Lesson 1: Built-in Functions

- Understanding the purpose and usage of common built-in R functions (e.g., `mean()`, `sum()`, `max()`, `min()`, `length()`).
- Function arguments and default values.
- Using R's help system (`?`, `help()`, `example()`).

R comes with a vast collection of built-in functions that allow you to perform a wide variety of tasks without needing to write complex code yourself. These functions are the workhorses of R, enabling everything from simple calculations to complex statistical analyses.

Phase 1: Introduction to R and Fundamentals

Module 1.4: Functions and Packages

Lesson 1: Built-in Functions

This lesson introduces you to some of the most commonly used and important built-in functions in R. These functions provide powerful capabilities for performing mathematical operations, statistical analysis, string manipulation, and data structure manipulation.

---

1. Common Mathematical Functions These functions are used for basic mathematical calculations on numeric data.

- `sum()`: Calculates the sum of all elements in a numeric vector.
- `mean()`: Calculates the arithmetic mean (average) of a numeric vector.
- `median()`: Calculates the median of a numeric vector.
- `min()`: Finds the minimum value in a numeric vector.
- `max()`: Finds the maximum value in a numeric vector.
- `sqrt()`: Calculates the square root of numbers.
- `abs()`: Calculates the absolute value of numbers.
- `round()`: Rounds numbers to a specified number of decimal places.

Code Snippets:

```r
# Create a numeric vector for examples
numbers <- c(10, 25, 5, 30, 15, NA, 20)

print("--- Common Mathematical Functions ---")
```

```
## [1] "--- Common Mathematical Functions ---"
```

```r

print(paste("1. Sum of numbers:", sum(numbers, na.rm = TRUE))) # na.rm=TRUE removes NA for calculation
```

sum()

```
## [1] "1. Sum of numbers: 105"
```

```r
print(paste("   Sum of numbers (without na.rm):", sum(numbers)))
```

```
## [1] "   Sum of numbers (without na.rm): NA"
```

```r
print(paste("2. Mean of numbers:", mean(numbers, na.rm = TRUE)))
```

mean()

```
## [1] "2. Mean of numbers: 17.5"
```

```r
print(paste("   Mean of numbers (without na.rm):", mean(numbers)))
```

```
## [1] "   Mean of numbers (without na.rm): NA"
```

```r
print(paste("3. Median of numbers:", median(numbers, na.rm = TRUE)))
```

median()

```
## [1] "3. Median of numbers: 17.5"
```

```r
print(paste("4. Minimum number:", min(numbers, na.rm = TRUE)))
```

min()

```
## [1] "4. Minimum number: 5"
```

```r
print(paste("5. Maximum number:", max(numbers, na.rm = TRUE)))
```

max()

```
## [1] "5. Maximum number: 30"
```

```r
positive_num <- 16
print(paste("6. Square root of", positive_num, ":", sqrt(positive_num)))
```

sqrt()

```
## [1] "6. Square root of 16 : 4"
```

```r
vec_for_sqrt <- c(4, 9, 25)
print(paste("   Square roots of c(4,9,25):", paste(sqrt(vec_for_sqrt), collapse = ", ")))
```

```
## [1] "   Square roots of c(4,9,25): 2, 3, 5"
```

```r
negative_num <- -7.5
print(paste("7. Absolute value of", negative_num, ":", abs(negative_num)))
```

abs()

```
## [1] "7. Absolute value of -7.5 : 7.5"
```

```r
vec_for_abs <- c(-10, 5, -2)
print(paste("   Absolute values of c(-10,5,-2):", paste(abs(vec_for_abs), collapse = ", ")))
```

```
## [1] "   Absolute values of c(-10,5,-2): 10, 5, 2"
```

```
decimal_num <- 12.3456
print(paste("8. Round 12.3456 to 2 decimal places:", round(decimal_num, digits = 2)))
```

round()

```
## [1] "8. Round 12.3456 to 2 decimal places: 12.35"
```

```
print(paste("   Round 12.3456 to nearest integer:", round(decimal_num)))
```

```
## [1] "   Round 12.3456 to nearest integer: 12"
```

---

2. Statistical Functions R provides a rich set of functions for basic statistical analysis.

- `sd()`: Calculates the standard deviation.
- `var()`: Calculates the variance.
- `summary()`: Provides a statistical summary of a vector or data frame.
- `quantile()`: Calculates quantiles (e.g., 25th, 50th, 75th percentiles).

Code Snippets:

```
# Create a numeric vector for examples
data_points <- c(10, 12, 15, 13, 18, 11, 14)
scores_with_na <- c(85, 92, 78, NA, 65, 95, 88)

print("--- Statistical Functions ---")
```

```
## [1] "--- Statistical Functions ---"
```

```
print(paste("1. Standard deviation of data_points:", round(sd(data_points), 2)))
```

sd() - Standard Deviation

```
## [1] "1. Standard deviation of data_points: 2.69"
```

```
print(paste("   Standard deviation of scores_with_na (na.rm=TRUE):", round(sd(scores_with_na, na.rm = T
```

```
## [1] "   Standard deviation of scores_with_na (na.rm=TRUE): 10.94"
```

```
print(paste("2. Variance of data_points:", round(var(data_points), 2)))
```

var() - Variance

```
## [1] "2. Variance of data_points: 7.24"
```

```
print("3. Summary of data_points:")
```

summary() - Provides a statistical summary

```
## [1] "3. Summary of data_points:"
```

```
summary(data_points)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.00   11.50   13.00   13.29   14.50   18.00
```

```r
print("   Summary of a data frame:")
```

```
## [1] "   Summary of a data frame:"
```

```r
my_df <- data.frame(
    ID = 1:3,
    Name = c("A", "B", "C"),
    Value = c(10, 20, 30),
    stringsAsFactors = FALSE
)
summary(my_df)
```

```
##        ID          Name               Value
##  Min.   :1.0   Length:3           Min.   :10
##  1st Qu.:1.5   Class :character   1st Qu.:15
##  Median :2.0   Mode  :character   Median :20
##  Mean   :2.0                      Mean   :20
##  3rd Qu.:2.5                      3rd Qu.:25
##  Max.   :3.0                      Max.   :30
```

quantile()

```r
print("4. Quantiles of data_points:")
```

```
## [1] "4. Quantiles of data_points:"
```

```r
print(quantile(data_points)) # Default quantiles (0%, 25%, 50%, 75%, 100%)
```

```
##   0%  25%  50%  75% 100%
## 10.0 11.5 13.0 14.5 18.0
```

```r
print("   Specific quantiles (10%, 90%):")
```

```
## [1] "   Specific quantiles (10%, 90%):"
```

```r
print(quantile(data_points, probs = c(0.10, 0.90)))
```

```
##  10%  90%
## 10.6 16.2
```

3. Character Manipulation Functions These functions are used for working with text (character strings).

- `nchar()`: Counts the number of characters in a string or each element of a character vector.
- `paste()`: Concatenates (joins) strings. Can be used with `paste0()` for no separator.
- `grep()`: Searches for matches to a regular expression (pattern) within character vectors.
- `sub()`: Replaces the *first* occurrence of a pattern in strings.
- `gsub()`: Replaces *all* occurrences of a pattern in strings.

Code Snippets:

```r
# Create character vectors for examples
text1 <- "Hello R"
names_vec <- c("Alice", "Bob", "Charlie", "David")
sentence <- "R is a powerful statistical language, R is great."

print("--- Character Manipulation Functions ---")
```

```
## [1] "--- Character Manipulation Functions ---"
```

nchar()

```r
print(paste("1. Number of characters in '", text1, "':", nchar(text1)))
```

```
## [1] "1. Number of characters in ' Hello R ': 7"
```

```r
print(paste("   Characters in names_vec:", paste(nchar(names_vec), collapse = ", ")))
```

```
## [1] "   Characters in names_vec: 5, 3, 7, 5"
```

```r

combined_text_space <- paste("Welcome", "to", "R")
print(paste("2. Using paste() (default sep=' '):", combined_text_space))
```

paste() / paste0()

```
## [1] "2. Using paste() (default sep=' '): Welcome to R"
```

```r
combined_text_no_space <- paste0("Welcome", "to", "R")
print(paste("   Using paste0() (no separator):", combined_text_no_space))
```

```
## [1] "   Using paste0() (no separator): WelcometoR"
```

```r
combined_with_sep <- paste("Item", 1:3, sep = "-")
print(paste("   Paste with separator (sep='-'):", paste(combined_with_sep, collapse = ", ")))
```

```
## [1] "   Paste with separator (sep='-'): Item-1, Item-2, Item-3"
```

```r
#### grep() - find elements matching a pattern
# Returns indices by default, use value=TRUE to get the values
search_results_idx <- grep("a", names_vec)
print(paste("3. Indices of names containing 'a':", paste(search_results_idx, collapse = ", ")))
```

```
## [1] "3. Indices of names containing 'a': 3, 4"
```

```r
search_results_val <- grep("a", names_vec, value = TRUE)
print(paste("Names containing 'a':", paste(search_results_val, collapse = ", ")))
```

```
## [1] "Names containing 'a': Charlie, David"
```

```r

replaced_first <- sub("R", "Python", sentence)
print(paste("4. Replace first 'R' with 'Python':", replaced_first))
```

sub() - replace first occurrence

```
## [1] "4. Replace first 'R' with 'Python': Python is a powerful statistical language, R is great."
```

```r

replaced_all <- gsub("R", "Python", sentence)
print(paste("5. Replace all 'R' with 'Python':", replaced_all))
```

gsub() - replace all occurrences

```
## [1] "5. Replace all 'R' with 'Python': Python is a powerful statistical language, Python is great."
```

---

4. Vector/Matrix Manipulation Functions These functions are used to get information about or manipulate vectors and matrices.

- `length()`: Returns the number of elements in a vector.
- `dim()`: Returns the dimensions (rows, columns) of a matrix or data frame.
- `nrow()`: Returns the number of rows of a matrix or data frame.
- `ncol()`: Returns the number of columns of a matrix or data frame.
- `t()`: Transposes a matrix or data frame.
- `c()`: Combines vectors or lists into a new vector.
- `rbind()`: Combines vectors, matrices, or data frames by rows.
- `cbind()`: Combines vectors, matrices, or data frames by columns.

Code Snippets: #### Create vectors and matrices for examples

```r
my_vec <- c(10, 20, 30, 40, 50)
my_matrix <- matrix(1:9, nrow = 3, byrow = TRUE)
vec_to_combine_1 <- c("A", "B")
vec_to_combine_2 <- c("C", "D")

print("--- Vector/Matrix Manipulation Functions ---")
```

```
## [1] "--- Vector/Matrix Manipulation Functions ---"
```

length()

```r
print(paste("1. Length of my_vec:", length(my_vec)))
```

```
## [1] "1. Length of my_vec: 5"
```

```r

print("2. Dimensions of my_matrix (rows, cols):")
```

dim()

```
## [1] "2. Dimensions of my_matrix (rows, cols):"
```

```r
print(dim(my_matrix))
```

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

```r

print(paste("3. Number of rows in my_matrix:", nrow(my_matrix)))
```

nrow()

```
## [1] "3. Number of rows in my_matrix: 3"
```

```r

print(paste("4. Number of columns in my_matrix:", ncol(my_matrix)))
```

ncol()

```
## [1] "4. Number of columns in my_matrix: 3"
```

```r

print("5. Original Matrix:")
```

t() - Transpose

```
## [1] "5. Original Matrix:"
```

```r
print(my_matrix)
```

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

```r
print("   Transposed Matrix:")
```

```
## [1] "   Transposed Matrix:"
```

```r
print(t(my_matrix))
```

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

```r

combined_vec <- c(my_vec, 60, 70)
print(paste("6. Combined vector (c()):", paste(combined_vec, collapse = ", ")))
```

c() - Combine elements (coerces to common type)

```
## [1] "6. Combined vector (c()): 10, 20, 30, 40, 50, 60, 70"
```

```r
mixed_type_c <- c(1, "hello", TRUE) # Coerces all to character
print(paste("   Mixed type combined with c():", paste(mixed_type_c, collapse = ", ")))
```

```
## [1] "   Mixed type combined with c(): 1, hello, TRUE"
```

```r
print(paste("   Class of mixed_type_c:", class(mixed_type_c)))
```

```
## [1] "   Class of mixed_type_c: character"
```

rbind() - Row bind (combines by rows)

```r
row1 <- c(1, 2, 3)
row2 <- c(4, 5, 6)
combined_rows <- rbind(row1, row2)
print("7. Row-bound matrix:")
```

```
## [1] "7. Row-bound matrix:"
```

```r
print(combined_rows)
```

```
##      [,1] [,2] [,3]
## row1    1    2    3
## row2    4    5    6
```

cbind() - Column bind (combines by columns)

```r
col1 <- c(10, 20)
col2 <- c(30, 40)
```

```r
combined_cols <- cbind(col1, col2)
print("8. Column-bound matrix:")
```

```
## [1] "8. Column-bound matrix:"
```

```r
print(combined_cols)
```

```
##      col1 col2
## [1,]   10   30
## [2,]   20   40
```

```r
# Note: rbind/cbind can also be used with data frames,
# but require same number of columns/rows respectively.
```

5. Data Frame Specific Functions These functions are particularly useful for inspecting and working with data frames, the most common data structure for tabular data.

- `head()`: Displays the first few rows of a data frame (default 6).
- `tail()`: Displays the last few rows of a data frame (default 6).
- `str()`: Displays the internal structure of an R object, very useful for data frames to see column names and types.
- `summary()`: Provides a statistical summary of each column in a data frame.
- `colnames()`: Gets or sets the column names of a data frame.
- `rownames()`: Gets or sets the row names of a data frame.

Code Snippets: #### Create a data frame for examples

```r
employee_data <- data.frame(
    EmpID = c("E001", "E002", "E003", "E004", "E005"),
    Name = c("John", "Jane", "Peter", "Mary", "Chris"),
    Department = c("HR", "IT", "Sales", "HR", "IT"),
    Salary = c(60000, 85000, 70000, 62000, 90000),
    YearsExp = c(5, 10, 7, 6, 12),
    stringsAsFactors = FALSE
)

print("--- Data Frame Specific Functions ---")
```

```
## [1] "--- Data Frame Specific Functions ---"
```

```r
print("Original Data Frame:")
```

```
## [1] "Original Data Frame:"
```

```r
print(employee_data)
```

```
##   EmpID  Name Department Salary YearsExp
## 1  E001  John         HR  60000        5
## 2  E002  Jane         IT  85000       10
## 3  E003 Peter      Sales  70000        7
## 4  E004  Mary         HR  62000        6
## 5  E005 Chris         IT  90000       12
```

head()

```r
print("1. First 3 rows of employee_data:")
```

```
## [1] "1. First 3 rows of employee_data:"
```

```r
print(head(employee_data, n = 3))
```

```
##   EmpID  Name Department Salary YearsExp
## 1  E001  John         HR  60000        5
## 2  E002  Jane         IT  85000       10
## 3  E003 Peter      Sales  70000        7
```

```r

print("2. Last 2 rows of employee_data:")
```

tail()

```
## [1] "2. Last 2 rows of employee_data:"
```

```r
print(tail(employee_data, n = 2))
```

```
##   EmpID  Name Department Salary YearsExp
## 4  E004  Mary         HR  62000        6
## 5  E005 Chris         IT  90000       12
```

```r

print("3. Structure of employee_data (str()):")
```

str()

```
## [1] "3. Structure of employee_data (str()):"
```

```r
str(employee_data)
```

```
## 'data.frame':    5 obs. of  5 variables:
##  $ EmpID     : chr  "E001" "E002" "E003" "E004" ...
##  $ Name      : chr  "John" "Jane" "Peter" "Mary" ...
##  $ Department: chr  "HR" "IT" "Sales" "HR" ...
##  $ Salary    : num  60000 85000 70000 62000 90000
##  $ YearsExp  : num  5 10 7 6 12
```

summary()

```r
print("4. Summary of employee_data:")
```

```
## [1] "4. Summary of employee_data:"
```

```r
summary(employee_data)
```

```
##     EmpID               Name            Department            Salary
##  Length:5           Length:5           Length:5           Min.   :60000
##  Class :character   Class :character   Class :character   1st Qu.:62000
##  Mode  :character   Mode  :character   Mode  :character   Median :70000
##                                                           Mean   :73400
##                                                           3rd Qu.:85000
##                                                           Max.   :90000
##     YearsExp
##  Min.   : 5
##  1st Qu.: 6
##  Median : 7
##  Mean   : 8
##  3rd Qu.:10
##  Max.   :12
```

colnames() - Get column names

```
print("5. Column names of employee_data:")
```

```
## [1] "5. Column names of employee_data:"
```

```
print(colnames(employee_data))
```

```
## [1] "EmpID"      "Name"       "Department" "Salary"     "YearsExp"
```

```
# colnames() - Set column names (example: rename 'YearsExp')
colnames(employee_data)[5] <- "YearsExperience"
print("   Updated column names:")
```

```
## [1] "   Updated column names:"
```

```
print(colnames(employee_data))
```

```
## [1] "EmpID"           "Name"            "Department"      "Salary"
## [5] "YearsExperience"
```

```
print("   Data frame with updated column name:")
```

```
## [1] "   Data frame with updated column name:"
```

```
print(employee_data) # Show effect of renaming
```

```
##   EmpID  Name Department Salary YearsExperience
## 1  E001  John         HR  60000               5
## 2  E002  Jane         IT  85000              10
## 3  E003 Peter      Sales  70000               7
## 4  E004  Mary         HR  62000               6
## 5  E005 Chris         IT  90000              12
```

rownames() - Get row names (usually just numbers by default)

```
print("6. Row names of employee_data:")
```

```
## [1] "6. Row names of employee_data:"
```

```
print(rownames(employee_data))
```

```
## [1] "1" "2" "3" "4" "5"
```

```
# rownames() - Set row names (e.g., to EmpID)
# rownames(employee_data) <- employee_data$EmpID
# print("   Updated row names (using EmpID):")
# print(rownames(employee_data))
```

This lesson provided a foundational understanding of many common and essential built-in functions in R, covering mathematical, statistical, character, vector/matrix, and data frame operations. Familiarity with these functions will significantly enhance your ability to perform data manipulation and analysis in R.

Next, we will proceed to Lesson 2: Creating Custom Functions.