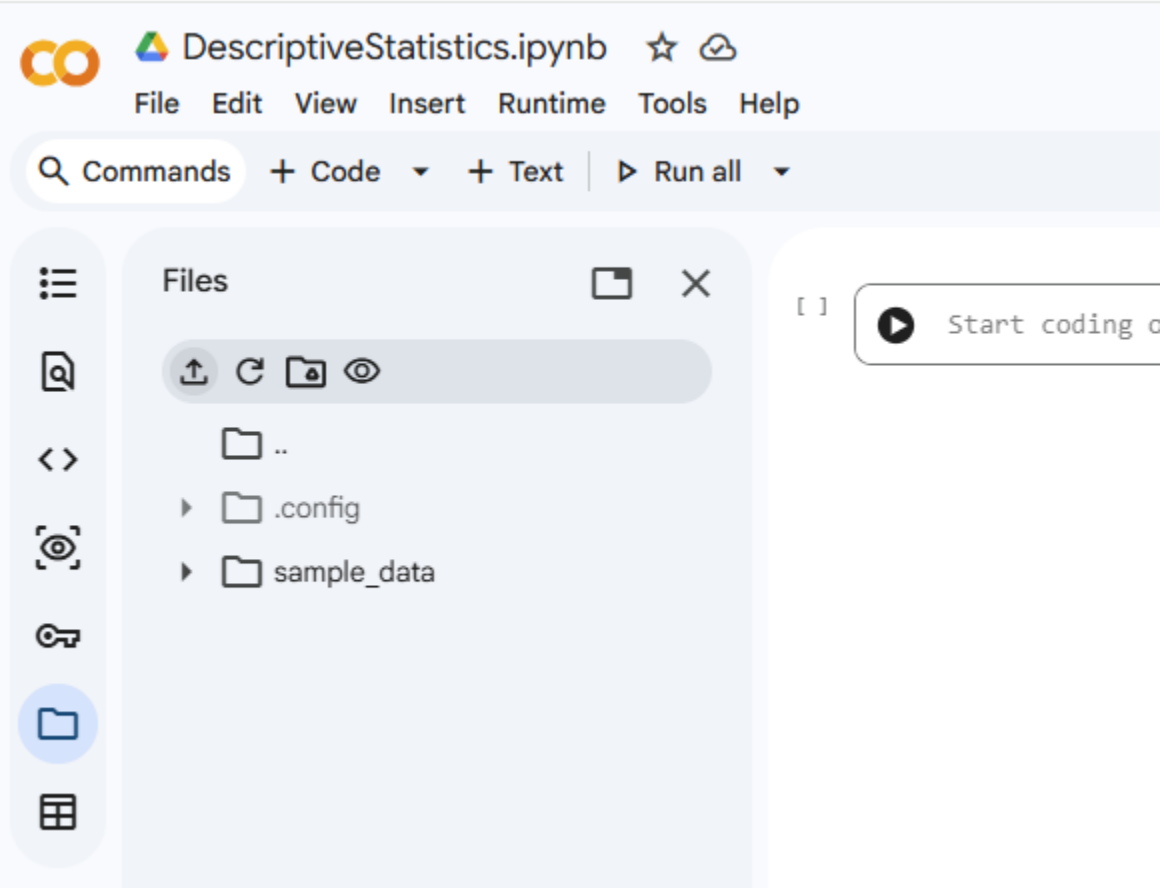

DescriptiveStatistics - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Console
Terminal
Background Jobs
R 4.3.1 · ~/DescriptiveStatistics/
> load("~/DescriptiveStatistics/.RData")
>
> # Declare the dataset
> # Let us name it: dataset
> # Input the data values in the dataset
> dataset <- c(39, 35, 76, 37, 23, 6, 82, 28, 31, 61, 70)
>
> # Determine the mean of the dataset
> mean(dataset)
[1] 44.36364
>
> # Determine the median of the dataset
> median(dataset)
[1] 37
>
> # Determine the mode of the dataset
> # mode is a built-in function but it does not give the mode
> # So, we shall use the user-defined function: Mode
> Mode(dataset)
6, 23, 28, 31, 35, 37, 39, 61, 70, 76, 82
>
> # Determine the midrange of the dataset
> # There is no built-in function for the midrange
> # But we defined the function to calculate the midrange: Midrange
> Midrange(dataset)
[1] 44
>

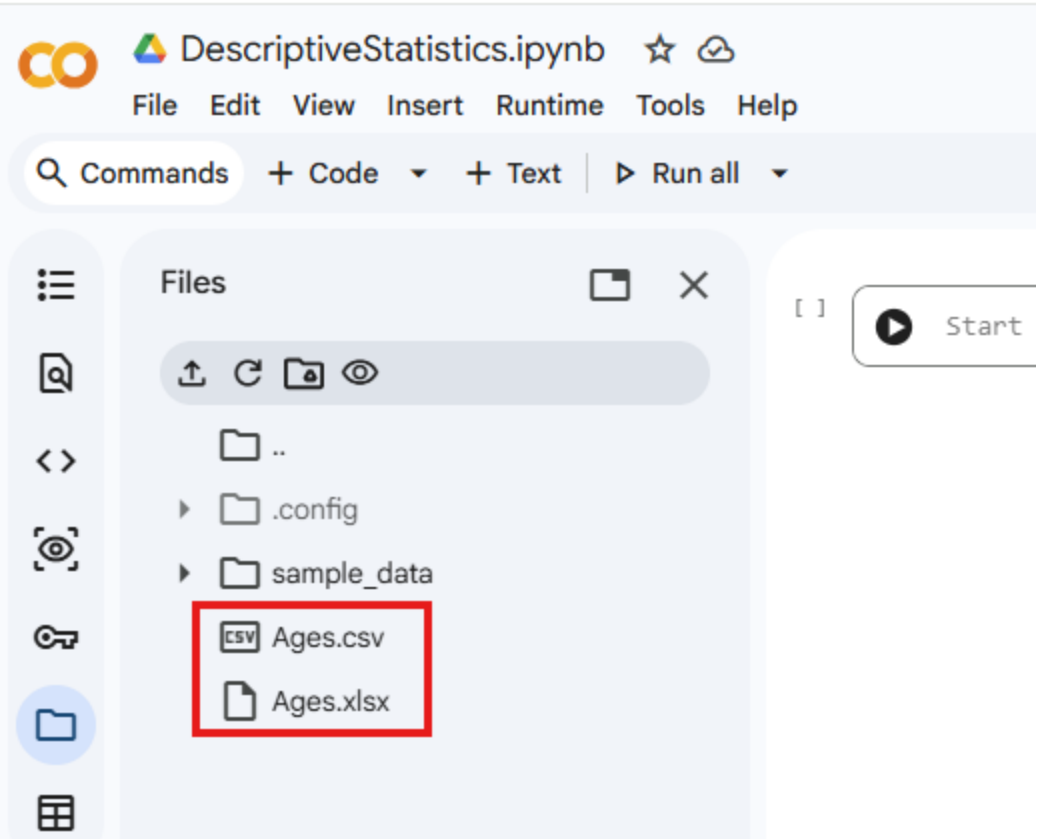

DescriptiveStatistics - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Console
Terminal
Background Jobs
R 4.3.1 · ~/DescriptiveStatistics/
> load("~/DescriptiveStatistics/.RData")
>
> # Declare the dataset: Ages
> # Include the data values
> Ages <- c(42, 24, 30, 32, 31, 26, 25, 28, 25, 30, 28)
>
> # Mean Age
> mean(Ages)
[1] 29.18182
>
> # Median Age
> median(Ages)
[1] 28
>
> # Mode Age
> Mode(Ages)
25, 28, 30
>
> # Midrange of the Ages
> Midrange(Ages)
[1] 33
>

What about Ungrouped data and Grouped Data: where the file is an Excel file or Comma-Separated Values file?

Ages.xlsx

Ages.csv




DescriptiveStatistics.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Files
..
.config
sample_data
Ages.csv
Ages.xlsx
[ ]
Start

```r
# If the uploaded file is an Excel file
library(readxl)

# (1.) Read the Excel file: Ages.xlsx
# .name_repair = "minimal" stops the "New names" message
datasetAges <- read_excel("Ages.xlsx", col_names = FALSE, .name_repair = "minimal")

# (2.) The Cleaning Process
# Step A: unlist() turns the grid into a single list
# Step B: as.numeric() turns them into numbers
# Step C: na.omit() deletes all the empty/junk cells
cleanData <- na.omit(suppressWarnings(as.numeric(unlist(datasetAges))))

# (3.) Calculate the Mean
resultMean <- mean(cleanData)

# 4. Final Display: Include the sample size and the mean.
cat("The sample size of the Ages dataset is:", length(cleanData), "\n")
cat("The mean age (without rounding) is:", resultMean, "years. \n")
cat("The mean age (rounded to 3 decimal places) is:", round(resultMean, 3), "years. \n")
```

```
The sample size of the Ages dataset is: 11
The mean age (without rounding) is: 29.18182 years.
The mean age (rounded to 3 decimal places) is: 29.182 years.
```

```r
# If the uploaded file is a CSV file

# (1.) Read the CSV file: Ages.csv
# header = FALSE because your file is just a grid of numbers
datasetAgesPlayers <- read.csv("Ages.csv", header = FALSE)

# (2.) The Cleaning Process
# Step A: unlist() flattens the columns into one long list
# Step B: suppressWarnings() silences the "NAs introduced by coercion" message
# Step C: na.omit() removes the empty/junk cells entirely
cleanDataset <- na.omit(suppressWarnings(as.numeric(unlist(datasetAgesPlayers))))

# (3.) Calculate the Mean
resultAverage <- mean(cleanDataset)

# 4. Final Display: Include the sample size and the mean.
cat("The sample size of the Ages dataset is:", length(cleanDataset), "\n")
cat("The mean age (without rounding) is:", resultAverage, "years. \n")
cat("The mean age (rounded to 3 decimal places) is:", round(resultAverage, 2), "years. \n")
```

```
The sample size of the Ages dataset is: 11
The mean age (without rounding) is: 29.18182 years.
The mean age (rounded to 3 decimal places) is: 29.18 years.
```