lab0

Parker DeBruyne - V00837207

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
1 + 1
```

```
[1] 2
```

You can add options to executable code like this

```
[1] 4
```

The `echo: false` option disables the printing of code (only output is displayed).

```
# install.packages("rmarkdown")
library(rmarkdown)

# Welcome to R! This script introduces basic commands and concepts.
# Any text after "#" is a comment and will not be executed.

# R as a Calculator
# You can use R like a calculator for basic arithmetic and mathematical operations.

# Basic Arithmetic
3 + 5   # Addition
```

```
[1] 8
```

```
10 - 4  # Subtraction
```

```
[1] 6
```

```
6 * 7   # Multiplication
```

```
[1] 42
```

```
9 / 3   # Division
```

```
[1] 3
```

```
2^3     # Exponentiation
```

```
[1] 8
```

```
10 %% 3 # Modulus (remainder of 10 divided by 3)
```

```
[1] 1
```

```
10 %/% 3 # Integer division (quotient)
```

```
[1] 3
```

```
# Order of Operations (PEMDAS)
# R follows the precedence rules: Parentheses, Exponents, Multiplication/Division, Addition/S
2 + 3 * 5  # Multiplication is done before addition
```

```
[1] 17
```

```
(2 + 3) * 5  # Parentheses change the order
```

```
[1] 25
```

```r
# Assigning Values to Variables
# The assignment operator "<-" is used to assign values to variables.
x <- 5    # Assign 5 to x
y <- 10   # Assign 10 to y
x + y     # Add x and y
```

```
[1] 15
```

```r
z <- x * y  # Assign the product of x and y to z

print(z)
```

```
[1] 50
```

```r
# Data Types in R
# R has several fundamental data types, such as:
# - Numeric (e.g., 3.14, 42)
# - Integer (e.g., 5L, where "L" indicates an integer)
# - Character (e.g., "Hello")
# - Logical (e.g., TRUE, FALSE)

class(x)       # Check the data type of x
```

```
[1] "numeric"
```

```r
x <- "Hello"   # Change x to a character
class(x)       # Check the new data type of x
```

```
[1] "character"
```

```r
# Functions and Built-in Math
# Functions are pre-defined commands in R that perform specific tasks.
sqrt(16)      # Square root
```

```
[1] 4
```

```r
log(100, 10)  # Logarithm base 10
```

```
[1] 2
```

```r
exp(2)        # Exponential function
```

```
[1] 7.389056
```

```r
abs(-15)      # Absolute value
```

```
[1] 15
```

```r
round(3.14159, 2)  # Round to 2 decimal places
```

```
[1] 3.14
```

```r
ceiling(3.1)      # Round up
```

```
[1] 4
```

```r
floor(3.9)        # Round down
```

```
[1] 3
```

```r
# Logical Operators
# Logical operators allow comparisons and return TRUE or FALSE.
a <- 7
b <- 10
a > b    # Is a greater than b?
```

```
[1] FALSE
```

```r
a == b   # Is a equal to b?
```

```
[1] FALSE
```

```r
a != b   # Is a not equal to b?
```

```
[1] TRUE
```

```r
a < b && a > 5  # Logical AND
```

```
[1] TRUE
```

```r
a < b || a < 5  # Logical OR
```

```
[1] TRUE
```

```r
# Working with Vectors
# A vector is a sequence of data elements of the same type.
numbers <- c(5, 4, 8, 9, 1)  # Create a numeric vector
letters <- c("a", "b", "c", "abc")  # Create a character vector
logical_vec <- c(TRUE, FALSE, TRUE)  # Create a logical vector

# Perform operations on vectors
numbers + 2   # Add 2 to each element
```

```
[1]  7  6 10 11  3
```

```r
numbers * 3   # Multiply each element by 3
```

```
[1] 15 12 24 27  3
```

```r
sum(numbers)  # Sum of elements
```

```
[1] 27
```

```r
mean(numbers) # Average of elements
```

```
[1] 5.4
```

```r
length(numbers) # Length of the vector
```

```
[1] 5
```

```r
# Indexing and Slicing
# Access elements of a vector using square brackets.
numbers[1]    # First element
```

```
[1] 5
```

```r
numbers[2:4]  # Second to fourth elements
```

```
[1] 4 8 9
```

```r
numbers[-1]   # All elements except the first
```

```
[1] 4 8 9 1
```

```r
numbers[numbers > 2]  # Elements greater than 2
```

```
[1] 5 4 8 9
```

```r
head(numbers, 3) # Provides first 3 elements
```

```
[1] 5 4 8
```

```r
tail(numbers, 3) # Provides last 3 elements
```

```
[1] 8 9 1
```

```r
# Sequence and Repetition
1:10              # Sequence of numbers from 1 to 10
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

```r
seq(1, 10, 2)     # Sequence from 1 to 10, increment by 2
```

```
[1] 1 3 5 7 9
```

```r
rep(3, 5)         # Repeat 3 five times
```

```
[1] 3 3 3 3 3
```

```r
rep(c(1, 2), each = 3) # Repeat each element 3 times
```

```
[1] 1 1 1 2 2 2
```

```r
rep(c(1, 2), 3)   # Repeat the set (1,2) 3 times
```

```
[1] 1 2 1 2 1 2
```

```r
# Creating and Accessing Lists
# A list is a collection of elements that can have different types.
my_list <- list(name = "Andrew", age = 25, scores = c(85, 90, 88))
my_list
```

```
$name
[1] "Andrew"

$age
[1] 25

$scores
[1] 85 90 88
```

```r
my_list$name      # Access the "name" element
```

```
[1] "Andrew"
```

```r
my_list$scores[2] # Access the second score
```

```
[1] 90
```

```r
# Data Frames
# A data frame is a table-like structure where each column can have a different type.
names <- c("Student1", "Student2", "Student3")
scores <- c(85, 90, 88)
data <- data.frame(names, scores)  # Create a data frame
print(data)# Display the data frame
```

```
     names scores
1 Student1     85
2 Student2     90
3 Student3     88
```

```r
View(data)
data$scores          # Access the "scores" column
```

```
[1] 85 90 88
```

```r
data[2, "scores"]    # Access Student2's score
```

```
[1] 90
```

```r
data[1:2, ]          # Access the first two rows
```

```
     names scores
1 Student1     85
2 Student2     90
```

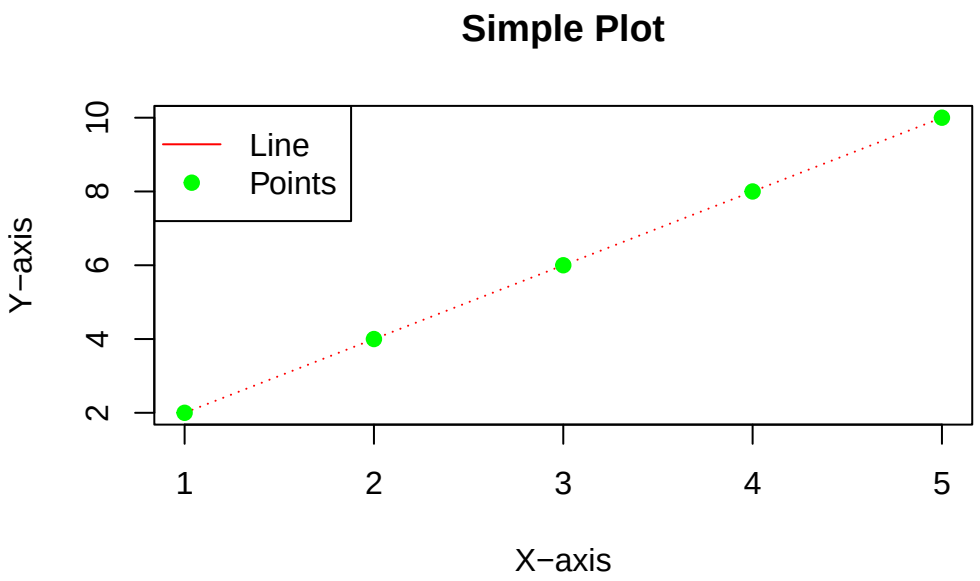
```r
data[, "names"]      # Access the "names" column
```

```
[1] "Student1" "Student2" "Student3"
```

```r
# Plotting Basics
# R has built-in functions for creating plots.
x <- c(1, 2, 3, 4, 5)
y <- c(2, 4, 6, 8, 10)
plot(x, y, main = "Simple Plot", xlab = "X-axis", ylab = "Y-axis", col = "blue", pch = 16)

# Advanced Plotting
```

```r
# Adding lines, points, and legends to plots.
lines(x, y, col = "red", lty = 3)   # Add a red line
points(x, y, col = "green", pch = 19) # Add green points
legend("topleft", legend = c("Line", "Points"), col = c("red", "green"), pch = c(NA, 19), lty
```




```r
# Install and Load Packages
# Packages extend R's functionality. Install a package using install.packages().
# install.packages("ggplot2")  # Install ggplot2 for advanced plotting
library(ggplot2)             # Load ggplot2 for use

# Getting Help
# Use ? or help() to access documentation.
?mean          # Documentation for "mean"
help.start()   # Launch the R help system
```

```
starting httpd help server ... done

If the browser launched by '/usr/bin/open' is already running, it is
    *not* restarted, and you must switch to its window.
Otherwise, be patient ...
```

```r
#-----------------Quick Practice----------------------#

# Install Packages
```

```r
# install.packages("readr") # Example ("package name")

# Call a package for use
library(readr)

# Before running these set of codes, guess which type of inputs?
guess_parser("2025-01-08")
```

```
[1] "date"
```

```r
guess_parser("16:10")
```

```
[1] "time"
```

```r
guess_parser(c("TRUE", "FALSE"))
```

```
[1] "logical"
```

```r
guess_parser(c("a", "b", "c"))
```

```
[1] "character"
```

```r
guess_parser(c("12,352,561"))
```

```
[1] "number"
```

```r
# install.packages("tidyverse")
# install.packages("palmerpenguins")
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v stringr   1.5.1
v forcats   1.0.0     v tibble    3.2.1
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```r
library(palmerpenguins)
```

$$\beta_1 = \frac{\Sigma_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\Sigma_{i=1}^{n}(x_i - \bar{x})^2}$$