Statistical Inference and Modelling Summary

Daniel Weronski Falcó

October 2024

Contents

1 Introduction

This cheat sheet provides a concise summary of key concepts in statistical inference and modeling. It is structured to help navigate through definitions, usages, and code snippets for statistical tools commonly used in data science.

1.1 Classical vs Fisherian Inference

Statistical inference is broadly classified into two paradigms:

- **Classical Inference (Frequentist Approach)**: Focuses on the long-run frequency of events. Relies heavily on hypothesis testing, confidence intervals, and p-values.
- **Fisherian Inference**: An approach that uses likelihood to measure support for different hypotheses. Unlike the classical approach, Fisher's inference does not require specification of alternative hypotheses.

1.2 Key Concepts in Inference

Statistical inference involves making decisions or drawing conclusions about a population based on a sample. The two primary goals are:

1. **Estimation**: Determining the approximate value of a population parameter.
2. **Hypothesis Testing**: Assessing evidence provided by the data against a specified hypothesis.

1.3 When to Use Different Methods

- **Parametric Tests** (e.g., t-tests, ANOVA): Assume underlying distribution of the data.
- **Non-Parametric Tests** (e.g., Mann-Whitney U test, Kruskal-Wallis test): Do not assume any specific distribution.

1.4 Useful R Functions for Inference

- `t.test()` — Performs one and two-sample t-tests.
- `chisq.test()` — Performs Chi-squared test for independence.
- `prop.test()` — Tests for the equality of proportions.

2 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a crucial step in understanding and summarizing the main characteristics of a dataset. It involves the use of various statistical tools and visualization techniques to identify patterns, detect outliers, and gain insights before conducting formal modeling or hypothesis testing.

2.1 Numerical Summaries

Numerical summaries provide quantitative insights into the dataset using measures of central tendency and variability. Key statistics include:

- **Mean** (μ): The average of all observations in a dataset.
- **Median** (Q_2): The middle value of the ordered dataset, less sensitive to outliers.
- **Mode**: The most frequently occurring value(s) in the dataset.
- **Variance** (σ^2): Measure of how far observations spread out from the mean.
- **Standard Deviation** (σ): The square root of the variance.
- **Interquartile Range** (IQR): Difference between the 75th percentile (Q_3) and 25th percentile (Q_1).

2.2 Graphical Summaries

Graphical representations help to visualize the distribution and relationships between variables. Common plots include:

- **Histograms**: Show the frequency distribution of a continuous variable.
- **Box Plots**: Display the spread of the data along with outliers.
- **Scatter Plots**: Used to visualize the relationship between two continuous variables.
- **Bar Charts**: Represent categorical data using bars of different heights.

2.3 Useful R Functions for EDA

- `summary()` — Provides a numerical summary of a dataset.
- `boxplot()` — Plots a box-and-whisker plot.
- `hist()` — Creates a histogram for a given variable.
- `plot()` — Generic plotting function, often used for scatter plots.

2.4 Example R Code for EDA

Below is an example of performing basic EDA on a sample dataset in R:

```r
# Load necessary library
library(ggplot2)

# Example dataset: mtcars
data("mtcars")

# Numerical summary
summary(mtcars)
```

```r

# Histogram of Miles per Gallon (mpg)
hist(mtcars$mpg, main="Histogram of MPG", xlab="Miles Per Gallon",
   col="lightblue", breaks=10)

# Boxplot for distribution of mpg by number of cylinders
boxplot(mpg ~ cyl, data=mtcars, main="Boxplot of MPG by Cylinder",
   xlab="Number of Cylinders", ylab="Miles Per Gallon", col="orange")

# Scatter plot of Horsepower vs MPG
plot(mtcars$hp, mtcars$mpg, main="Scatter Plot of Horsepower vs MPG",
    xlab="Horsepower", ylab="Miles Per Gallon", col="blue", pch=19)
```

Listing 1: Basic EDA in R

2.5 Key Points

- EDA helps to identify anomalies and understand the structure of the data.
- Both numerical and graphical summaries are essential in the preliminary stages of data analysis.
- Visualizations are often more intuitive and revealing compared to raw statistical measures.

3 Hypothesis Testing

Hypothesis testing is a statistical method used to make decisions or draw conclusions about a population based on sample data. It involves formulating two competing hypotheses and using sample data to determine which hypothesis is more plausible.

3.1 Key Concepts in Hypothesis Testing

- **Null Hypothesis (H_0)**: The hypothesis that there is no effect or no difference. This is the hypothesis that we aim to test against.
- **Alternative Hypothesis (H_1)**: The hypothesis that there is a significant effect or difference.
- **Significance Level (α)**: The probability of rejecting the null hypothesis when it is actually true. Common values are 0.05, 0.01, or 0.1.
- **p-value**: The probability of obtaining results as extreme as those observed, assuming H_0 is true. A small p-value (less than α) indicates strong evidence against H_0.
- **Type I Error**: Incorrectly rejecting H_0 when it is true (false positive).
- **Type II Error**: Failing to reject H_0 when H_1 is true (false negative).

3.2 Types of Hypothesis Tests

The choice of test depends on the nature of the data and the research question. Some common hypothesis tests include:

- **t-tests** — Used to compare means.
  - **One-sample t-test**: Compares the mean of a single group to a known value.
  - **Two-sample t-test**: Compares means between two independent groups.
  - **Paired t-test**: Compares means from the same group at different times.
- **ANOVA (Analysis of Variance)** — Compares means among three or more groups.
- **Chi-squared Test** — Assesses relationships between categorical variables.
- **Fisher's Exact Test** — An alternative to Chi-squared when sample sizes are small.

3.3 Useful R Functions for Hypothesis Testing

- `t.test()` — Performs one and two-sample t-tests, as well as paired t-tests.
- `aov()` — Fits an ANOVA model.
- `chisq.test()` — Performs the Chi-squared test for independence.
- `fisher.test()` — Conducts Fisher's Exact Test for count data.
- `wilcox.test()` — Performs non-parametric tests (e.g., Mann-Whitney U test).

3.4 Example R Code for Hypothesis Testing

The following example shows how to perform various hypothesis tests in R:

```r
# Load necessary data
data(mtcars)

# One-sample t-test
t.test(mtcars$mpg, mu = 20) # Test if mean mpg is different from 20

# Two-sample t-test
t.test(mpg ~ cyl, data = mtcars) # Test if mean mpg differs by number
    of cylinders

# Paired t-test
pre_weight <- c(70, 75, 80, 78, 72)
post_weight <- c(68, 74, 79, 76, 71)
t.test(pre_weight, post_weight, paired = TRUE) # Compare pre and post
    weights

# Chi-squared Test for independence
table_data <- table(mtcars$cyl, mtcars$gear)
chisq.test(table_data)

# ANOVA
fit <- aov(mpg ~ factor(cyl), data = mtcars) # Test if mpg differs
   across cylinders
summary(fit)
```

Listing 2: Hypothesis Testing in R

3.5 General Procedure for Hypothesis Testing

The standard approach for hypothesis testing involves the following steps:

1. **State the Hypotheses**:
   - Null hypothesis (H_0): Example: There is no difference in mean mileage between 4-cylinder and 6-cylinder cars.
   - Alternative hypothesis (H_1): Example: The mean mileage is different between the two groups.
2. **Set the Significance Level** (α): Common choices are 0.05 or 0.01.
3. **Calculate the Test Statistic**: Use the appropriate formula depending on the test being conducted.
4. **Find the p-value and Make a Decision**:
   - If $p \leq \alpha$: Reject H_0, and conclude that there is sufficient evidence to support H_1.
   - If $p > \alpha$: Fail to reject H_0, and conclude that there is not enough evidence to support H_1.

3.6 Choosing the Appropriate Hypothesis Test

Scenario	**Test to Use**	**R Function**
Comparing the mean of a sample to a known value	One-sample t-test	`t.test(x, mu = value)`
Comparing means between two independent groups	Two-sample t-test	`t.test(x group)`
Comparing means within the same group at different times	Paired t-test	`t.test(pre, post, paired = TRUE)`
Testing for relationship between categorical variables	Chi-squared Test	`chisq.test(table)`
Comparing means across multiple groups	ANOVA	`aov()`

4 Confidence Intervals

Confidence intervals provide a range of plausible values for an unknown population parameter based on sample data. They are used to quantify the uncertainty around a point estimate and are typically expressed as a percentage (e.g., 95% confidence interval).

4.1 Key Concepts in Confidence Intervals

- **Point Estimate**: A single value estimate of a population parameter (e.g., sample mean).
- **Confidence Level**: The proportion of times the interval would contain the true parameter if we were to repeat the sampling process many times (e.g., 95%).
- **Margin of Error**: The maximum expected difference between the point estimate and the true population parameter. Influenced by sample size and variability.
- **Confidence Interval Formula**:

$$\text{CI} = \text{Point Estimate} \pm \text{Critical Value} \times \text{Standard Error}$$

4.2 Confidence Interval for the Mean (Known Variance)

When the population variance σ^2 is known, the confidence interval for the mean μ is calculated as:

$$\text{CI} = \bar{x} \pm z_{\alpha/2} \times \frac{\sigma}{\sqrt{n}}$$

Where:

- $\bar{x}$ — Sample mean.
- $z_{\alpha/2}$ — Critical value from the standard normal distribution for the given confidence level.
- σ — Population standard deviation.
- n — Sample size.

4.3 Confidence Interval for the Mean (Unknown Variance)

When the population variance is unknown, use the sample standard deviation s and the t-distribution:

$$\text{CI} = \bar{x} \pm t_{\alpha/2,n-1} \times \frac{s}{\sqrt{n}}$$

Where $t_{\alpha/2,n-1}$ is the critical value from the t-distribution with $n-1$ degrees of freedom.

4.4 Confidence Interval for Proportions

For categorical data, the confidence interval for a proportion p is calculated using:

$$\text{CI} = \hat{p} \pm z_{\alpha/2} \times \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

Where:

- $\hat{p}$ — Sample proportion.
- n — Sample size.
- $z_{\alpha/2}$ — Critical value from the standard normal distribution.

4.5 Useful R Functions for Confidence Intervals

- `t.test()` — Provides confidence intervals for one-sample, two-sample, and paired t-tests.
- `prop.test()` — Computes confidence intervals for proportions.
- `confint()` — Calculates confidence intervals for model parameters.

4.6 Example R Code for Confidence Intervals

Here's how to compute various types of confidence intervals in R:

```r
# One-sample t-test for confidence interval
x <- c(12.9, 14.2, 13.7, 15.3, 14.8, 13.6, 14.0)
t.test(x) # 95% CI for the mean

# Confidence interval for a proportion
prop.test(x = 34, n = 230, conf.level = 0.95) # 95% CI for proportion

# Confidence interval for a difference in means
group1 <- c(15, 17, 16, 14, 18)
group2 <- c(12, 14, 13, 15, 11)
t.test(group1, group2) # 95% CI for the difference in means

# Confidence intervals for regression coefficients
model <- lm(mpg ~ hp + wt, data = mtcars)
confint(model, level = 0.95) # 95% CI for model coefficients
```

Listing 3: Confidence Interval Calculations in R

4.7 Choosing the Appropriate Confidence Interval

Scenario	**Parameter of Interest**	**Distribution**	**R Function**
Population mean with known variance	μ	Normal distribution	Manual calculation or `qnorm()`
Population mean with unknown variance	μ	t-distribution	`t.test()`
Population proportion	p	Binomial distribution	`prop.test()`
Difference in two population means	$\mu_1 - \mu_2$	t-distribution	`t.test()`
Regression coefficients	β	Normal approximation	`confint()`

4.8 Key Points

- Higher confidence levels (e.g., 99%) produce wider intervals compared to lower confidence levels (e.g., 90%).
- Confidence intervals are not probabilities; they describe the range of plausible values for the population parameter.
- If a confidence interval for a mean difference contains zero, it suggests no significant difference.

5 Distributions

Understanding different probability distributions is crucial in statistical modelling and inference. Each distribution has unique properties and applications, and choosing the correct one for your data is essential.

5.1 Key Distributions and Their Properties

- **Normal Distribution**:
  - Symmetrical, bell-shaped distribution defined by its mean μ and variance σ^2.
  - Widely used in inferential statistics due to the Central Limit Theorem.
  - **R Functions**: `dnorm()`, `pnorm()`, `rnorm()`.
- **Binomial Distribution**:
  - Models the number of successes in a fixed number of independent Bernoulli trials.
  - Defined by parameters n (number of trials) and p (probability of success).
  - **R Functions**: `dbinom()`, `pbinom()`, `rbinom()`.
- **Poisson Distribution**:
  - Models the count of events occurring in a fixed interval of time or space.
  - Parameterized by λ (rate of occurrence).
  - **R Functions**: `dpois()`, `ppois()`, `rpois()`.
- **Exponential Distribution**:
  - Models the time between successive events in a Poisson process.
  - Defined by the rate parameter λ.
  - **R Functions**: `dexp()`, `pexp()`, `rexp()`.
- **Chi-squared Distribution**:
  - Distribution of the sum of squared standard normal variables.
  - Commonly used in hypothesis testing and constructing confidence intervals.
  - **R Functions**: `dchisq()`, `pchisq()`, `rchisq()`.
- **Student's t Distribution**:
  - Similar to the normal distribution but with heavier tails.
  - Useful for small sample sizes or when population variance is unknown.
  - **R Functions**: `dt()`, `pt()`, `rt()`.

5.2 Choosing the Right Distribution

The choice of distribution depends on the type of data and the problem context. Consider the following guidelines:

- **Continuous Data**: Use the normal distribution if the data is symmetrically distributed. Use the exponential distribution for time-to-event data.
- **Discrete Data**: Use the binomial distribution for binary outcomes and the Poisson distribution for count data.
- **Small Sample Size**: Use the t-distribution if the sample size is small and the variance is unknown.

5.3 Useful R Functions for Distributions

- `dnorm()`, `dpois()`, `dbinom()` — Probability density functions.
- `pnorm()`, `ppois()`, `pbinom()` — Cumulative distribution functions.
- `rnorm()`, `rpois()`, `rbinom()` — Random number generation.
- `qnorm()`, `qpois()`, `qbinom()` — Quantile functions.

5.4 Example R Code for Distributions

Here's how to generate and visualize different distributions in R:

```r
# Set up parameters
mu <- 0
sigma <- 1
lambda <- 2
n <- 10
p <- 0.3

# Generate and visualize a normal distribution
x <- seq(-3, 3, by = 0.1)
y <- dnorm(x, mean = mu, sd = sigma)
plot(x, y, type = "l", main = "Normal Distribution", xlab = "X", ylab
    = "Density")

# Generate and plot a binomial distribution
binom_data <- dbinom(0:n, size = n, prob = p)
barplot(binom_data, main = "Binomial Distribution (n=10, p=0.3)",
   xlab = "Number of Successes")

# Generate and plot a Poisson distribution
pois_data <- dpois(0:20, lambda = lambda)
barplot(pois_data, main = "Poisson Distribution (lambda=2)", xlab = "
   Number of Events")

# Generate and plot an exponential distribution
exp_data <- rexp(1000, rate = lambda)
hist(exp_data, main = "Exponential Distribution (rate=2)", xlab = "
   Time", breaks = 20)
```

Listing 4: R Code for Generating and Visualizing Distributions

5.5 When to Use Each Distribution

Distribution	Type of Data	Common Usage	R Function
Normal	Continuous	Modeling measurement errors, heights, weights	`rnorm(), dnorm()`
Binomial	Discrete	Modeling binary outcomes (e.g., coin flips)	`rbinom(), dbinom()`
Poisson	Discrete	Modeling count data (e.g., number of calls per hour)	`rpois(), dpois()`
Exponential	Continuous	Modeling time to next event (e.g., failure times)	`rexp(), dexp()`
Chi-squared	Continuous	Goodness-of-fit tests	`rchisq(), dchisq()`
Student's t	Continuous	Small sample data, unknown variance	`rt(), dt()`

5.6 Key Points

- Understanding the properties of each distribution is essential for selecting the appropriate model.
- Use visualizations (e.g., histograms, density plots) to assess whether your data fits a particular distribution.
- Some statistical tests rely on specific distributions; always verify assumptions before proceeding.

6 Regression and Modelling

Regression analysis is a powerful statistical tool used to understand the relationship between one or more independent variables and a dependent variable. It is widely used for prediction, inference, and uncovering causal relationships.

6.1 Linear Regression

Linear regression models the relationship between a dependent variable y and one or more independent variables $x_1, x_2, \ldots, x_n$ using a linear function.

Simple Linear Regression Model:

$$y = \beta_0 + \beta_1 x + \epsilon$$

Where:

- β_0 — Intercept (the value of y when $x = 0$).
- β_1 — Slope (change in y for a one-unit change in x).
- ϵ — Random error term, assumed to be normally distributed with mean 0.

6.1.1 Multiple Linear Regression

Multiple linear regression models the relationship between a dependent variable and multiple independent variables:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

Where $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients for each independent variable.

6.1.2 Assumptions of Linear Regression

- **Linearity**: The relationship between the independent and dependent variables is linear.
- **Independence**: Observations are independent.
- **Homoscedasticity**: Constant variance of the error terms across all levels of independent variables.
- **Normality**: The residuals (errors) follow a normal distribution.

6.2 Logistic Regression

Logistic regression is used when the dependent variable is categorical, often binary (e.g., success/failure). Instead of modelling the dependent variable directly, logistic regression models the log odds of the probability of an event occurring:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$$

Where:

- p — Probability of success (e.g., $p(\text{event} = 1)$).
- $\beta_0, \beta_1, \ldots, \beta_n$ — Coefficients estimated using maximum likelihood.

6.3 Model Evaluation Metrics

Once a model is fitted, it is essential to evaluate its performance using appropriate metrics:

- **R-squared** — Proportion of variance explained by the model. Higher values indicate better fit.
- **Adjusted R-squared** — Adjusted version of R-squared that penalises adding irrelevant predictors.
- **Mean Squared Error (MSE)** — Average of the squared differences between observed and predicted values.
- **Akaike Information Criterion (AIC)** — Measures the goodness of fit while penalising for the number of predictors.

6.4 Useful R Functions for Regression and Modelling

- `lm()` — Fits a linear regression model.
- `summary()` — Summarises the results of a linear or logistic model.
- `glm()` — Fits a generalised linear model, including logistic regression.
- `predict()` — Makes predictions using a fitted model.
- `anova()` — Compares models using ANOVA.

6.5 Example R Code for Regression Analysis

Here is an example of fitting linear and logistic regression models in R:

```r
# Load the dataset
data(mtcars)

# Linear Regression: Predicting MPG from Horsepower and Weight
linear_model <- lm(mpg ~ hp + wt, data = mtcars)
summary(linear_model)

# Visualising the fit
plot(mtcars$hp, mtcars$mpg, main = "MPG vs Horsepower", xlab = "
    Horsepower", ylab = "MPG")
abline(linear_model, col = "red")

# Logistic Regression: Predicting Automatic vs Manual Transmission
mtcars$am <- factor(mtcars$am, labels = c("Automatic", "Manual"))
logistic_model <- glm(am ~ hp + wt, data = mtcars, family = "binomial
    ")
summary(logistic_model)

# Predict probabilities
predict(logistic_model, type = "response")
```

Listing 5: Regression Analysis in R

6.6 Choosing Between Models

When building regression models, consider the following:

- **Linear vs Logistic Regression**:
  - Use linear regression when the dependent variable is continuous.
  - Use logistic regression for binary outcomes.
- **Regularisation (Ridge/Lasso)**:
  - Use regularisation techniques when dealing with high-dimensional data to prevent overfitting.
  - **R Function**: `glmnet()` for regularised models.

6.7 When to Use Different Regression Models

Model Type	**Dependent Variable Type**	**Purpose**	**R Function**
Linear Regression	Continuous	Predict a continuous outcome	`lm()`
Logistic Regression	Binary/Categorical	Classify into two categories	`glm(family = "binomial")`
Ridge/Lasso Regression	Continuous	Prevent overfitting in high-dimensional data	`glmnet()`
Poisson Regression	Count Data	Model count outcomes (e.g., number of events)	`glm(family = "poisson")`

6.8 Key Points

- Always check the assumptions of your regression model (e.g., linearity, homoscedasticity).
- Evaluate model performance using appropriate metrics (e.g., R-squared, AIC).
- Visualise residuals to check for patterns and validate model assumptions.

7 Advanced Topics

This section covers some of the more complex statistical concepts and methods that go beyond basic hypothesis testing and regression. Understanding these topics is essential for tackling more sophisticated analytical problems and gaining deeper insights from data.

7.1 Central Limit Theorem (CLT)

The Central Limit Theorem is a fundamental concept in statistics that states that the distribution of the sample mean will approximate a normal distribution, regardless of the original distribution of the population, as the sample size becomes large.

Mathematical Statement:

If $X_1, X_2, \ldots, X_n$ are independent and identically distributed random variables with mean μ and variance σ^2, the sample mean $\bar{X}$ is approximately normally distributed:

$$\bar{X} \sim N\left(\mu, \frac{\sigma^2}{n}\right)$$

Key Points:

- The larger the sample size, the better the approximation to the normal distribution.
- This property justifies the use of normal-based methods (e.g., t-tests) even when the underlying data is not normally distributed.

7.2 Law of Large Numbers (LLN)

The Law of Large Numbers states that as the sample size increases, the sample mean $\bar{X}$ converges to the true population mean μ.

Types of LLN:

- **Weak Law**: The sample mean converges in probability to the population mean.
- **Strong Law**: The sample mean converges almost surely to the population mean.

Implications:

- Ensures that larger samples provide more reliable estimates of population parameters.
- The variability of the sample mean decreases as the sample size increases.

7.3 Non-Parametric Tests

Non-parametric tests do not assume a specific distribution for the data, making them useful for small samples or data that do not meet parametric assumptions.

- **Mann-Whitney U Test**:
  - Compares the ranks of two independent samples.
  - Analogous to the two-sample t-test but does not require normality.
  - **R Function**: `wilcox.test()`.
- **Kruskal-Wallis Test**:
  - Extends the Mann-Whitney U test to compare more than two groups.
  - Equivalent to one-way ANOVA for ranked data.
  - **R Function**: `kruskal.test()`.

7.4 Bootstrap Methods

Bootstrap methods involve resampling with replacement from a dataset to estimate the distribution of a statistic. They are useful for constructing confidence intervals and assessing the stability of estimates without relying on parametric assumptions.

Procedure:

1. Draw B bootstrap samples from the original dataset.
2. Calculate the statistic of interest (e.g., mean, median) for each bootstrap sample.
3. Use the distribution of these B statistics to estimate confidence intervals or other properties.

Example R Code for Bootstrapping:

```r
# Load the boot library
library(boot)

# Create a function for the statistic of interest
mean_stat <- function(data, indices) {
  return(mean(data[indices]))
}

# Sample data
data <- c(12.9, 14.2, 13.7, 15.3, 14.8, 13.6, 14.0)

# Perform bootstrapping
bootstrap_results <- boot(data, statistic = mean_stat, R = 1000)

# Bootstrap confidence interval
boot.ci(bootstrap_results, type = "bca")
```

Listing 6: Bootstrap Example in R

When to Use Bootstrapping:

- When sample sizes are small and standard parametric assumptions are not valid.
- When you need robust estimates of standard errors and confidence intervals.

7.5 Choosing Between Advanced Techniques

Technique	Data Type	Purpose	R Function
Central Limit Theorem	Continuous/Discrete	Justifies normal approximation	Manual calculations
Law of Large Numbers	Continuous/Discrete	Ensures convergence of sample mean	Manual calculations
Mann-Whitney U Test	Non-normal, Ranked Data	Compares two independent groups	`wilcox.test()`
Kruskal-Wallis Test	Non-normal, Ranked Data	Compares more than two groups	`kruskal.test()`
Bootstrap	Any	Estimate confidence intervals	`boot()`

7.6 Key Points

- Advanced techniques are necessary for handling non-standard data and assessing the robustness of models.

- Non-parametric tests are useful when the assumptions of parametric tests are violated.
- Bootstrap methods provide flexible, assumption-free approaches for estimating confidence intervals.

8 Useful R Functions and Syntax

R is a powerful language for data analysis and statistical modeling, providing a vast collection of functions for handling, visualising, and modelling data. This section summarises some of the most commonly used R functions to help streamline data analysis.

8.1 Data Handling and Manipulation

- `read.csv("file.csv")` — Reads a CSV file into a dataframe.
- `head(df, n)` — Displays the first n rows of the dataframe.
- `subset(df, condition)` — Extracts rows from a dataframe based on a condition.
- `merge(df1, df2, by = "column")` — Merges two dataframes based on a common column.
- `dplyr` Package Functions:
  - `filter(df, condition)` — Selects rows that meet a condition.
  - `select(df, columns)` — Selects specific columns.
  - `mutate(df, new_var = expression)` — Creates new variables.
  - `group_by(df, column)` — Groups data by a specified column.
  - `summarise(df, new_var = summary_function)` — Aggregates grouped data.

8.2 Basic Plotting Functions

- `plot(x, y)` — Generic plotting function for scatter plots, line plots, etc.
- `boxplot(x)` — Creates a box-and-whisker plot for visualising the distribution.
- `hist(x)` — Creates a histogram of a vector x.
- `barplot(height)` — Plots a bar chart with the specified heights.
- `pairs(df)` — Creates a matrix of scatterplots for pairwise comparison of variables.

8.3 Advanced Visualisation with `ggplot2`

`ggplot2` is a versatile package for creating complex and aesthetically pleasing graphics. It uses a grammar of graphics approach to layer different elements.

- `ggplot(data, aes(x, y))` — Initialises a ggplot object with data and aesthetic mappings.
- `geom_point()` — Adds points (scatter plot).
- `geom_boxplot()` — Adds a box plot.
- `geom_histogram(binwidth = value)` — Creates a histogram.
- `facet_wrap( ~ factor)` — Creates a series of plots for each level of a factor.
- `theme_minimal()` — Applies a minimal theme to the plot.

8.4 Statistical Modelling Functions

- `lm(formula, data)` — Fits a linear regression model.
- `glm(formula, family = "binomial")` — Fits a logistic regression model.
- `anova(model1, model2)` — Compares two nested models using ANOVA.
- `predict(model, newdata)` — Generates predictions based on a fitted model.
- `summary(model)` — Summarises the results of a model fit.

8.5 Cheat Sheet Table: Common R Functions

Category	Function	Description
Data Import	`read.csv("file.csv")`	Reads a CSV file into R.
Data Inspection	`head(df)`	Displays the first few rows of a dataframe.
Subsetting	`subset(df, condition)`	Extracts rows based on a condition.
Merging	`merge(df1, df2, by = "col")`	Merges two dataframes by a common column.
Summarising	`summarise(df, new_var = mean(col))`	Computes summary statistics.
Basic Plots	`plot(x, y)`	Creates scatter or line plots.
Histograms	`hist(x)`	Plots a histogram of the variable x.
Box Plots	`boxplot(x)`	Creates a box plot to show data spread.
ggplot2	`ggplot(df, aes(x, y))`	Initialises a ggplot object.
Linear Modelling	`lm(mpg ~ hp, data)`	Fits a linear regression model.
Logistic Modelling	`glm(am ~ hp, data, family = "binomial")`	Fits a logistic regression model.

8.6 R Markdown for Reproducible Reports

R Markdown allows you to create dynamic documents that include code, text, and visualisations. Use the following basic syntax:

- Code chunks: ```` ```{r} ... ``` ````
- Inline R code: `` `r variable` ``.
- Text formatting: `**bold**`, `*italics*`, `# Heading`.
- Export formats: PDF, HTML, Word.

8.7 Additional Resources and Tips

- Use the `?function_name` to access documentation for any R function.
- Leverage the `tidyverse` packages for a cohesive data science workflow.
- Use RStudio's auto-complete features to explore function arguments and documentation.