This cheat sheet provides a quick reference for essential R programming commands, helping you perform data manipulation, visualization, and statistical analysis with confidence. It covers foundational topics like installing packages and understanding R's data structures, alongside advanced tasks such as building models and applying machine learning techniques.

Each section includes concise syntax and practical examples to illustrate how R commands are used in real-world scenarios. You'll find guidance on working with vectors, lists, matrices, and data frames, performing common data wrangling tasks like filtering and summarizing, and creating visualizations such as histograms, bar plots, and boxplots. The cheat sheet also highlights R's capabilities for statistical analysis with commands like mean, lm, and cor.

Designed for clarity and accessibility, this resource is ideal for data analysts, statisticians, and programmers seeking to enhance their workflows in R. Whether you're exploring data, developing algorithms, or building reproducible reports, this cheat sheet ensures you can quickly apply R's powerful tools to your projects.

Table of Contents

Basics

Syntax for	How to use	Explained
Install Package	`install.packages("dplyr")`	Installs the `dplyr` package.
Load Package	`library(dplyr)`	Loads the `dplyr` package into the current R session.
Assignment	`x <- 5`	Assigns value `5` to the variable `x`.
Print Output	`print(x)`	Prints the value of `x` to the console.
Literals and Data Types	`TRUE, 125, 12.5, "Hello"`	Examples of logical, integer, numeric, and character literals in R.
Extracting Numbers from Strings	`library(readr)` `data_frame <- mutate(data_frame, column = parse_number(column))`	Uses `parse_number` to extract numeric values from string columns.
Basic String Indexing	`str_sub("Dataquest is awesome", 1, 9)`	Extracts `"Dataquest"` as a substring by specifying start and end indices.

Data Structures

Syntax for	How to use	Explained
Create Vector	`c(1, 2, 3)`	Combines elements into a vector.
Create List	`list(a=1, b="two")`	Creates a list with named elements.
Create Matrix	`matrix(1:6, nrow=2)`	Creates a matrix with 2 rows and 3 columns.
Create Data Frame	`data.frame(a=1:3, b=4:6)`	Creates a data frame with columns `a` and `b`.
Access Element	`df$a | df[1, 1]`	Performs a logical OR operation between a column and a specific element.
Loading stringr package	`library(stringr)`	Loads the `stringr` library to work with strings in R.
Opening a JSON File	`f <- fromJSON('filename.json')`	Loads a JSON file into an R dataframe using the jsonlite package.
Creating a List	`new_list <- list("data scientist", c(50000,40000), "programming experience")`	Defines a list containing diverse data types.

Data Manipulation

Syntax for	How to use	Explained
Filter Rows	`filter(df, a > 2)`	Filters rows where column a is greater than 2.
Select Columns	`select(df, a, b)`	Selects specific columns by name.
Mutate Columns	`mutate(df, c = a + b)`	Adds a new column `c` as sum of `a` and `b`.
Summarize Data	`summarize(df, avg=mean(a))`	Calculates mean of column `a` and returns as `avg`.
Arrange Rows	`arrange(df, desc(a))`	Sorts rows by column `a` in descending order.
Importing Data	`data <- read_csv("name_of_file_with_data.csv")`	Imports dataset into R using the `read_csv` function from `readr`.
Summing Values Across Rows	`df %>% mutate(new_column_name = rowSums(.[1:3]))`	Sums specified columns for each row and adds as a new column.
Summing Values Across Columns	`df %>% bind_rows(tibble(total = colSums(across(everything()))))`	Sums specified rows for each column and adds as a new row.
Importing CSV files	`dataframe <- read_csv("name_of_the_dataset.csv")`	Read CSV files into R using readr's `read_csv()` for efficient data import.

Data Visualization

Syntax for	How to use	Explained
Creating a Basic Plot	`data %>% ggplot()`	Initialize a basic `ggplot2` chart without specifying any aesthetics.
Creating Subplots	`data %>% ggplot(aes(x = variable_1, y = variable_2)) + geom_line() + facet_wrap(~variable_3)`	Plots subsets of data in separate facets.
Creating Bar Chart	`data_frame %>% ggplot(aes(x = variable_1, y = variable_2)) + geom_col()`	Create a bar chart using `ggplot2`, mapping variables to x and y axes.
Plotting multiple columns	`data %>% ggplot(aes(x = variable_1)) + geom_line(aes(y = variable_2)) + geom_line(aes(y = variable_3))`	Plots multiple columns on the same axes using `ggplot2`.
Scatterplots	`ggplot(data = uber_trips, aes(x = distance, y = cost)) + geom_point()`	Generate scatterplots to visualize bivariate relationships in `ggplot2`.
Scatterplots with Labels	`ggplot(data = df, aes(x = predictor, y = response)) + geom_point() + scale_y_continuous(labels = scales::comma)`	Create scatterplots with y-axis labels formatted using commas instead of scientific notation.

Data Visualization

Syntax for	How to use	Explained
Scatterplot with Comma Labels	`ggplot(data = df, aes(x = predictor, y = response)) + scale_y_continuous(labels = scales::comma) + geom_point()`	Plots a scatterplot with y-axis labels in comma format.
Scatterplot with Groups	`ggplot(data = df, aes(x = predictor, y = response)) + geom_point() + facet_wrap(~ categorical_variable, ncol = 2)`	Creates scatterplots of response vs predictor, grouped by a categorical variable.
Scatterplot with Groups	`ggplot(data = df, aes(x = predictor, y = response)) + geom_point() + facet_wrap(~ categorical_variable, ncol = 2)`	Creates scatterplots of response vs predictor, grouped by a categorical variable.
Vertical Bar Chart	`ggplot(data = df, aes(x = col)) + geom_bar()`	Creates a vertical bar chart to visualize counts of data.
Grouped Bar Plot	`ggplot(data = df, aes(x = col_1, fill = col_2)) + geom_bar(position = "dodge")`	Creates a grouped bar plot to compare frequency distributions of categorical variables.

Statistics & Probability

Syntax for	How to use	Explained
Mean	`mean(x)`	Calculates the mean of vector `x`.
Median	`median(x)`	Calculates the median of vector `x`.
Weighted Mean	`mean <- weighted.mean(x = distribution, w = weights)`	Computes the weighted mean of a numerical vector using specific weights.
Standard Deviation	`sd(x)`	Calculates the standard deviation of `x`.
Correlation	`cor(x, y)`	Calculates correlation between `x` and `y`.
Linear Model	`lm(y ~ x, data=df)`	Fits a linear regression model.
Types of Variables	`# Example Variables: Age (Quantitative), Gender (Qualitative)`	Classify variables as Quantitative (numerical) or Qualitative (categorical).
P-Value Decision Threshold	`if (p_value < 0.05) { print('Reject null hypothesis') } else { print('Fail to reject null hypothesis') }`	Decide on hypothesis rejection using a common p-value threshold of `0.05`.

Statistics & Probability

Syntax for	How to use	Explained
Chi-Squared Distribution	`pchisq(3.84, df = 1)`	Calculates the cumulative probability for a chi-squared distribution with specific degrees of freedom.
Chi-Squared Test	`pchisq(q = 10, df = 5)`	Calculate cumulative probability for a chi-squared statistic of 10 with 5 degrees of freedom.
Multi-category Chi-squared Test	`data <- table(income$sex, income$high_income)`	Performs a chi-squared test on the given contingency table.
Computing Mode in R	`compute_mode <- function(vector) {counts_df <- tibble(vector) %>% group_by(vector) %>% summarise(frequency=n()) %>% arrange(desc(frequency)); counts_df$vector[1]}`	Defines a function to calculate the mode of a given vector using `dplyr` functions.
Calculate Z-score	`z_score <- function(value, vector) { (value - mean(vector)) / sd(vector) }`	This calculates the Z-score for a value relative to a vector's distribution.
Chi-Squared Distribution	`pchisq(3.84, df = 1)`	Calculates the cumulative probability for a chi-squared distribution with specific degrees of freedom.

Syntax for	How to use	Explained
Simulate Coin Toss	`set.seed(1) coin_toss <- function() { if (runif(1) <= 0.5) 'HEADS' else 'TAILS' }`	Simulates a random coin toss using R's uniform random numbers.
Addition Rule for Probability	`P(A ∪ B) = P(A) + P(B) - P(A ∩ B)`	Formula to calculate probabilities of unions of events, adjusting for overlap in non-exclusive cases.
Independent Events	`P(A ∩ B) = P(A) * P(B)`	Probability of independent events occurs as product of individual probabilities.
Product Rule in Experiments	`total_outcomes <- a * b`	Calculate the total outcomes for two independent experiments using the product rule.
Uniform Distribution	`# Assuming all outcomes have equal chance outcomes <- c(1, 2, 3, 4, 5, 6) probabilities <- rep(1/6, 6) paste('Outcome:', outcomes, 'Probability:', probabilities)`	Demonstrates a uniform distribution for a dice roll, where outcomes equally likely.

Statistics & Probability

Syntax for	How to use	Explained
Conditional Probability Calculation	`P_A_given_B <- P_A_and_B / P_B`	Compute P(A|B) given the probability of A and B, and probability of B.
Conditional Probability	`P_A_given_B <- length(intersect(A, B)) / length(B)`	Compute P(A|B) using set cardinalities.
Conditional Probability Definition	`P_A_given_B <- 1 - P_Ac_given_B`	Conditional probabilities are interrelated; P(A|B) and its complement P(Ac|B) can be calculated mutually.
Independence	`P_A_and_B <- P_A * P_B`	Defines independent events: joint probability equals product of individual probabilities.

Programming

Syntax for	How to use	Explained
If Statement	`if (x > 0) print("positive")`	Executes code if condition is true.
For Loop	`for (i in 1:3) print(i)`	Iterates over a sequence.
While Loop	`while (x < 5) x <- x + 1`	Repeats code while the `x < 5` condition is true.

Syntax for functions	`function_name <- function(input) {`
	`  # Code to manipulate the input`
	`  return(output)`
	`}`
		Defines a reusable function structure in R.

Define Function	`f <- function(a, b) a + b`	Defines a function with two arguments.
Apply Function	`apply(m, 1, sum)`	Applies a function over rows/columns of a matrix.
Exponentiation	`3^5`	Calculates 3 raised to the power of 5.
Creating Dates	`ymd('20/04/21')`	Converts a string into a Date object using 'year-month-day'.
Creating Dates from Strings	`ymd("20/04/21")`	Converts a string to a date object using the specified format.
Define Window Frame	`ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING`	Defines a window frame including one row before and after the current row for computations.

Machine Learning

Syntax for	How to use	Explained
Fitting a Linear Model	`lm_fit <- lm(response ~ predictor, data = df)`	Fit a linear regression model with a response and a predictor variable.
Visualize Residuals	`library(ggplot2)` `ggplot(data.frame(residuals = lm_fit$residuals), aes(x = residuals)) + geom_histogram()`	Visualize the distribution of residuals to check the linear model's fit.

Hyperparameter Grid Search	`knn_grid <- expand.grid(k = 1:20)`
	`knn_model <- train(tidy_price ~ accommodates + bathrooms + bedrooms, data = training_data, method = "knn", trControl = train_control, preProcess = c("center", "scale"), tuneGrid = knn_grid)`
	`plot(knn_model)`
		Performs grid search to optimize `k` for k-NN model and visualizes results.

Naive Bayes Algorithm	`P(Spam|w1,...,wn) ∝ P(Spam) * ΠiP(wi|Spam)`	Classifies messages as spam using conditional probabilities.

File I/O

Syntax for	How to use	Explained
Read CSV	`read.csv("file.csv")`	Reads a CSV file into a data frame.
Write CSV	`write.csv(df, "file.csv")`	Writes a data frame to a CSV file.
Read RDS	`readRDS("file.rds")`	Reads an RDS file into R.
Write RDS	`saveRDS(df, "file.rds")`	Saves an object as an RDS file.
List Files	`list.files()`	Lists files in the current directory.