Statistics

Dr. Polla Fattah

What You Will Learn

By the end of this crash course, you'll be able to:

- Summarize and visualize data
- Measure variability and shape
- Construct and interpret confidence intervals
- Conduct hypothesis testing (t-tests, ANOVA, chi-square)
- Analyze categorical and numeric outcomes
- Choose appropriate statistical tests for your thesis
- Communicate statistical results effectively

Key Questions Every Researcher Must Answer

- Does my algorithm really perform better than others?
- How confident can I be in my results?
- Is the difference I observed due to chance?
- How do I report my findings objectively?

Statistics provides the **language of evidence** in science.

Why Postgraduates Can't Ignore Statistics

Without statistical analysis, a statement like:

> *"I ran my new sorting algorithm and it seemed faster than QuickSort."*
> *is anecdotal at best.*

In a thesis defense, your examiners will ask:

- How many times did you test it?
- What was the average run-time?
- How much variability was there?
- Could this be due to randomness?

- Was your test design valid?

  These are not programming questions — they are **statistical**.

Statistics in Experimental Design and Validation

Activity	**Statistical Role**
Comparing algorithms	t-tests, ANOVA
Classifier performance	Confidence intervals
Survey data	Chi-square, proportions
Benchmarking systems	Regression
System reliability	Error rate, standard deviation

Good experimental design includes:

- Stating clear hypotheses
- Designing fair trials
- Collecting and summarizing results
- Performing appropriate statistical analysis

Our Guiding Scenario: The Algorithm Showdown

To make this course practical, we'll follow one running example:

You're defending your MSc thesis:

"Comparing MergeSort and QuickSort performance."

You've:

- Run each algorithm 30 times
- Recorded run-time (ms)
- Collected performance data

Your core research question:

> *"Is MergeSort significantly faster than QuickSort?"*

We will use this example throughout to introduce and apply statistical concepts.

Descriptive Statistics: Center and Spread

To begin answering our thesis question (“Is MergeSort faster than QuickSort?”), we need to **describe** the data.
We’ll start with:

- Measures of central tendency
- Measures of variability
- How they behave in symmetric and skewed data
- Practical examples with our simulated results

Understanding Central Tendency

Central tendency describes the “middle” of the data — where values tend to cluster.

Three common measures:

- **Mean**: Arithmetic average
- **Median**: Middle value when sorted
- **Mode**: Most frequent value (less useful in continuous data)

When Do These Measures Differ?

In a perfectly **symmetric distribution**, all three measures are equal:

$$\text{Mean} = \text{Median} = \text{Mode}$$

In a **right-skewed** distribution (tail on the right):

Mean > Median > Mode

In a **left-skewed** distribution (tail on the left):

Mean < Median < Mode

This helps analysts infer the **shape** of the data from summary statistics.

Simulating Our Data (QuickSort vs MergeSort)

We'll work with the following data throughout this presentation:

```r
set.seed(42)
qs <- rnorm(30, mean = 52, sd = 6)
ms <- rnorm(30, mean = 48, sd = 5)

algo_data <- data.frame(
  Algorithm = rep(c("QuickSort", "MergeSort"), each =
         30),
  Time_ms = c(qs, ms)
)
```

Each row represents one run of either algorithm. We'll use this data in visualizations and statistical analysis.

Displaying the Data as a Table

It’s useful to see the actual values used:

Trial	QuickSort	MergeSort
1	60.23	50.28
2	48.61	51.52
3	54.18	53.18
4	55.80	44.96
5	54.43	50.52
6	51.36	39.41
7	61.07	44.08
8	51.43	43.75
9	64.11	35.93

Trial	QuickSort	MergeSort
10	51.62	48.18

Mean and Median of Each Algorithm

```
mean(qs); median(qs)
```

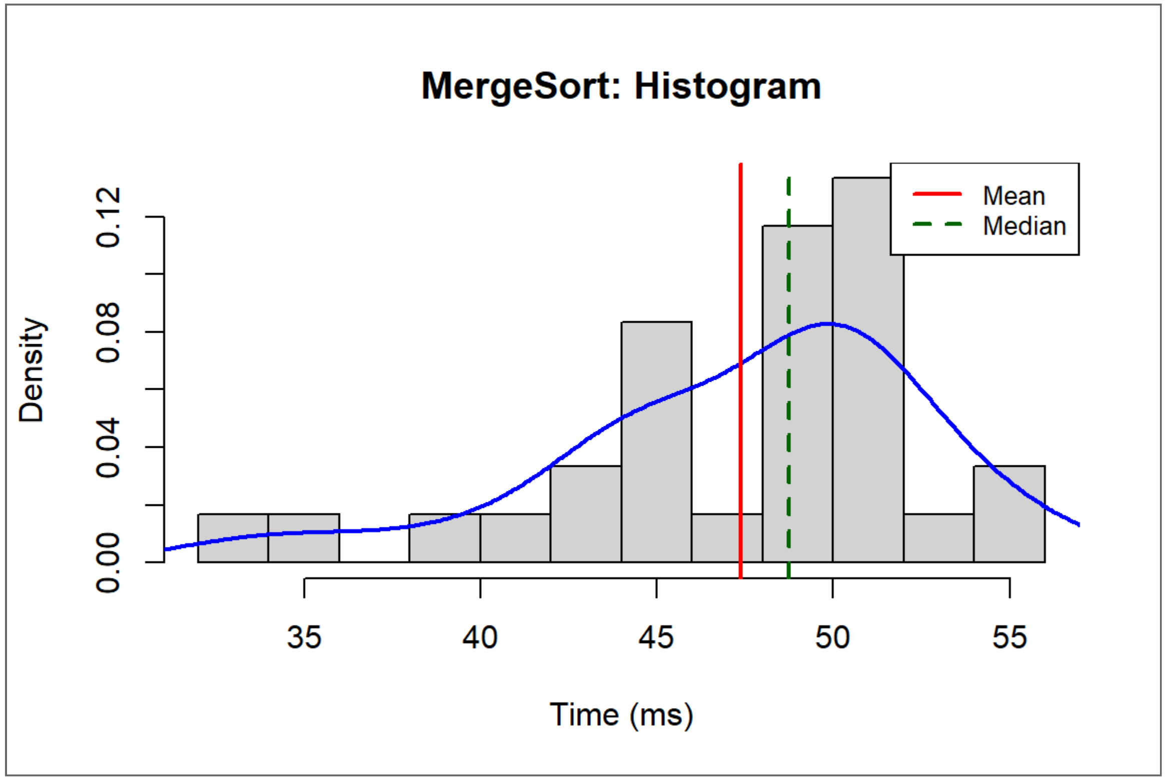
```
## [1] 52.41152
```

```
## [1] 51.39765
```

Do the mean and median differ? That can tell us about symmetry or skewness.

Visualizing the Center of Data

Let's visualize both distributions using histograms with **mean** and **median** lines.



Understanding Variability

Two datasets can have the same mean but very different **spread**. Measures of variability include:

- **Range**: max - min
- **Deviations** from the mean
- **Variance**: average squared deviation
- **Standard Deviation (SD)**: square root of variance
- **IQR**: interquartile range (Q3 - Q1)

Why Standard Deviation?

Standard deviation (SD) is the most common way to quantify **how much the data spread** around the mean.

- Low SD = tightly clustered
- High SD = widely spread

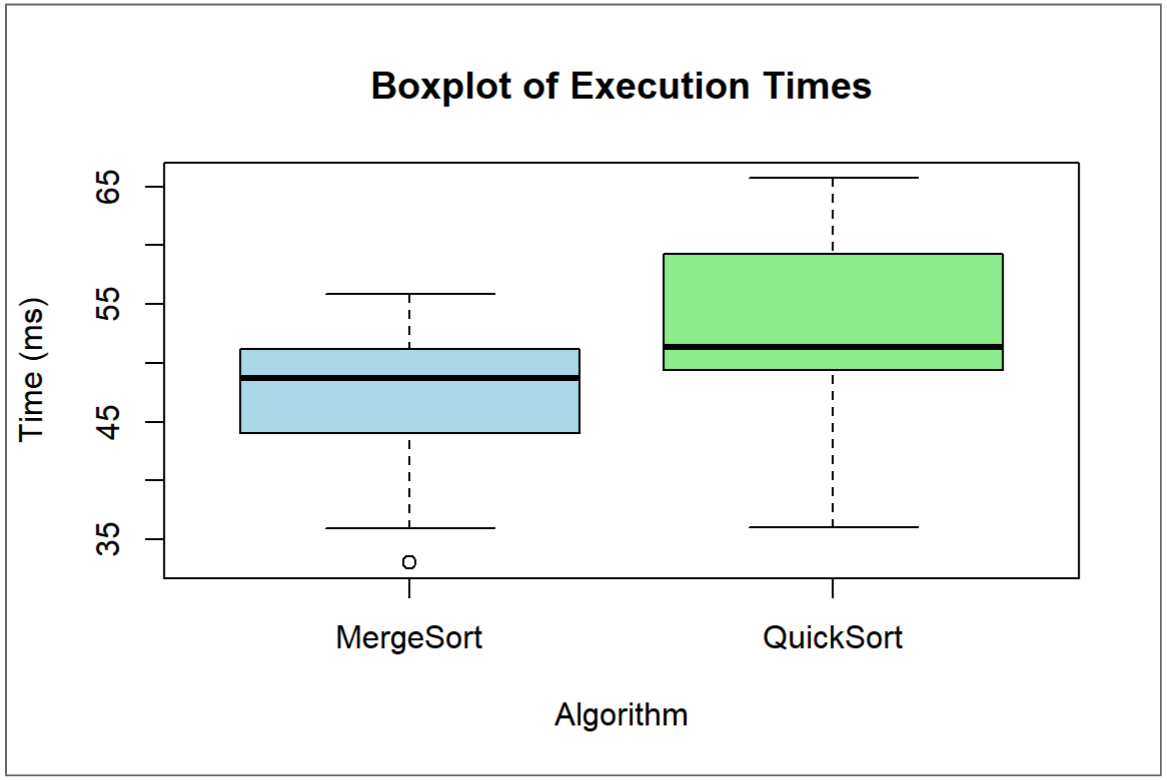
```
tapply(algo_data$Time_ms, algo_data$Algorithm, sd)
```

```
## MergeSort QuickSort
##  5.250282  7.530168
```

Boxplots: A Visual Summary

Boxplots show:

- Median (center line)
- Q1 and Q3 (box edges)
- IQR (box width)
- Outliers (dots beyond whiskers)


Boxplot of Execution Times
Time (ms)
35
45
55
65
MergeSort
QuickSort
Algorithm

What Can Boxplots Tell Us?

Boxplots help you:

- Compare medians
- Spot skewed distributions (asymmetry)
- Detect outliers
- Evaluate overall spread (IQR vs total range)

Use them early and often during data exploration!

Distributions and Statistical Assumptions

Before conducting statistical tests, we must understand **how the data are distributed**.
This section covers:

- What distributions are
- The normal (bell-shaped) distribution
- How to assess normality in R
- When and why normality matters

What Is a Distribution?

A **distribution** describes how values are spread out.

Key properties:

- **Symmetry** vs **skewness**
- **Unimodal** vs **bimodal**
- **Continuous** vs **discrete**

Examples:

- Uniform: all values equally likely
- Normal: bell-shaped
- Skewed: lopsided toward one side

The Normal Distribution

The **normal distribution** (Gaussian) is:

- Symmetric
- Bell-shaped
- Defined by **mean** and **standard deviation**

In a normal distribution:

- Mean = Median = Mode
- Most values lie near the center

The Empirical Rule

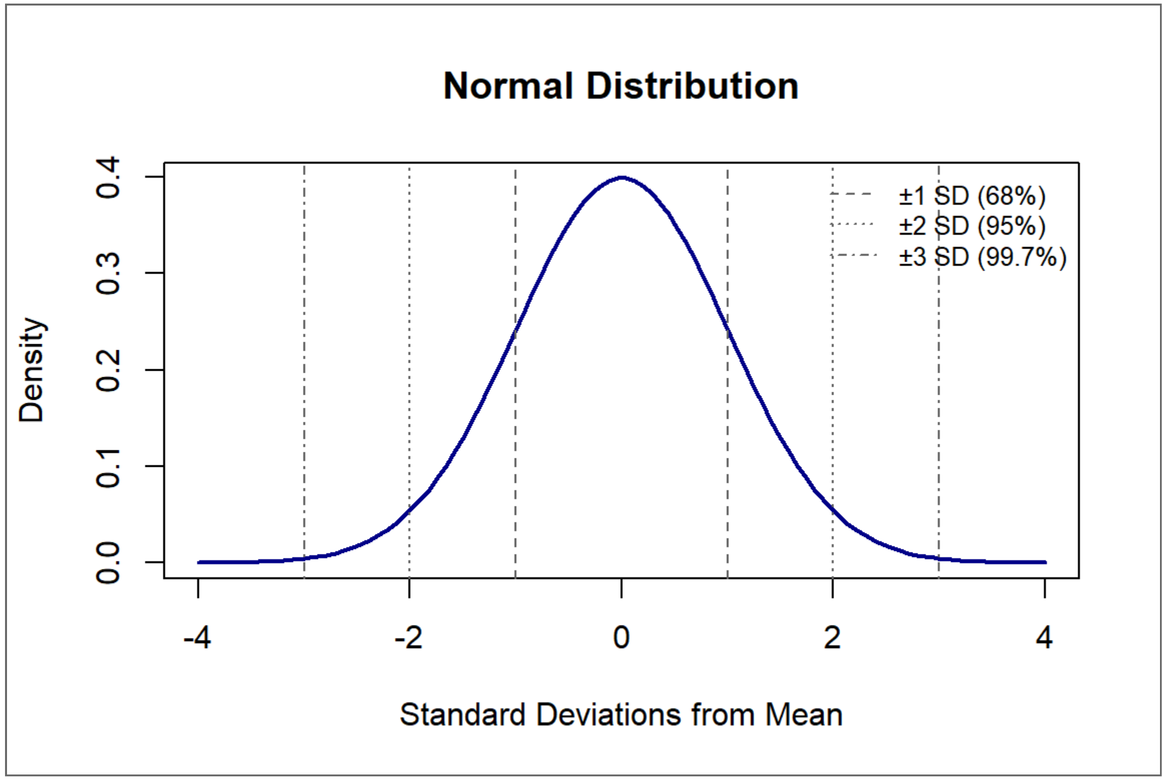
The **Empirical Rule** states:

- ~68% of data lie within **±1 SD**
- ~95% within **±2 SD**
- ~99.7% within **±3 SD**

This rule is essential for:

- Understanding **confidence intervals**
- Identifying **outliers**

Visualizing the Normal Curve



Why Does Normality Matter?

Many statistical tests assume normality — especially:

- **t-tests**
- **ANOVA**
- **Regression**

We assume:

> *"If the sample comes from a normal distribution, the test results are valid."*

Fortunately, many tests are **robust** to small violations.

Central Limit Theorem (CLT)

The **CLT** says:

> *The distribution of the sample mean becomes approximately* ***normal*** *as the sample size increases, even if the data are not normal.*

This allows us to:

- Use t-tests and CIs on non-normal data
- Rely on sampling distributions instead of raw data

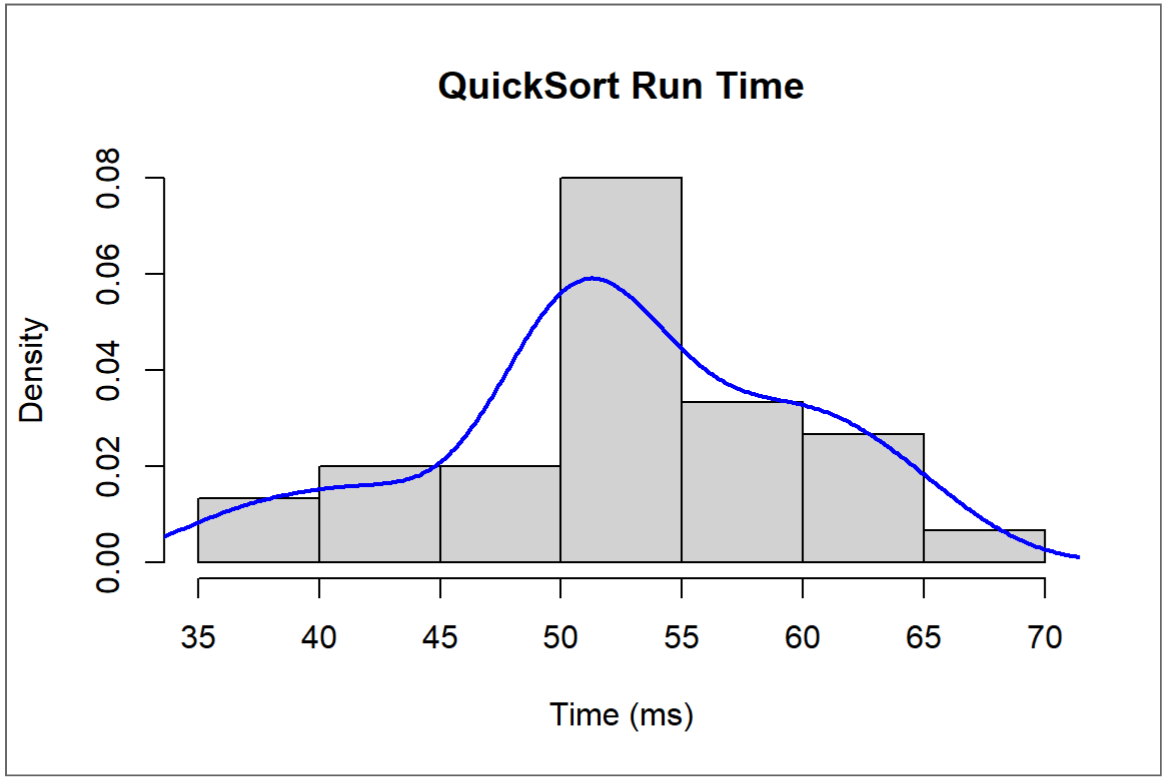
CLT usually kicks in around **$n \geq 30$**

Graphical Checks for Normality

Three useful visual tools:

- **Histogram** — see symmetry, outliers
- **Density Plot** — smoother histogram
- **QQ Plot** — plot of quantiles vs theoretical normal values

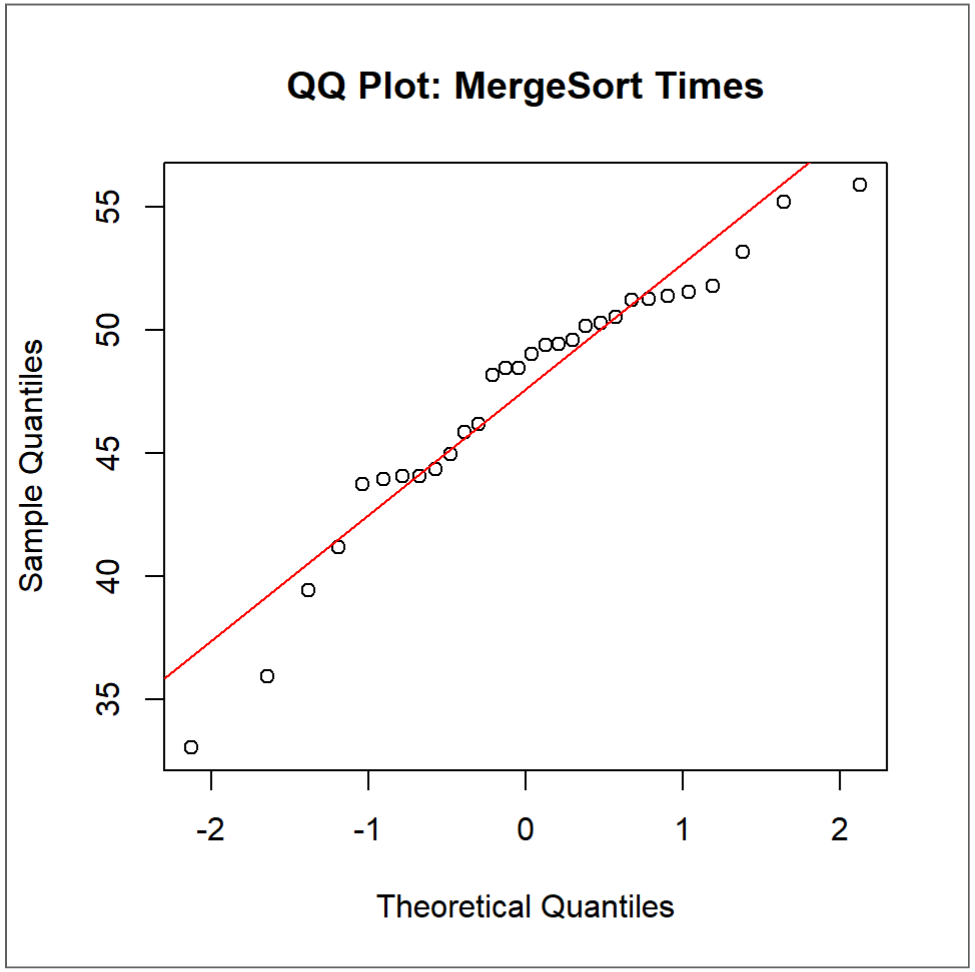
Example: Histogram + Density Plot


QuickSort Run Time
Density
0.00
0.02
0.04
0.06
0.08
35
40
45
50
55
60
65
70
Time (ms)

QQ Plot (Quantile–Quantile Plot)

In a QQ plot:

- Points should lie on a straight line
- Curves at ends $\rightarrow$ skewness
- Strong deviations $\rightarrow$ non-normal


QQ Plot: MergeSort Times
Sample Quantiles
35
40
45
50
55
-2
-1
0
1
2
Theoretical Quantiles

Formal Test: Shapiro-Wilk Test

Tests whether data are normally distributed.

```r
shapiro.test(ms)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  ms
## W = 0.93428, p-value = 0.06386
```

Interpretation:

- **H_0**: Data are from a normal distribution
- **H_1**: Data are not normal
- If $p > 0.05 \rightarrow$ fail to reject $H_0 \rightarrow$ data are *normal enough*

What Does "Assume Normality" Mean?

It doesn't mean your data must be perfect.
It means:

- You've **checked assumptions**
- The data are **close enough**
- You can justify using parametric tests

When in doubt:

- Use **non-parametric** tests (next section)
- Be **transparent** in your thesis

Confidence Intervals and Statistical Inference

After describing our data and checking assumptions, we’re ready for **inference** — making conclusions about a population based on sample data.
The first tool of inference is the **confidence interval**.
What Is a Confidence Interval?
A **confidence interval (CI)** is a range of values likely to contain the **true population parameter** (like a mean).
For example:

> *“MergeSort is faster by 4.1 ms, 95% CI [1.6, 8.4]”*

This means we’re 95% confident the **true difference** is between 1.6 and 8.4 ms.

How to Interpret "95% Confidence"

It **does not mean** there's a 95% chance the true value lies in one specific interval.
It means:

> *If we repeated this experiment 100 times, 95 of those intervals would contain the true value.*

It's about the **method**, not one single outcome.

Confidence Interval Formula

The general formula for a CI is:

$$\mathrm{CI} = \bar{x} \pm t \cdot \mathrm{SE}$$

Where:

- $\bar{x}$: sample mean
- t: critical value from the t-distribution
- **SE**: standard error = $s/\sqrt{n}$

CI vs Standard Error

Standard Error (SE) shows **how much sample means vary**:

$$\mathrm{SE} = \frac{s}{\sqrt{n}}$$

The **CI** uses SE to build a range:

- SE = “spread of sample means”
- CI = “range of plausible population values”

Computing Confidence Intervals in R

We can use `t.test()` to compute the mean and its confidence interval:

```
qs_data <- algo_data$Time_ms[algo_data$Algorithm ==
         "QuickSort"]
ms_data <- algo_data$Time_ms[algo_data$Algorithm ==
         "MergeSort"]

t.test(qs_data)$conf.int
```

```
## [1] 49.59971 55.22333
## attr(,"conf.level")
## [1] 0.95
```

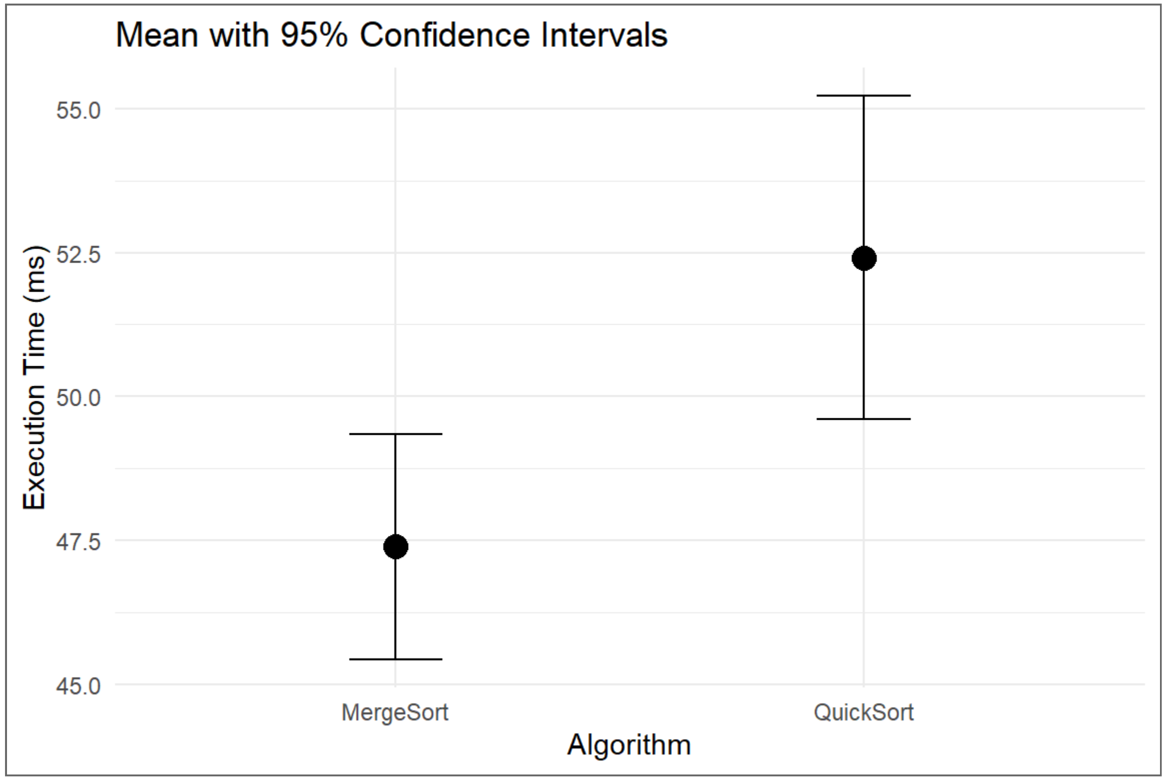
```
t.test(ms_data)$conf.int
```

```
## [1] 45.42997 49.35095
## attr(,"conf.level")
## [1] 0.95
```

This gives the **95% confidence interval** around the mean run-time.

Visualizing Confidence Intervals

We can plot mean ± CI error bars for each algorithm.



CI for the Difference in Means

What if we want to compare MergeSort and QuickSort directly?

```r
t.test(Time_ms ~ Algorithm, data = algo_data)$conf.int
```

```
## [1] -8.384496 -1.657631
## attr(,"conf.level")
## [1] 0.95
```

If the CI includes 0 → **no significant difference**
If it excludes 0 → **significant difference**

Example Interpretation

Suppose the CI is:

```
[-8.38, -1.66]
```

Interpretation: - We are 95% confident MergeSort is faster by **between 1.66 and 8.38 ms** - CI does **not include 0** → result is **statistically significant**

Summary: Confidence Intervals

Concept	Purpose
CI	Range of plausible population values
95% CI	95 out of 100 such intervals contain the true value
CI includes 0	No significant difference
CI excludes 0	Significant difference
Narrow CI	More precision (less variability)

Confidence intervals are a powerful tool for **quantifying uncertainty** — and they naturally lead us into **hypothesis testing**.

Foundations of Hypothesis Testing

Once we have summary statistics and confidence intervals, we’re ready for formal decision-making.
Hypothesis testing helps us answer questions like: > “Is MergeSort significantly faster than QuickSort?”
It is the standard method for drawing scientific conclusions from data.
What Is a Hypothesis?
A **hypothesis** is a testable statement about a population or process.
It must be:

- **Testable** — can be confirmed or refuted with data
- **Falsifiable** — you can imagine evidence against it
- **Specific** — clear about what is being measured

Types of Hypotheses

We always define **two hypotheses**:

- **Null Hypothesis (H_0)**
  The default assumption — no difference, no effect
  E.g., "The algorithms have equal mean run-time"
- **Alternative Hypothesis (H_1)**
  What we want to show — there is a difference or effect
  E.g., "MergeSort is faster than QuickSort"

We test **whether we can reject H_0** in favor of H_1.

What Makes a Good Hypothesis?

A strong hypothesis should: - Be based on theory or prior research
- Identify specific variables
- Be framed for statistical testing
- Use clear operational definitions

Examples:

Research Question	**H_0**	**H_1**
Is MergeSort faster?	$\mu_1 = \mu_2$	$\mu_1 < \mu_2$
Do students prefer UI A or B?	Equal preferences	At least one differs
Is crash rate linked to platform?	Independent	Dependent

The Logic of Hypothesis Testing

1. **Assume** H_0 is true
2. Collect data
3. Calculate how likely the data are **if H_0 were true**
4. If the data are **very unlikely**, reject H_0

   This logic uses the **p-value** to quantify that likelihood.

   What Is a p-value?

   The **p-value** is the probability of observing your results (or more extreme), **assuming H_0 is true.**

- Small p-value → data are unlikely under H_0
- Large p-value → data are plausible under H_0

> *It is **not** the probability that H_0 is true.*

Interpreting the p-value

p-value	Interpretation
≤ 0.05	Statistically significant — reject H_0
> 0.05	Not significant — fail to reject H_0

Significance level (α) is usually set at **0.05**

Type I and Type II Errors

Type	Description	Risk Symbol
Type I Error	Rejecting H_0 when it is actually true	α
Type II Error	Failing to reject H_0 when it is false	β

Reducing one type often increases the other.
We aim to **balance** them.

One-Tailed vs Two-Tailed Tests

- **Two-tailed**: Checks for **any** difference
  $H_1: \mu_1 \neq \mu_2$
- **One-tailed**: Checks for a **specific** direction
  $H_1: \mu_1 < \mu_2$
  Use **one-tailed only** when justified by theory — not after looking at data!

The Significance Level (α)

We set a threshold to control Type I error:

- Typically: **α = 0.05**
- Stricter for high-risk domains: α = 0.01

We compare:

- If **$p \leq \alpha$** → reject H_0 (significant)
- If **$p > \alpha$** → fail to reject H_0 (not significant)

Summary: Hypothesis Testing Flow

Step	**Description**
1. State H_0 and H_1	Define competing hypotheses
2. Choose α	Set risk tolerance (usually 0.05)
3. Compute p-value	Based on sample and test
4. Compare p to α	Decide whether to reject H_0
5. Interpret	Report significance and effect size

This process will now guide us into **specific tests** — starting with the **t-test** for comparing means.

Comparing Means: t-Tests

When your outcome is **numeric** and you want to compare **means**, the most common tool is the **t-test**.

There are three main types:

- One-sample t-test
- Independent two-sample t-test
- Paired-sample t-test

We'll explore each with assumptions, examples, and interpretation.

One-Sample t-Test

Use when: You compare a sample mean to a known value.

Is MergeSort's mean run-time different from 50ms?

```
ms_data <- algo_data$Time_ms[algo_data$Algorithm ==
         "MergeSort"]
t.test(ms_data, mu = 50)
```

```
##
##  One Sample t-test
##
## data:  ms_data
## t = -2.7223, df = 29, p-value = 0.01085
## alternative hypothesis: true mean is not equal to 50
## 95 percent confidence interval:
##  45.42997 49.35095
## sample estimates:
```

```
## mean of x
##  47.39046
```

- H_0: $\mu = 50$
- H_1: $\mu \neq 50$
  Reject H_0 if the p-value is less than 0.05.

Independent Two-Sample t-Test

Use when: You compare two independent groups.

> *Is there a significant difference in mean run-time between MergeSort and QuickSort?*

```r
t.test(Time_ms ~ Algorithm, data = algo_data, var.equal
        = FALSE)
```

```
##
##  Welch Two Sample t-test
##
## data:  Time_ms by Algorithm
## t = -2.9959, df = 51.806, p-value = 0.004192
## alternative hypothesis: true difference in means
between group MergeSort and group QuickSort is not equal
to 0
## 95 percent confidence interval:
```

```
##  -8.384496 -1.657631
## sample estimates:
## mean in group MergeSort mean in group QuickSort
##                47.39046                52.41152
```

- H_0: $\mu_1 = \mu_2$
- H_1: $\mu_1 \neq \mu_2$
  Welch's t-test (`var.equal = FALSE`) handles unequal variances.

Paired-Sample t-Test

Use when: You compare two values from the **same subject** (e.g., two algorithms on the same input).

Simulated Paired Data

```r
set.seed(123)
paired_data <- data.frame(
  Input = 1:30,
  QuickSort = rnorm(30, 52, 6),
  MergeSort = rnorm(30, 48, 5)
)

t.test(paired_data$QuickSort, paired_data$MergeSort,
         paired = TRUE)
```

```
##
##  Paired t-test
##
## data:  paired_data$QuickSort and paired_data$MergeSort
## t = 2.001, df = 29, p-value = 0.05483
## alternative hypothesis: true mean difference is not
equal to 0
## 95 percent confidence interval:
##  -0.06243577  5.71380733
## sample estimates:
## mean difference
##        2.825686
```

- H_0: Median difference = 0
- H_1: Median difference ≠ 0

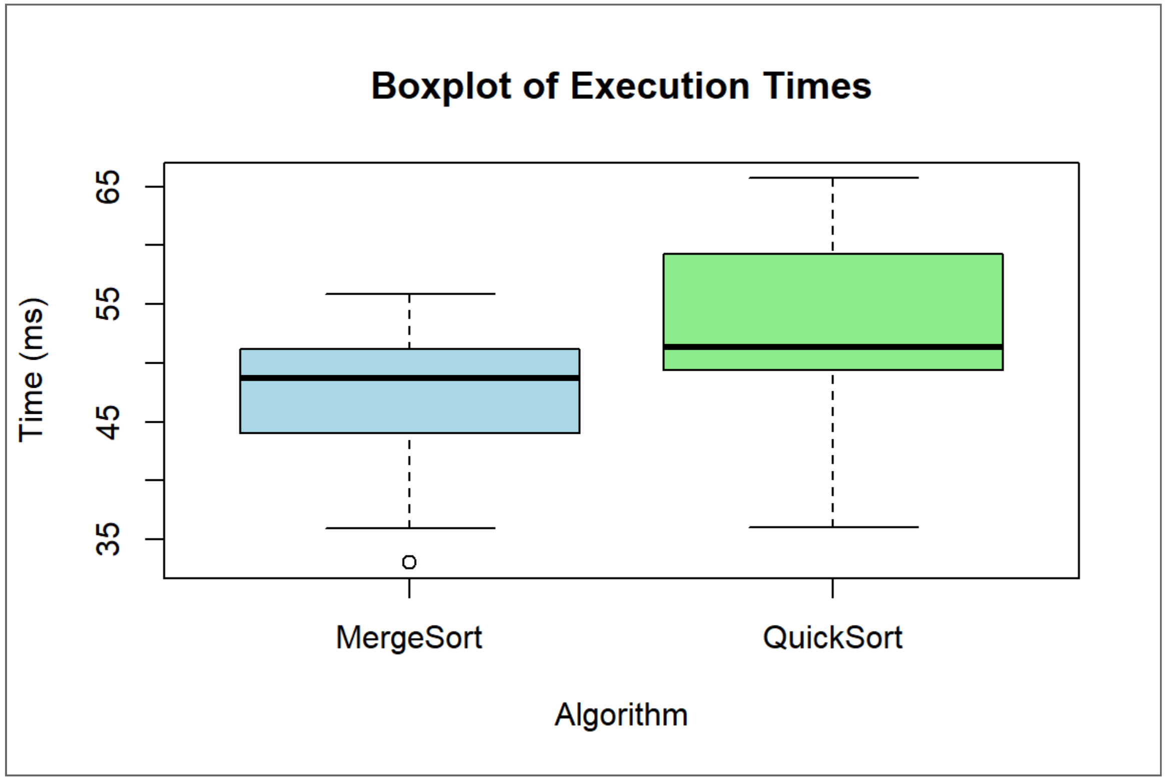
Assumptions of the t-Test

Assumption	Description
Normality	Each group’s data ~ normal distribution
Equal variances	Use Welch’s test if not true
Independence	Observations are not repeated
Interval-level data	Values must be numeric and continuous

Violations of these assumptions lead to **non-parametric tests** (we’ll cover these later).

Visualizing Group Differences

```r
boxplot(Time_ms ~ Algorithm, data = algo_data,
        col = c("lightblue", "lightgreen"),
        main = "Boxplot of Execution Times",
        ylab = "Time (ms)")
```



Use boxplots and CI bars to complement test results.

Summary: Which t-Test?

Test	Use Case	Function
One-sample t-test	Sample vs fixed value	`t.test(x, mu = val)`
Two-sample t-test	Two independent groups	`t.test(y ~ group)`
Paired-sample t-test	Two related measurements	`t.test(x, y, paired = TRUE)`

Comparing 3+ Groups: ANOVA

When comparing **more than two groups**, we use **ANOVA** instead of multiple t-tests to avoid inflating the Type I error rate.
ANOVA tells us:

> *“Is there* ***any*** *significant difference between group means?”*

We’ll explore:

- One-way ANOVA
- Two-way ANOVA
- Post-hoc tests (Tukey HSD)
- Assumptions and visual inspection

One-Way ANOVA: Setup

Let’s simulate a third algorithm: HeapSort

```r
set.seed(2025)
qs <- rnorm(30, mean = 52, sd = 6)
ms <- rnorm(30, mean = 48, sd = 5)
hs <- rnorm(30, mean = 50, sd = 4.5)

algo3_data <- data.frame(
  Algorithm = rep(c("QuickSort", "MergeSort",
                "HeapSort"), each = 30),
  Time_ms = c(qs, ms, hs)
)
```

Running One-Way ANOVA

```
anova_model <- aov(Time_ms ~ Algorithm, data =
         algo3_data)
summary(anova_model)
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## Algorithm    2  408.2  204.12   7.499 0.00099 ***
## Residuals   87 2368.2   27.22
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'
0.1 ' ' 1
```

- H_0: $\mu_1 = \mu_2 = \mu_3$ (all means are equal)
- H_1: At least one group differs

  If $p < 0.05$ → Reject H_0 → At least one mean is significantly different

Post-Hoc Testing: Tukey HSD

ANOVA tells us **there is a difference**, but not **which groups differ**.
Use **Tukey's Honest Significant Difference (HSD)** to compare all pairs:

```
TukeyHSD(anova_model)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Time_ms ~ Algorithm, data =
algo3_data)
##
## $Algorithm
##                           diff        lwr      upr
p adj
## MergeSort-HeapSort  -0.9764591 -4.1886528 2.235735
0.7494717
## QuickSort-HeapSort   3.9499434  0.7377497 7.162137
```

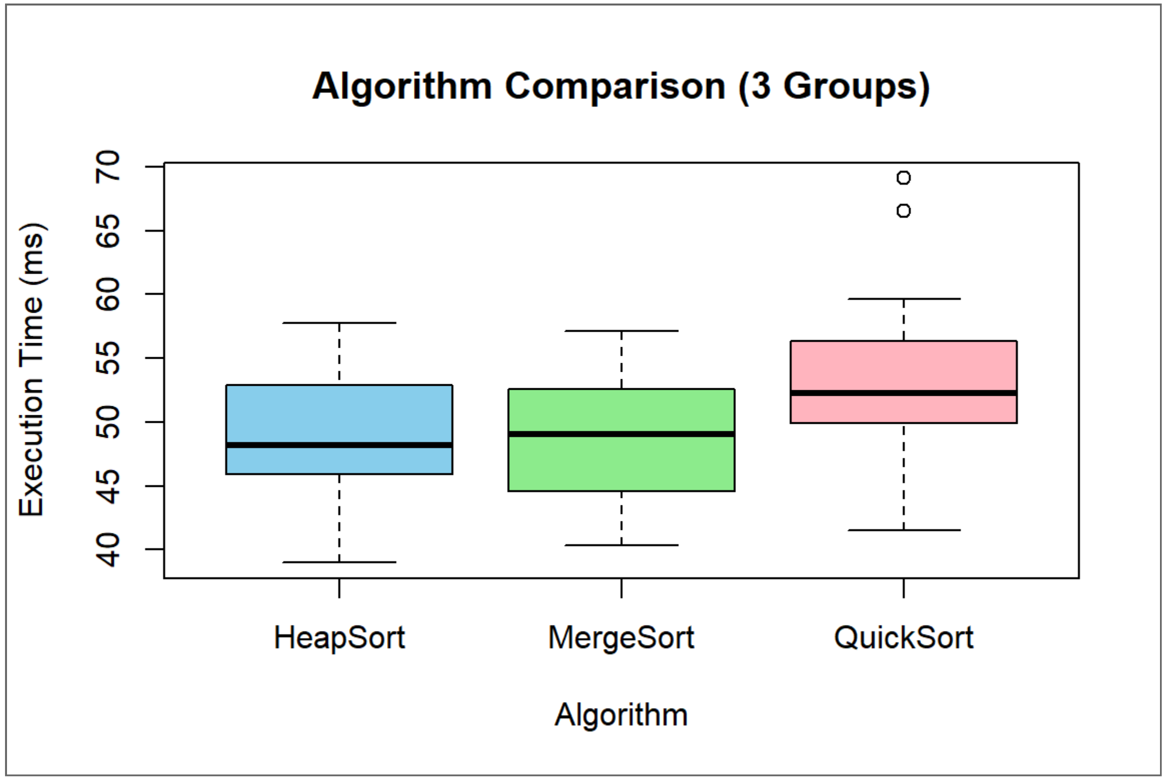
```
0.0118579
```

Look for:

- Which comparisons are significant
- Confidence intervals that do **not** include 0

Visualizing Group Differences

```r
boxplot(Time_ms ~ Algorithm, data = algo3_data,
        col = c("skyblue", "lightgreen", "lightpink"),
        ylab = "Execution Time (ms)",
        main = "Algorithm Comparison (3 Groups)")
```


Algorithm Comparison (3 Groups)
Execution Time (ms)
40
45
50
55
60
65
70
HeapSort
MergeSort
QuickSort
Algorithm

Assumptions of ANOVA

Assumption	How to Check
Independence	Experimental design
Normality	Use QQ plot or Shapiro test per group
Equal variances	Use Bartlett or Levene test

Example: Test for Equal Variances

```r
bartlett.test(Time_ms ~ Algorithm, data = algo3_data)
```

```
##
##  Bartlett test of homogeneity of variances
##
```

```
## data:  Time_ms by Algorithm
## Bartlett's K-squared = 1.3655, df = 2, p-value = 0.5052
```

Two-Way ANOVA: Including More Factors

Let’s add **Input Size** as a second factor: Small vs Large datasets.

```r
set.seed(123)
algo <- rep(c("QuickSort", "MergeSort", "HeapSort"),
         times = 30)
size <- rep(rep(c("Small", "Large"), each = 15), times
         = 3)

runtime <- rnorm(90,
                 mean = ifelse(algo == "MergeSort",
         48,
                      ifelse(algo == "QuickSort",
         52, 50)) +
                     ifelse(size == "Large", 3, 0),
                 sd = 4.5)
```

Two-Way ANOVA Model

```
two_way_model <- aov(Time_ms ~ Algorithm * InputSize,
          data = two_way_data)
summary(two_way_model)
```

```
##                     Df Sum Sq Mean Sq F value
Pr(>F)
## Algorithm            2  477.3  238.67  14.485 3.94e-
06 ***
## InputSize            1  142.4  142.44   8.645
0.00424 **
## Algorithm:InputSize  2    4.9    2.45   0.149
0.86182
## Residuals           84 1384.1   16.48
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'
0.1 ' ' 1
```

This tests:

- Main effect of **Algorithm**
- Main effect of **Input Size**
- **Interaction** between them

A significant interaction means:

> *The effect of Algorithm **depends on Input Size***

Summary: ANOVA

Test	Use Case	Tool
One-way ANOVA	3+ group means	`aov(y ~ group)`
Tukey HSD	Post-hoc pairwise	`TukeyHSD()`
Bartlett Test	Equal variances	`bartlett.test()`
Two-way ANOVA	2 factors	`aov(y ~ A * B)`

Chi-Squared Tests for Categorical Data

When your data are **categorical** (e.g., success/fail, error type, UI preference), you need a test designed for **counts**, not means.
The **Chi-squared (χ^2) test** helps answer questions like:

- Do observed results match expectations?
- Are two categorical variables related?

Two Main Types of Chi-Squared Tests

Test	Purpose
Goodness-of-Fit	Do observed counts match expected proportions?
Test of Independence	Are two categorical variables related?

We'll explore both with examples.

Chi-Squared Goodness-of-Fit

Use when: You compare observed outcomes to an expected distribution.

Example: HeapSort ran 60 times
You expect 95% Success, 5% Fail.
You observed: 54 Success, 6 Fail

```r
observed <- c(Success = 54, Fail = 6)
expected_probs <- c(0.95, 0.05)
chisq.test(x = observed, p = expected_probs)
```

```
## Warning in chisq.test(x = observed, p =
expected_probs): Chi-squared
## approximation may be incorrect
```

```
##
##  Chi-squared test for given probabilities
```

```
##
## data:  observed
## X-squared = 3.1579, df = 1, p-value = 0.07556
```

Interpreting Goodness-of-Fit

- H_0: The observed distribution matches the expected
- H_1: The distributions differ
- If $p < 0.05 \rightarrow$ significant $\rightarrow$ reject H_0
  This helps check whether results conform to a known or ideal pattern.

Chi-Squared Test of Independence

Use when: You want to know if **two categorical variables are associated**.

Example: Is success rate dependent on the algorithm used?

Simulated Example (Success/Failure by Algorithm)

```
## Warning in chisq.test(tab): Chi-squared approximation
may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab
## X-squared = 9.6041, df = 2, p-value = 0.008213
```

Interpreting the Test of Independence

- H_0: The variables are independent (algorithm does not affect outcome)
- H_1: The variables are dependent (algorithm affects outcome)
  If **$p < 0.05$**, we reject H_0 and conclude that the outcome **depends** on the algorithm.

Assumptions of the Chi-Squared Test

Assumption	**How to Handle**
Independent observations	No repeated measurements
Expected count ≥ 5 in each cell	Use Fisher’s Exact Test if violated
Sufficient total sample size	Larger n = better approximation

When Expected Counts Are Too Small

If any expected cell count is **less than 5**, consider:

- Combining similar categories
- Using **Fisher's Exact Test** (especially for 2x2 tables)

```
fisher.test(tab)  # if dimensions are small
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  tab
## p-value = 0.01512
## alternative hypothesis: two.sided
```

Summary: Chi-Squared Tests

Test Type	Use Case	R Function
Goodness-of-fit	One variable vs expected proportions	`chisq.test(x, p)`
Test of independence	Two categorical variables	`chisq.test(table)`
Small samples	Any 2x2 table	`fisher.test()`

These tests are essential when your research deals with:

- User preferences
- Pass/fail outcomes
- Classification results
- Any system behavior encoded as **categories**

Correlation Analysis

When working with **two numeric variables**, we often ask:

> *“As one variable increases, does the other also increase (or decrease)?”*

Correlation analysis helps answer this by measuring:

- The **strength** of the relationship
- The **direction** (positive or negative)

Types of Correlation

Method	**Use When**
Pearson	Linear relationship, numeric data, normality
Spearman	Ordinal data or non-linear monotonic relationship
Kendall (optional)	Small samples, tied ranks

We’ll focus on Pearson and Spearman.

Pearson Correlation Coefficient (r)

Used for:

- Two continuous, normally distributed variables
- Linear relationships

Values range from:

- +1: perfect positive correlation
- 0: no correlation
- −1: perfect negative correlation

```r
set.seed(1)
input_size <- seq(100, 1000, length.out = 50)
qs_runtime <- 0.05 * input_size + rnorm(50, mean = 0, sd
         = 10)

cor(input_size, qs_runtime)  # Pearson correlation
```

```
## [1] 0.8478924
```

Spearman Rank Correlation

Used for:

- Ordinal or **non-linear monotonic** relationships
- Data with **outliers** or **non-normality**

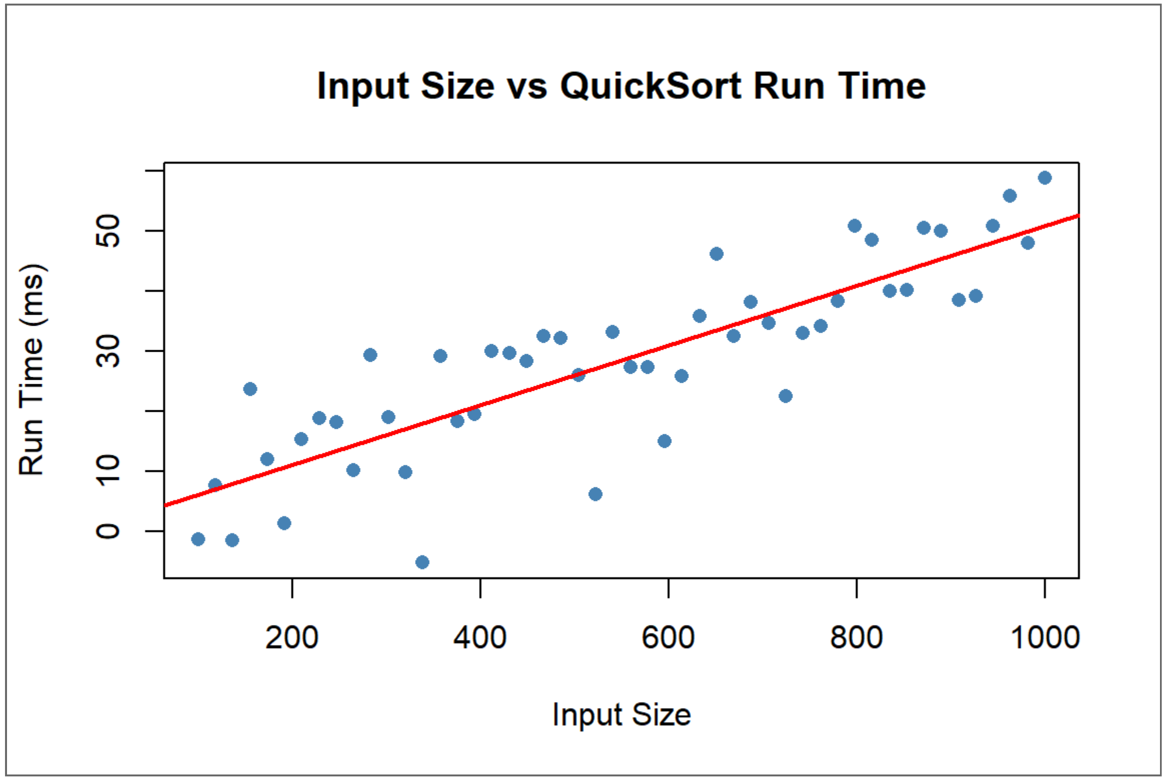
```
cor(input_size, qs_runtime, method = "spearman")
```

```
## [1] 0.8690036
```

Compares **ranks** instead of raw values.

Scatterplot: Input Size vs Run Time

Always visualize your correlations — outliers and non-linear trends are not visible from `cor()` alone.



Correlation Matrix (Multiple Variables)

You can quickly assess relationships between several numeric variables using a correlation matrix:

```
df <- data.frame(
  Size = input_size,
  QuickSort = qs_runtime,
  Noise = rnorm(50),
  Quadratic = input_size^2 + rnorm(50, sd = 100)
)

round(cor(df), 2)
```

```
##           Size QuickSort Noise Quadratic
## Size      1.00      0.85 -0.08      0.98
## QuickSort 0.85      1.00 -0.09      0.84
## Noise     -0.08      -0.09  1.00      -0.07
## Quadratic  0.98       0.84 -0.07       1.00
```

Use this to explore pairwise relationships at a glance.

Correlation ≠ Causation

Just because two variables move together does **not** mean one causes the other.

Correlation may result from:

- *Coincidence*
- *Confounding (a third variable affects both)*
- *Reverse causality*

Always interpret correlations with **context and caution**.

Summary: Correlation Analysis

Concept	**Pearson**	**Spearman**
Measures	Linear association	Monotonic association
Sensitive to	Outliers, skew	Less sensitive
Use with	Numeric, normal data	Ranked or skewed data
Output	-1 to +1 (strength & direction)	Same scale, different method

Correlation is a **powerful exploratory tool**, but should be followed by **modeling or experimental design** when aiming for causal claims.

Non-Parametric Alternatives

Sometimes, your data:

- Isn’t **normally distributed**
- Has **small sample sizes**
- Includes **outliers** or **ranks** instead of true numeric values

In these cases, we turn to **non-parametric tests.**

> *These tests don’t rely on distributional assumptions and operate on **ranks** instead of raw data.*

When to Use Non-Parametric Tests

Use them when:

- Normality is violated
- You’re working with **ordinal** (ranked) data
- Your sample size is too small for the Central Limit Theorem to apply
- Outliers distort the mean

Wilcoxon Signed-Rank Test

Alternative to: **Paired t-test**

Use when: - You have two **related** groups

- The **differences are not normally distributed**

```r
qs <- rnorm(30, mean = 52, sd = 7)
ms <- rnorm(30, mean = 48, sd = 7)

wilcox.test(qs, ms, paired = TRUE)
```

```
##
##  Wilcoxon signed rank exact test
##
## data:  qs and ms
## V = 358, p-value = 0.008705
## alternative hypothesis: true location shift is not
equal to 0
```

- H_0: Median difference = 0
- H_1: Median difference ≠ 0

Mann–Whitney U Test (Wilcoxon Rank-Sum)

Alternative to: **Independent two-sample t-test**

Use when:

- Comparing **two independent groups**
- The data are **skewed** or ordinal

```
group <- rep(c("QuickSort", "MergeSort"), each = 30)
times <- c(qs, ms)

wilcox.test(times ~ group)
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  times by group
## W = 283, p-value = 0.01307
## alternative hypothesis: true location shift is not
equal to 0
```

- Compares whether one group tends to rank higher than the other

Kruskal–Wallis Test

Alternative to: **One-way ANOVA**

Use when: - You want to compare **3 or more groups**
- Data is not normally distributed

```
group <- rep(c("QuickSort", "MergeSort", "HeapSort"),
             each = 30)
times <- c(
  rnorm(30, mean = 52, sd = 6),
  rnorm(30, mean = 48, sd = 6),
  rnorm(30, mean = 50, sd = 6)
)

kruskal.test(times ~ group)
```

```
##
##  Kruskal-Wallis rank sum test
##
```

```
## data:  times by group
## Kruskal-Wallis chi-squared = 3.4113, df = 2, p-value
= 0.1817
```

If significant → follow up with pairwise Wilcoxon tests.

Summary: Parametric vs Non-Parametric

Goal	**Parametric Test**	**Non-Parametric Alternative**
Compare one sample to a value	One-sample t-test	(rarely needed)
Compare two independent groups	Two-sample t-test	Mann–Whitney U
Compare two paired groups	Paired t-test	Wilcoxon signed-rank
Compare 3+ groups	One-way ANOVA	Kruskal–Wallis

When to Prefer Non-Parametric Tests

Choose non-parametric tests if:

- The **normality assumption fails**
- You’re analyzing **ordinal/ranked data**
- You want a method **robust to outliers**

They’re less powerful than parametric tests **when assumptions are met**, but much more **reliable** when assumptions are **violated**.

Choosing the Right Statistical Test

Goal	Data	Groups	Assumptions Met?	Test
Compare to known value	Numeric	1	✓	One-sample t-test
Compare two groups	Numeric	2 (independent)	✓	Two-sample t-test
Compare two paired sets	Numeric	2 (paired)	✓	Paired t-test
Compare 3+ groups	Numeric	3+	✓	ANOVA
Compare ranks/medians	Ordinal / skewed	2	✗	Mann–Whitney U / Wilcoxon

Goal	Data	Groups	Assumptions Met?	Test
Compare 3+ medians	Ordinal / skewed	3+	✗	Kruskal–Wallis
Compare counts to expected	Categorical	1 variable	–	Chi-squared (Goodness-of-Fit)
Association between categories	Categorical	2 variables	–	Chi-squared (Independence)
Correlation (linear)	Numeric	2 vars	✓	Pearson
Correlation (ranked)	Ordinal / skewed	2 vars	✗	Spearman