**R Programming**

**Introduction to R**

R is a programming language and environment primarily used for statistical computing, data analysis, and data visualization. It is open-source and widely used among statisticians, data scientists, and researchers for performing data manipulation, statistical modeling, and graphical representations.

**Installing R and RStudio**

1. **Install R**:
   * Visit the official CRAN website: <https://cran.r-project.org/>
   * Download and install the version appropriate for your operating system (Windows, macOS, or Linux).
2. **Install RStudio (Optional but Recommended)**:
   * RStudio is an integrated development environment (IDE) for R.
   * Visit: <https://posit.co/products/rstudio/>
   * Download and install it.

Once R and RStudio are installed, open RStudio to start writing and running your R code.

**Basic R Syntax**

**Variables and Data Types**

In R, you can assign values to variables using the <- operator (although = is also valid).

x <- 10

y <- 5.5

z <- "Hello, World!"

R supports several data types:

* **Numeric**: Numbers (e.g., 10, 2.5)
* **Character**: Text strings (e.g., "Hello")
* **Logical**: TRUE or FALSE
* **Complex**: Complex numbers (e.g., 2 + 3i)

**Data Structures**

R provides several built-in data structures to organize and manipulate data.

1. **Vectors**: Ordered collections of elements of the same type.

v <- c(1, 2, 3, 4, 5) # Numeric vector

names(v) <- c("A", "B", "C", "D", "E") # Assign names to vector elements

1. **Lists**: Can hold elements of different types.

my\_list <- list(name="John", age=30, scores=c(95, 88, 100))

1. **Data Frames**: Tables or spreadsheets with rows and columns, used for structured data.

df <- data.frame(

name = c("Alice", "Bob", "Charlie"),

age = c(25, 30, 35),

height = c(5.5, 5.7, 6.0)

)

**Functions**

R has many built-in functions. Here’s how to define a simple function:

add\_numbers <- function(a, b) {

return(a + b)

}

result <- add\_numbers(10, 20)

print(result) # Output: 30

**Control Structures**

R supports conditional statements (if, else) and loops (for, while).

* **if-else Statement**:

x <- 10

if (x > 5) {

print("x is greater than 5")

} else {

print("x is less than or equal to 5")

}

* **For Loop**:

for (i in 1:5) {

print(i)

}

* **While Loop**:

i <- 1

while (i <= 5) {

print(i)

i <- i + 1

}

**Data Manipulation with dplyr**

One of the most powerful R packages for data manipulation is dplyr. Here's a quick example using a data frame:

1. Install dplyr (if you don't have it already):

install.packages("dplyr")

1. Load the package:

library(dplyr)

1. Basic data manipulation using dplyr:

df <- data.frame(

name = c("Alice", "Bob", "Charlie", "David"),

age = c(25, 30, 35, 40),

height = c(5.5, 5.7, 6.0, 5.9)

)

# Filter rows where age is greater than 30

df\_filtered <- df %>% filter(age > 30)

# Select only 'name' and 'age' columns

df\_selected <- df %>% select(name, age)

# Add a new column 'age\_group'

df <- df %>% mutate(age\_group = ifelse(age < 30, "Young", "Old"))

**Data Visualization with ggplot2**

ggplot2 is a widely used package for creating high-quality graphics in R.

1. Install ggplot2:

install.packages("ggplot2")

1. Load the package:

library(ggplot2)

1. Basic plot example:

ggplot(df, aes(x=age, y=height)) +

geom\_point() +

labs(title="Age vs Height", x="Age", y="Height")

This will generate a scatter plot with age on the x-axis and height on the y-axis.

**Conclusion**

This document provides a quick introduction to R programming and covers the basics of data manipulation, function definition, and data visualization.