GUIDE TO R PROGRAMMING: FROM BEGINNER TO ADVANCED

This guide provides an in-depth exploration of R programming, with detailed explanations, statistical concepts, and extensive code examples. By the end, you will have a strong foundation in both programming and applying statistical techniques in R.

---

1. Getting Started with R

1.1 Installing R and RStudio

- Download and install R from CRAN.
- Download RStudio, an Integrated Development Environment (IDE) for R, from RStudio.

1.2 R Basics

R can perform arithmetic, assign variables, and much more.

Arithmetic Operations:

```
5 + 3  # Addition
10 / 2 # Division
2^3    # Exponentiation
sqrt(16) # Square root
log(100, base = 10) # Logarithm
```

Variables and Assignments:

```
x <- 10  # Assign 10 to x
y <- 20
z <- x + y  # z now contains 30
print(z)
```

Comments: Use # to add comments in your code.

---

2. Data Types and Structures

2.1 Data Types in R

1. Numeric: Continuous variables (e.g., height, weight).
2. `x <- 45.7  # Numeric value`
3. Integer: Whole numbers.
4. `y <- as.integer(7)`
5. Character: Textual data.

6. `name <- "R Programming"`
7. Logical: `TRUE` or `FALSE`.
8. `is_valid <- TRUE`

2.2 Data Structures

Vectors (1D Homogeneous Data):

```
vec <- c(1, 2, 3, 4)
vec * 2  # Multiply all elements by 2
mean(vec)  # Calculate mean
```

Matrices (2D Homogeneous Data):

```
mat <- matrix(1:9, nrow = 3)
dim(mat)  # Check dimensions
```

Data Frames (2D Heterogeneous Data):

Data frames allow you to mix different data types (numeric, character, etc.) in a tabular format.

```
df <- data.frame(
  Name = c("Alice", "Bob"),
  Age = c(25, 30),
  Score = c(90, 85)
)
head(df)  # View first rows
```

---

3. Data Manipulation

R provides powerful tools for manipulating datasets.

3.1 Subsetting Data

```
vec <- c(10, 20, 30)
vec[1]  # Access first element
vec[c(1, 3)]  # Access multiple elements
```

3.2 Using `dplyr`

`dplyr` simplifies data manipulation. Install it first:

```
install.packages("dplyr")
library(dplyr)
```

Examples:

- Select columns:
- `df %>% select(Name, Score)`
- Filter rows:
- `df %>% filter(Score > 85)`

- Create new variables:
- `df %>% mutate(Grade = ifelse(Score > 88, "A", "B"))`

---

4. Data Visualization

4.1 Base R Plotting

R’s base plotting functions are simple yet effective:

```
plot(cars$speed, cars$dist,
     main = "Speed vs Distance",
     xlab = "Speed", ylab = "Distance")
```

4.2 Using `ggplot2`

Install and load the package:

```
install.packages("ggplot2")
library(ggplot2)
```

Create an elegant scatterplot:

```
ggplot(data = mtcars, aes(x = wt, y = mpg)) +
  geom_point(color = "blue") +
  ggtitle("Car Weight vs MPG") +
  xlab("Weight") + ylab("Miles per Gallon")
```

---

5. Statistical Analysis

5.1 Descriptive Statistics

Summarize data using measures of central tendency and variability.

```
data <- c(4, 5, 8, 9, 10)
mean(data)  # Calculate mean
median(data)  # Calculate median
sd(data)  # Calculate standard deviation
```

5.2 Hypothesis Testing

t-Test Example:

```
group1 <- c(5, 7, 8, 6)
group2 <- c(8, 9, 10, 7)
t.test(group1, group2, alternative = "two.sided")
```

Chi-Square Test:

```
observed <- c(30, 50, 20)
expected <- c(33, 47, 20)
chisq.test(observed, p = expected / sum(expected))
```

---

6. Machine Learning in R

6.1 Linear Regression

Build and evaluate a linear regression model:

```
model <- lm(mpg ~ wt + hp, data = mtcars)
summary(model)
```

6.2 Classification with `caret`

Install `caret` for advanced modeling:

```
install.packages("caret")
library(caret)
```

Train a model:

```
set.seed(123)
train_data <- mtcars
model <- train(mpg ~ ., data = train_data, method = "lm")
```

---

7. Advanced Programming Concepts

7.1 Functions

Write reusable code blocks:

```
square <- function(x) {
  return(x^2)
}
square(4)  # Returns 16
```

7.2 Loops and Apply Functions

Using `lapply`:

```
nums <- list(1, 2, 3)
lapply(nums, function(x) x^2)  # Square each element
```

Best Practices

1. Keep your code modular and commented.
2. Use version control systems like Git.
3. Test your scripts on subsets of data.
4. Use R Markdown to create reproducible reports.

This guide will help you develop a strong foundation in R, enabling you to apply statistical concepts and build complex solutions. Would you like further elaboration on any specific section?