R Programming Language Cheatsheet

A comprehensive cheat sheet for the R programming language, covering data structures, syntax, data manipulation, statistical analysis, and common functions.

Data Structures

Vectors

Definition	A one-dimensional array of elements of the same data type.

Creating Vectors	`c(element1, element2, ...)`
	`vector(mode = "numeric", length = 5)`
	`seq(from = 1, to = 10, by = 2)`
	`rep(x = 1:3, times = 2)`

Accessing Elements	`vector[index]`
	`vector[c(index1, index2)]`
	`vector[start:end]`

Common Functions	`length(vector)`
	`is.vector(object)`
	`as.vector(object)`

Example	`my_vector <- c(1, 2, 3, 4, 5)` `print(my_vector[3]) # Output: 3`

Matrices

Definition	A two-dimensional array of elements of the same data type.
Creating Matrices	`matrix(data, nrow, ncol, byrow = FALSE, dimnames = NULL)`

Accessing Elements	`matrix[row, column]`
	`matrix[row, ] # Entire row`
	`matrix[, column] # Entire column`

Common Functions	`row(matrix)`
	`col(matrix)`
	`dim(matrix)`
	`is.matrix(object)`
	`as.matrix(object)`

Example	`my_matrix <- matrix(1:9, nrow = 3, ncol = 3)` `print(my_matrix[2, 3]) # Output: 5`

Lists

Definition	An ordered collection of elements, which can be of different data types.
Creating Lists	`list(element1, element2, ...)` `list(name1 = element1, name2 = element2, ...)`
Accessing Elements	`list[[index]]` `list$name`

Common Functions	`length(list)`
	`is.list(object)`
	`as.list(object)`
	`names(list)`

Example	`my_list <- list(name = "John", age = 30, grades = c(85, 90, 92))` `print(my_list$age) # Output: 30`

Data Frames

Definition	A table-like structure with columns of potentially different data types.
Creating Data Frames	`data.frame(col1 = vector1, col2 = vector2, ...)` `read.csv("file.csv")`

Accessing Elements	`dataframe$column`
	`dataframe[row, column]`
	`dataframe[row, ]`
	`dataframe[, column]`

Common Functions	`row(dataframe)`
	`col(dataframe)`
	`dim(dataframe)`
	`names(dataframe)`
	`str(dataframe)`
	`summary(dataframe)`

Example	`my_df <- data.frame(name = c("Alice", "Bob"), age = c(25, 30))` `print(my_df$name) # Output: "Alice" "Bob"`

Syntax and Basic Operations

Operators

Arithmetic	`+`, `-`, `*`, `/`, `^` (exponentiation), `%%` (modulo), `%/%` (integer division)
Relational	`>`, `<`, `>=`, `<=`, `==` (equal to), `!=` (not equal to)
Logical	`&` (AND), `|` (OR), `!` (NOT)
Assignment	`<-`, `=`, `<<-` (global assignment)

Example	`x <- 10`
	`y <- 5`
	`z <- x + y # z is now 15`

Control Flow

if Statement	`if (condition) {`
	`  # Code to execute if condition is TRUE`
	`}`

if...else Statement	`if (condition) {`
	`  # Code to execute if condition is TRUE`
	`} else {`
	`  # Code to execute if condition is FALSE`
	`}`

for Loop	`for (variable in sequence) {`
	`  # Code to execute for each element in the sequence`
	`}`

while Loop	`while (condition) {`
	`  # Code to execute while condition is TRUE`
	`}`

Example	`for (i in 1:5) {`
	`  print(i)`
	`}`

Functions

Definition	Reusable blocks of code that perform a specific task.

Defining a Function	`function_name <- function(argument1, argument2, ...) {`
	`  # Function body`
	`  return(value)`
	`}`

Calling a Function	`function_name(value1, value2, ...)`

Example	`add <- function(x, y) {`
	`  return(x + y)`
	`}`
	`result <- add(3, 5) # result is now 8`

Data Manipulation

dplyr Package

Description	A powerful package for data manipulation.
Key Functions	`filter()`: Filter rows based on conditions. `select()`: Select columns. `arrange()`: Arrange rows in order. `mutate()`: Add new columns or modify existing ones. `summarize()`: Compute summary statistics. `group_by()`: Group data by one or more variables.

Example	`library(dplyr)`
	`df <- data.frame(group = c("A", "A", "B", "B"), value = c(10, 15, 20, 25))`
	`df %>% group_by(group) %>% summarize(mean_value = mean(value))`

tidyr Package

Description	A package for tidying data.
Key Functions	`gather()`: Convert wide format to long format. `spread()`: Convert long format to wide format. `separate()`: Separate one column into multiple columns. `unite()`: Unite multiple columns into one.

Example	`library(tidyr)`
	`df <- data.frame(id = 1:2, var1 = c(10, 15), var2 = c(20, 25))`
	`gather(df, key = "variable", value = "value", var1, var2)`

Data Subsetting

Using Indices	`data[rows, columns]`
Using Logical Vectors	`data[logical_vector, ]`
Using `subset()` function	`subset(data, condition)`
Example	`df <- data.frame(id = 1:5, value = c(10, 15, 20, 25, 30))` `df[df$value > 15, ]`

Statistical Analysis

Descriptive Statistics

Functions

`mean(x)`: Mean of vector `x`.
`median(x)`: Median of vector `x`.
`sd(x)`: Standard deviation of vector `x`.
`var(x)`: Variance of vector `x`.
`quantile(x, probs)`: Quantiles of vector `x`.
`summary(x)`: Summary statistics of vector `x`.

Example

```r
x <- c(1, 2, 3, 4, 5)
mean(x) # Output: 3
sd(x) # Output: 1.581139
summary(x)
```

Hypothesis Testing

t-tests

`t.test(x, y, alternative = "two.sided", mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.95)`

- `x`, `y`: Numeric vectors.
- `alternative`: Type of test ("two.sided", "less", "greater").
- `mu`: Null hypothesis value.
- `paired`: TRUE for paired t-test.
- `var.equal`: TRUE for equal variances.

Chi-squared Test

`chisq.test(x, y, correct = TRUE)`

- `x`, `y`: Numeric vectors or matrices.
- `correct`: Apply Yates' continuity correction.

Example

```r
x <- rnorm(50, mean = 10, sd = 2)
y <- rnorm(50, mean = 12, sd = 2)
t.test(x, y)
```

Linear Regression

Function

`lm(formula, data)`

- `formula`: Model formula (e.g., `y ~ x`).
- `data`: Data frame.

Example

```r
df <- data.frame(x = 1:10, y = 2*(1:10) + rnorm(10))
model <- lm(y ~ x, data = df)
summary(model)
```