📘 Chapter Notes: *Getting Started in R*

🧮 1. Using R as a Calculator

R can perform **basic arithmetic** just like a calculator:

r
CopyEdit
```
2 + 2         # [1] 4
2 * 3         # [1] 6
2 / 2         # [1] 1
2 ^ 2         # [1] 4
```

-

You can **use semicolons** to run multiple operations in one line:

r
CopyEdit
```
2 * 3; 2 / 2; 2 ^ 2
```

-

📦 2. Storing Values with Variables

Assign values using `=` or `<-`:

r
CopyEdit
```
x = 2
x * 3; x / x; x ^ x
# Output: 6, 1, 4
```

-

🔧 3. Using Functions

R uses **functions** like:

r

CopyEdit

```
sqrt(1)      # Square root → [1] 1
exp(2)       # Exponential (e^2) → [1] 7.389056
```

-
- Functions take **arguments inside parentheses**.

Use `?function_name` for help:

r
CopyEdit

```
?sqrt
```

-

📋 4. Vectors in R

- A **vector** is a 1D list of numbers.

Create with the `c()` function (combine/concatenate):

r
CopyEdit

```
x = c(3, 1, 9)
```

-

Perform operations on vectors:

r
CopyEdit

```
mean(x)    # [1] 4.33
sum(x)     # [1] 13
```

-

➕ 5. Vector Arithmetic

R supports **element-wise vector operations**:

r
CopyEdit
```r
x = c(3, 1, 9)
y = c(2, 5, 6)
x + y       # [1] 5 6 15
```

-

🔍 6. Indexing Vectors

Use square brackets `[ ]` to access elements:

r
CopyEdit
```r
x[2]        # [1] 1
```

-

💬 7. Comments in R

Use `#` to write comments in code (ignored by R):

r
CopyEdit
```r
# This is a comment
```

-

🤖 8. Logic in R

- R includes:
    - `if` / `else` statements
    - Logical operators: `&` (and), `|` (or)

- These tools enable **conditional code execution**.

💡 9. RStudio Recommendation

- **RStudio** is suggested for beginners:
    - Combines editor, console, environment, and plots
    - Free to download and user-friendly

✅ Key Takeaways

- R is intuitive for math, data, and logic
- Vector operations are fast and natural
- Learn to explore R's built-in functions
- RStudio is the best interface to start with

📘 Chapter Notes: Functions in R

🧠 What is a Function?

- A **function** is a reusable block of code that takes **inputs (arguments)**, performs operations, and **returns an output**.

Syntax:

r
CopyEdit

```
function_name = function(arg1, arg2, ...) {
  # code block
  return(result)
}
```

-

✅ Example 1: Basic Function

r
CopyEdit

```
my.first.function = function(x){
  return(x^2)
}
my.first.function(2)   # Output: 4
```

- `my.first.function` is the name of the function.
- Takes `x` as input and returns `x^2`.

✅ Example 2: Pythagorean Theorem

r
CopyEdit

```
pyth = function(a, b){
  c = sqrt(a^2 + b^2)
  return(c)
}
pyth(3, 4)   # Output: 5
```

- Computes hypotenuse `c` given `a` and `b`.

✅ Example 3: Even or Odd Checker

r
CopyEdit

```
is.even = function(x){
  if(x %% 2 == 0){
    return(TRUE)
  }
  if(x %% 2 != 0){
    return(FALSE)
  }
```

```
}
is.even(31); is.even(44)   # Output: FALSE, TRUE
```

- Uses `%%` (modulo operator) to check divisibility.
- `%/%` gives integer division; `%%` gives remainder.

🗝️ Key Takeaways:

- Use functions to simplify and organize code.
- You can pass multiple arguments.
- Always use `return()` to output the result.
- Use `#` for comments to explain your code.

🔁 Chapter Notes: Looping in R

🔁 For Loop

- A **for loop** repeats code for a fixed number of iterations.

Syntax:

r
CopyEdit

```
for(i in 1:n){
  # code using i
}
```

-

✅ Example 1: Fill a Vector

r
CopyEdit

```r
x = rep(NA, 10)
for(i in 1:10){
  x[i] = i
}
x   # Output: 1 2 3 4 5 6 7 8 9 10
```

- Pre-fills vector `x` with `NA`, then sets `x[i] = i`.

✅ Example 2: Simulating a Stochastic Process

r
CopyEdit

```r
set.seed(110)
S = rep(NA, 100)
S[1] = rnorm(1, 0, 1)
for(i in 2:100){
  S[i] = S[i - 1] + rnorm(1, 0, 1)
}
plot(S, main = "S", type = "l", xlab = "i", col = "darkred", lwd = 3)
abline(h = 0, col = "black", lty = 3, lwd = 2)
```

- Creates a simulated random walk (`S[i] = S[i-1] + N(0,1)`)
- `set.seed()` makes the result reproducible.
- `rnorm(1, 0, 1)` generates one random value from N(0,1)

🔄 While Loop

- A **while loop** runs as long as a condition is `TRUE`.

Syntax:

r
CopyEdit
```
while(condition){
  # code
}
```

-

✅ Example 1: Simple Increment

r
CopyEdit
```
i = 0
while(i < 10){
  i = i + 1
}
i   # Output: 10
```

- Loops until `i` reaches 10.

✅ Example 2: Using `break` in a While Loop

r
CopyEdit
```
set.seed(110)
while(TRUE){
  X = runif(1)
  Y = runif(1)
  if(X + Y < 1){
    break
  }
}
X + Y   # Output: < 1
```

- Runs until condition `X + Y < 1` is met.

- `break` exits the loop once satisfied.

🗝️ Key Takeaways:

- **For loops**: use when the number of repetitions is known.
- **While loops**: use when repeating until a condition is met.
- `break` is used to exit a loop early.
- Pre-allocating memory (e.g., `rep(NA, n)`) is good practice for performance.

📊 Chapter Notes: Graphics in R

🔹 Overview

- R has powerful built-in plotting capabilities.
- Most visualizations begin with the `plot()` function.
- You can create **scatter plots**, **line charts**, **histograms**, and more with simple syntax and many customizable options.

📍 1. The `plot()` Function

✅ **Basic Example:**

r

CopyEdit

```
# Define vectors
x = 1:10
y = x^2

# Create plot
plot(x, y,
     main = "Our First Plot!",
     xlab = "x",
```

```
     ylab = "y",
     xlim = c(0, 10),
     ylim = c(0, 100),
     type = "l",
     lwd = 5,
     col = "darkred")
```

✅ **Argument Breakdown:**

Argument	Purpose
`x`, `y`	X and Y data vectors for the plot
`main`	Title of the plot
`xlab`, `ylab`	Axis labels (quoted)
`xlim`, `ylim`	Axis ranges, using `c(min, max)`
`type`	Plot type: `"p"` = points, `"l"` = lines
`lwd`	Line width (thicker line = higher number)
`col`	Color of points or lines (accepts color names)

📉 2. Histograms with `hist()`

- Similar in structure to `plot()`
- Often used to explore **distribution** of a numeric variable
- You can set:
    - `main`, `xlab`, `col`, `breaks`, etc.

Example:

r

CopyEdit

```
hist(rnorm(100), main = "Histogram", col = "lightblue")
```

-

🧩 3. Plotting Multiple Graphs

✅ Using `par(mfrow = c(nrows, ncols))`

r

CopyEdit

```
# Set grid to 3x3
par(mfrow = c(3, 3))

# Generate 9 random plots
for(i in 1:9){
  plot(rnorm(i * 5),
       main = "",
       xlab = "x",
       ylab = "y",
       type = "p",
       pch = 16,
       col = "dodgerblue4")
}

# Reset layout to single plot
par(mfrow = c(1, 1))
```

🔸 **Notes:**

- `par(mfrow = c(3, 3))` divides the plot area into a grid.
- `pch = 16` sets the point style (solid circle).
- Always reset to `par(mfrow = c(1, 1))` when done.

➕ 4. Adding Lines with `abline()`

✅ **Example:**

r

CopyEdit

```r
# Define vectors
x = seq(-10, 10, by = 0.1)
y = x^2

# Plot curve
plot(x, y,
     main = "R Plot with Lines",
     xlab = "x",
     ylab = "y",
     type = "l",
     col = "darkred",
     lwd = 4)

# Add vertical and horizontal lines
abline(v = 0, lwd = 2, col = "dodgerblue4")  # vertical at x=0
abline(h = 40, lwd = 2, col = "dodgerblue4") # horizontal at y=40
```

🔸 **`abline()` Key Arguments:**

Argument	Effect
`v = x`	Draws vertical line at x
`h = y`	Draws horizontal line at y
`col`	Line color
`lwd`	Line width

🎨 Color Tip

- You can use named colors like `"dodgerblue4"` or `"darkred"`

To explore more colors:

r

CopyEdit

```
colors()      # Lists all color names
```

-

✅ Summary: R Plotting Tools Covered

Tool	Use
`plot()`	General plotting: points, lines, scatter
`hist()`	Histograms
`abline()`	Adds lines (vertical/horizontal) to plots
`par()`	Customizes the plotting area (e.g., grid view)
`pch`	Point character/type
`col`	Color settings for visual appeal

``