Statistics for MSc

Dr Polla Fattah

2025-04-22

Statistical Analysis

1. Introduction: Why Statistics in Research?

1.1 The Role of Statistics in Computer Science Research

In computer science, especially at the postgraduate level, **research goes beyond building systems**. It must involve **evaluating, validating, and communicating** those systems' effectiveness using data.

This is where **statistics becomes essential**.

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, or is it meaningful?*
- *How do I report my findings objectively?*

Statistics provides the **language of evidence** in science. It transforms raw performance numbers into **scientifically defensible claims**.

1.2 Why Postgraduates Can't Ignore Statistics

In computer science research, especially at the postgraduate level, **an observation alone is not evidence**. Without statistical analysis, a claim such as:

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

is anecdotal at best. It lacks the rigor required for scientific validation.

In a thesis defense, such a statement will immediately prompt scrutiny. Examiners are trained to ask questions like:

- How many times did you test each algorithm?
- What was the average execution time?
- How consistent were the results?
- Could the difference be explained by random variation?
- Did you use different datasets or input sizes?
- Have you demonstrated that the difference is statistically significant?

These are not simply technical questions — they are **statistical questions**. Without a solid understanding of statistical tools, it is difficult to answer them convincingly.

In short, **statistics is what transforms a set of observations into a defensible research finding**. It allows you to quantify uncertainty, test assumptions, justify claims, and communicate results clearly — all of which are essential for academic credibility.

1.3 Statistics in Experimental Design and Validation

In computer science research, statistical methods are used for:

Research Activity	Statistical Role
Algorithm performance comparison	t-tests, ANOVA, boxplots
Classifier accuracy evaluation	Confidence intervals, hypothesis testing
Survey analysis	Chi-squared tests, proportions, ranking
Input/output performance modeling	Regression, correlation
Software/system reliability	Error rate analysis, standard deviation

1.4 Our Guiding Scenario: The Algorithm Showdown

To make the statistical concepts in this lecture more tangible and relevant to your experience as a postgraduate student, we will return repeatedly to a **single research scenario** — one that resembles the kind of problem you might actually face in your thesis.

The Scenario

> **You are defending your MSc thesis in Computer Science.**
> Your research investigates the performance of two well-known sorting algorithms: **MergeSort** and **QuickSort**.

You claim that **MergeSort is more efficient**, and you want to prove this claim **based on experimental data**.

The Experiment

To support your hypothesis, you design a controlled experiment:

- You run each algorithm **30 times** on large, randomly generated datasets.
- You record the **execution time in milliseconds** for every trial.
- You maintain consistent conditions across runs to ensure fairness.

This results in a dataset containing **120 numerical measurements** — 60 from each algorithm — which will form the basis for all the statistical techniques explored in this course.

2.1 Introducing the Hypothetical Experiment

To demonstrate statistical concepts in this lecture, we will simulate a **controlled experiment** involving two sorting algorithms: **MergeSort** and **QuickSort**.

You are a postgraduate student evaluating the performance of these algorithms for your thesis. You want to answer the research question:

> *Is MergeSort significantly faster than QuickSort on average?*

To explore this, you conduct the following experiment:

- You run **MergeSort** and **QuickSort** on the same large dataset.
- Each algorithm is executed **30 times**.
- You record the **execution time in milliseconds** for each run.

These measurements will form the basis of our statistical analysis throughout the lecture.

We use the a random function to simulate realistic, normally distributed data with slight differences in mean and variability.

Run	QuickSort (ms)	MergeSort (ms)
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
10	51.62	48.18
11	59.83	49.03
12	65.72	46.19
13	43.67	51.79
14	50.33	44.37
15	51.20	41.16
16	55.82	50.16
17	50.29	43.94
18	36.06	55.22
19	37.36	45.84
20	59.92	51.28

Note on Data Format: While the table above shows the data in a **wide format** (QuickSort and MergeSort side-by-side) for clarity, our actual dataset is stored in **long format**.

This allows us to easily calculate group-wise statistics, create plots, and run statistical tests using tidy R tools such as `t.test()` , `aggregate()` , and `ggplot2` , that we will learn about them later in this lecture. So the data is actaully in this format

```
print(head(algo_data))
```

```
##   Algorithm  Time_ms
## 1 QuickSort 60.22575
## 2 QuickSort 48.61181
## 3 QuickSort 54.17877
## 4 QuickSort 55.79718
## 5 QuickSort 54.42561
## 6 QuickSort 51.36325
```

This dataset contains: - A column `Algorithm` indicating which sorting algorithm was used
- A column `Time_ms` representing the measured execution time for that run

You now have a clean and realistic dataset with which to explore: - Measures of central tendency and spread
- Visualizations
- Assumption checking
- Confidence intervals
- Hypothesis testing

The Core Research Question

The central question you aim to answer is:

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

To answer this convincingly, you cannot rely on visual inspection or gut feeling. You must:

- Summarize the data with appropriate statistics
- Visualize distributions and check for skewness or outliers
- Verify statistical assumptions
- Calculate confidence intervals
- Perform hypothesis testing
- Communicate your findings in a clear, objective, and statistically sound manner

In other words, your thesis defense becomes a **statistical investigation** — one in which every concept we learn will help you build a scientifically credible argument.

1.5 What You Will Learn in This Lecture

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

- **Describe and interpret key statistical concepts**, including mean, median, variance, and standard deviation
- **Summarize and visualize data** using appropriate tools such as histograms, boxplots, and scatterplots
- **Assess the distribution of data**, including symmetry, skewness, and normality assumptions using QQ plots and statistical tests
- **Quantify uncertainty** through the construction and interpretation of confidence intervals
- **Perform and interpret statistical comparisons**, including t-tests, ANOVA, and their non-parametric alternatives
- **Analyze categorical variables** using chi-squared tests for goodness-of-fit and independence
- **Understand and apply hypothesis testing**, including p-values, significance levels, and effect size
- **Present and defend statistical findings** clearly and rigorously in the context of a research thesis

2. Summarizing Data: Center and Spread

2.1 Central Tendency

When analyzing data, one of the first questions we often ask is:

> "Where is the center of the data?"

Central tendency refers to the concept of a typical or central value around which data tend to cluster. It is a foundational idea in statistics, providing a **numerical summary of an entire dataset** using a single value.

The three most common measures of central tendency are:

- **Mean**: the arithmetic average
- **Median**: the middle value
- **Mode**: the most frequently occurring value

Each has strengths and weaknesses depending on the **shape** and **nature** of the data.

The Mean (Arithmetic Average)

The **mean** is the sum of all values divided by the number of values:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

It is widely used due to its simplicity and because many statistical techniques are based on it. However, the mean is **sensitive to outliers**. A single extreme value can distort the result significantly.

Example: If most run-times are around 50ms, but one test took 200ms due to a system hiccup, the mean will be pulled upward.

The Median (Middle Value)

The **median** is the middle value when all observations are ordered. If the number of observations is even, it is the average of the two middle values.

It is **not affected by outliers**, which makes it a more robust indicator of central tendency in **skewed distributions**.

Example: In the presence of an unusually long run-time in a generally stable set of results, the median may reflect the "typical" behavior more faithfully than the mean.

The Mode (Most Frequent Value)

The **mode** is the value that appears most frequently in a dataset.

- In continuous numerical data, the mode is often not informative or may not exist.
- In categorical or ordinal data (e.g., UI preference: A, B, or C), the mode becomes much more useful.

Calculating Central Tendency

Let’s use our experimental dataset comparing QuickSort and MergeSort execution times.

```
mean(algo_data$Time_ms)      # Overall mean
```

```
## [1] 49.90099
```

```
median(algo_data$Time_ms)    # Overall median
```

```
## [1] 50.28587
```

```
# Mode is not defined directly for continuous data, but we can approximate:
table(round(algo_data$Time_ms))  # Frequency table of rounded values
```

```
##
## 33 36 37 39 41 44 45 46 48 49 50 51 52 53 54 55 56 59 60 61 63 64 66
##  1  2  1  1  3  6  1  2  4  5  7  8  3  1  2  2  3  1  3  1  1  1  1
```

Interpretation in a Research Context

When preparing your thesis: - Use the **mean** if your data are **normally distributed** and free of outliers. - Use the **median** if your data are **skewed** or contain outliers. - Use the **mode** only for **categorical or ordinal variables**.

Try to pair numerical summaries with **visualizations** to get a fuller picture.

2.2 Understanding Variability

While central tendency tells us *where* the data are centered, it does not tell us *how spread out* the values are around that center. In research, especially in performance analysis, knowing whether values are **consistently close to the mean** or **widely scattered** is crucial.

Variability measures help us quantify **how much uncertainty or inconsistency** exists in our data.

In this section, we will introduce four core concepts:

- **Range**
- **Deviation from the mean**
- **Variance**
- **Standard deviation**

Each concept builds upon the previous, leading to a deeper understanding of the data’s behavior.

2.2.1 Range

The **range** is the simplest measure of spread:

$$\text{Range} = \text{Maximum} - \text{Minimum}$$

It gives the full span of values but is **very sensitive to outliers**, since it depends only on the two extreme points.

```
range(algo_data$Time_ms)
```

```
## [1] 33.03455 65.71987
```

```
diff(range(algo_data$Time_ms))  # Actual range value
```

```
## [1] 32.68532
```

This tells us the **total spread** of execution times across both algorithms. However, it does not tell us how values are distributed within this range.

2.2.2 Deviations from the Mean

To get a deeper sense of spread, we consider how far each observation is from the mean:

$$x_i - \bar{x}$$

Where: - x_i is each value - $\bar{x}$ is the sample mean

These are called **deviations**. They reflect the distance between each value and the center of the distribution.

However, if we add all the deviations, they **sum to zero** due to symmetry:

```
mean(algo_data$Time_ms)
```

```
## [1] 49.90099
```

```
sum(algo_data$Time_ms - mean(algo_data$Time_ms))  # Always zero
```

```
## [1] -6.394885e-14
```

This leads us to **square the deviations** to avoid cancellation — and that gives us the **variance**.

2.2.3 Variance (s²)

Variance is the average of the squared deviations from the mean:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

- The numerator captures the **total squared deviation**
- Dividing by $n - 1$ (not n) corrects for bias — this is known as **Bessel's correction**

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

```
## MergeSort QuickSort
##  27.56546  56.70344
```

Interpretation:
Variance gives us a numerical sense of spread, but its unit is **squared**, making it harder to interpret directly (e.g., ms²).

2.2.4 Standard Deviation (SD)

The **standard deviation** is simply the **square root of the variance**:

$$s = \sqrt{s^2}$$

It expresses variability in the **same units as the original data**, making it far more interpretable.

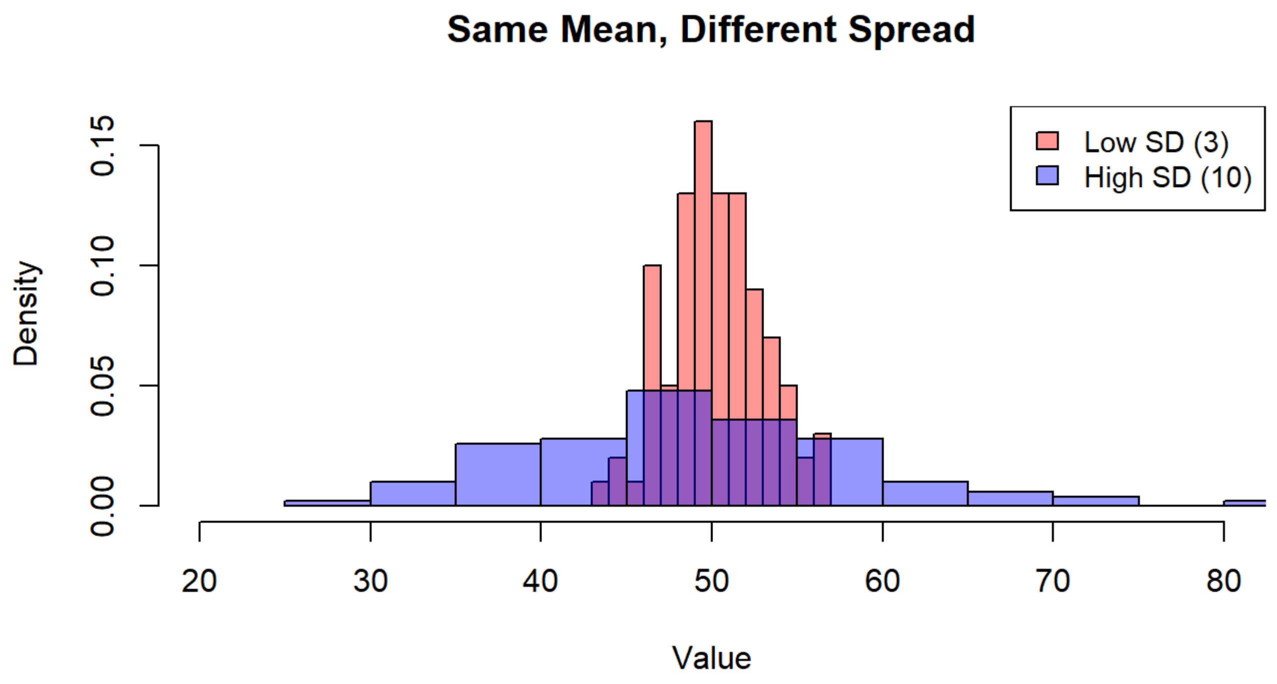
```
tapply(algo_data$Time_ms, algo_data$Algorithm, sd)
```

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

Interpretation:
The standard deviation tells us, on average, **how far values deviate from the mean**. A higher SD indicates greater variability (i.e., less consistency), while a lower SD implies tighter clustering.

2.2.5 Visual Intuition for Spread

Let's visually compare two distributions that have **the same mean**, but **different spreads**.



This illustrates: - **Higher SD** leads to a **wider, flatter** distribution - **Lower SD** results in a **narrower, more peaked** shape

Even with the **same average**, variability changes how reliable and predictable the data are.

2.3 Visualizing Distributions

While numerical summaries (like mean and standard deviation) are useful, they can sometimes hide important details about how data behave. **Visualizing the distribution** of your data allows you to:

- Detect **patterns**, **skewness**, or **outliers**
- Evaluate assumptions like **symmetry** and **normality**
- Compare distributions across groups
- Communicate findings more clearly to others

In this section, we will explore the two most important tools for visualizing distributions: - **Histograms** - **Boxplots**

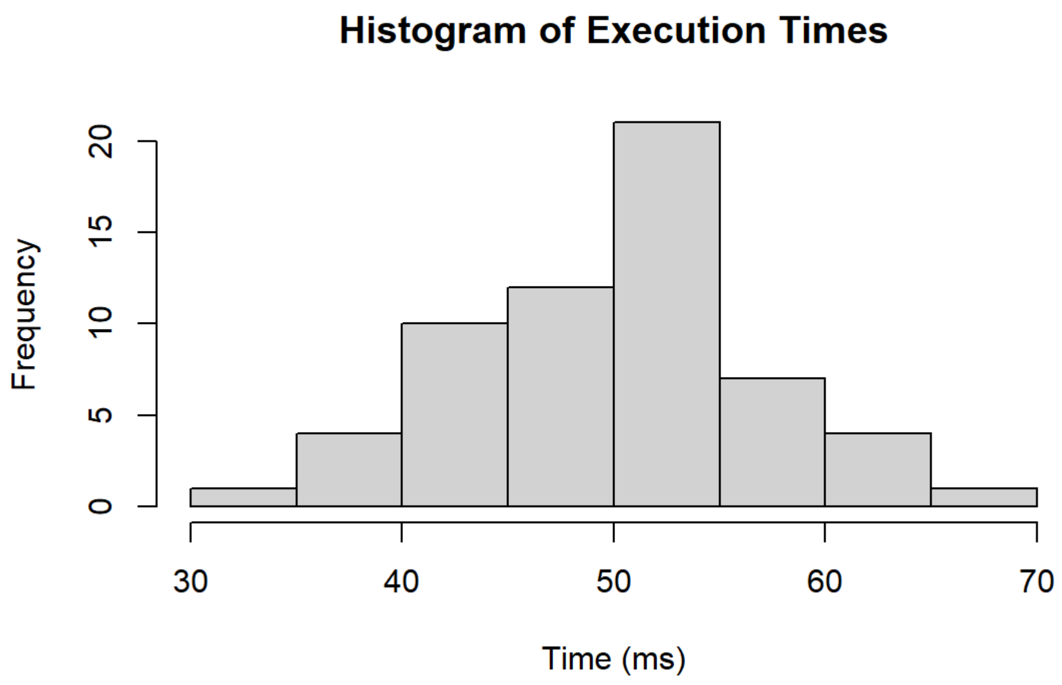
We will also introduce supporting concepts: **quartiles**, **interquartile range (IQR)**, and **outlier detection**.

2.3.1 Histograms

A **histogram** shows how data are distributed across intervals (called *bins*). It answers the question:

> "How many values fall within each range of the variable?"

Each bar represents a range of values, and its height indicates how many observations fall within that range.



Why Use a Histogram?

- To see **how values are spread**
- To detect **skewness** or unusual peaks
- To explore whether the data look **normally distributed**
- To get a rough sense of **center and spread**

Limitation: Histograms don't handle group comparisons well — for that, we need boxplots.

2.3.2 Understanding Quartiles and IQR

Before we explore boxplots, we must understand **quartiles**, which divide a dataset into four equal parts:

- **Q1 (25th percentile)**: 25% of data falls below this value
- **Q2 (50th percentile)**: the **median** of the dataset
- **Q3 (75th percentile)**: 75% of data falls below this value

The **interquartile range (IQR)** is defined as:

$$\text{IQR} = Q_3 - Q_1$$

It measures the **middle 50% of the data**, excluding extremes. It's a robust alternative to standard deviation when data are skewed or contain outliers.

You can compute quartiles and IQR in R using:

```
summary(algo_data$Time_ms)  # includes Q1, Median, Q3
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   33.03   45.62   50.29   49.90   53.43   65.72
```

```
IQR(algo_data$Time_ms)
```

```
## [1] 7.805412
```

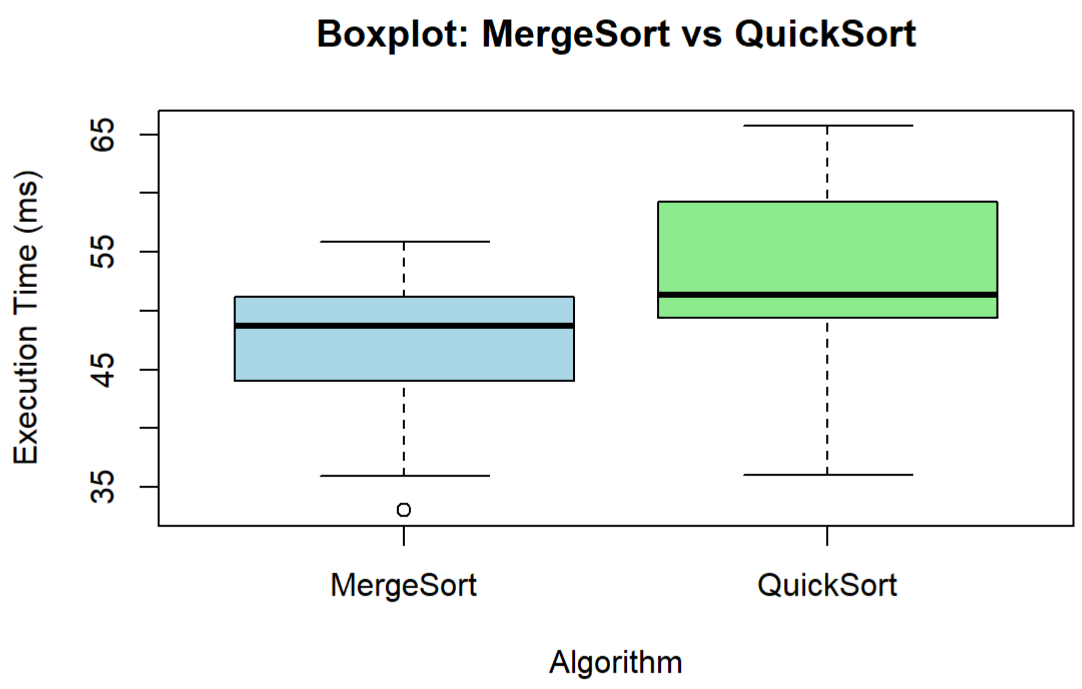
2.3.3 Boxplots: A Compact Summary of Distribution

A **boxplot** (or box-and-whisker plot) is a compact visual summary of the distribution of a variable. It is built on the five-number summary:

- **Minimum** (excluding outliers)
- **Q1** (25th percentile)
- **Median** (Q2, 50th percentile)
- **Q3** (75th percentile)
- **Maximum** (excluding outliers)

Additionally, boxplots **mark outliers** as individual points if they fall too far from the central range.

```
boxplot(Time_ms ~ Algorithm,
        data = algo_data,
        col = c("lightblue", "lightgreen"),
        main = "Boxplot: MergeSort vs QuickSort",
        ylab = "Execution Time (ms)")
```



How to Interpret a Boxplot

- The **box** spans from **Q1 to Q3** — this is the IQR.
- The **line inside the box** shows the **median**.
- The **"whiskers"** extend to the minimum and maximum values **within 1.5 × IQR** from the box.
- **Points outside that range** are considered **outliers** and plotted individually.

Example: What the Boxplot Tells You

From the boxplot above, you can immediately tell:

- Which algorithm has a **lower median time**
- Which one has **more variability** (based on IQR and whisker length)
- Whether either algorithm has **outliers**
- Whether distributions are **symmetric or skewed**

For instance, a **longer upper whisker** or a **cluster of outliers on one side** may indicate **right skewness**.

2.3.4 Outliers and Skewness

Outliers are values that lie **far from the rest of the data**. In boxplots, a value is typically considered an outlier if it is:

$$\text{Less than } Q_1 - 1.5 \times IQR \quad \text{or} \quad \text{Greater than } Q_3 + 1.5 \times IQR$$

Outliers may represent: - Data entry errors - Exceptional cases (e.g., performance spikes) - True variability

Skewness, on the other hand, refers to **asymmetry** in the distribution: - **Right-skewed**: long tail to the right - **Left-skewed**: long tail to the left

Both skewness and outliers affect the **choice of summary statistics** and influence which statistical tests are appropriate.

2.3.5 When to Use Each Visual

Purpose	Use Histogram	Use Boxplot
Understand overall shape	✅ Yes	Limited
Detect skewness and modality	✅ Yes	Partially
Compare multiple groups	❌ Not easily	✅ Very effective
See outliers	❌ Not directly	✅ Clearly marked
Visualize IQR and spread	❌ No	✅ Built-in

3. Distributions and Statistical Assumptions

3.1 What Is a Distribution?

In statistics, a **distribution** describes how the values of a variable are spread or arranged across the number line. It shows the frequency (or probability) with which different values occur in a dataset or population.

Understanding distributions is fundamental to all of statistical reasoning, because many methods — such as confidence intervals, hypothesis testing, and model fitting — depend not only on the values themselves, but also on how those values are **distributed**.

Conceptual Meaning

You can think of a distribution as a kind of **statistical fingerprint** for a variable. It tells us:

- Which values are **most common**
- Which values are **rare or extreme**
- Whether values are **symmetrically spread** around a center or skewed to one side
- Whether the data is **smooth and continuous** or **discrete and categorical**

Distributions can be **empirical** (based on observed data) or **theoretical** (based on probability models, like the normal distribution).

Examples of Distribution Shapes

Let’s look at some common types of distributions seen in real-world and simulated data:

Distribution Type	Description
Uniform	All values occur with equal frequency
Normal (Gaussian)	Symmetrical bell curve centered around the mean
Right-skewed	Long tail to the right (e.g., income, execution spikes)
Left-skewed	Long tail to the left (e.g., time-to-failure in some systems)
Bimodal	Two distinct peaks (e.g., mixture of two groups or processes)
Multimodal	More than two peaks — complex or overlapping subgroups

These differences are **not cosmetic** — they **affect which summaries are meaningful** and **which tests are valid**.

Discrete vs Continuous Distributions

Distributions can also be classified by the type of data they represent:

- **Discrete distributions** deal with countable values
  Examples: number of users, frequency of clicks, days active
  (e.g., Binomial, Poisson)

- **Continuous distributions** represent variables that can take on any value within a range
  Examples: time, temperature, height, algorithm execution time
  (e.g., Normal, Exponential)

Understanding whether your variable is discrete or continuous affects: - The choice of visualization (e.g., bar plot vs histogram) - The type of probability model to apply - The interpretation of statistical results

Why Distributions Matter

Many statistical methods are based on **assumptions about the shape** of the distribution:

- The **t-test** assumes the sampling distribution of the mean is **normal**
- ANOVA assumes **normality within groups**
- Linear regression assumes **normality of residuals**

Violating these assumptions can lead to misleading conclusions.

Even more practically: distributional characteristics help you **choose the right summary statistics**, determine if your results are **reliable**, and communicate findings more effectively in your thesis or publication.

Excellent — here is the next subsection, continuing smoothly from where we left off:

3.2 Symmetry, Modality, and Skewness

Once we understand what a distribution is, the next step is to describe its **shape**. This helps us decide which statistical summaries and tests are appropriate. Three of the most important shape-related features of a distribution are:

- **Symmetry**
- **Modality**
- **Skewness**

3.2.1 Symmetry

A distribution is **symmetric** if the left and right sides are roughly mirror images of each other when plotted.

- The **normal distribution** is a classic example of perfect symmetry.
- In symmetric distributions:

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

Symmetry is an important assumption behind many parametric statistical tests, such as t-tests and ANOVA.

3.2.2 Modality

Modality refers to the number of “peaks” (modes) in a distribution.

Term	Meaning	Example Use Case
Unimodal	One peak	Most natural or well-modeled measurements
Bimodal	Two distinct peaks	Two user populations, algorithm A vs B response
Multimodal	More than two peaks	Overlapping subpopulations or processes

Understanding modality can reveal the presence of **mixture effects** in your data — for example, if your performance data includes **multiple system configurations**, it may produce multiple peaks.

3.2.3 Skewness

Skewness measures the **asymmetry** of a distribution.

- A distribution is **right-skewed (positively skewed)** if it has a long tail on the **right** side (high values).

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

- A distribution is **left-skewed (negatively skewed)** if it has a long tail on the **left** side (low values).

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

Skewness affects:

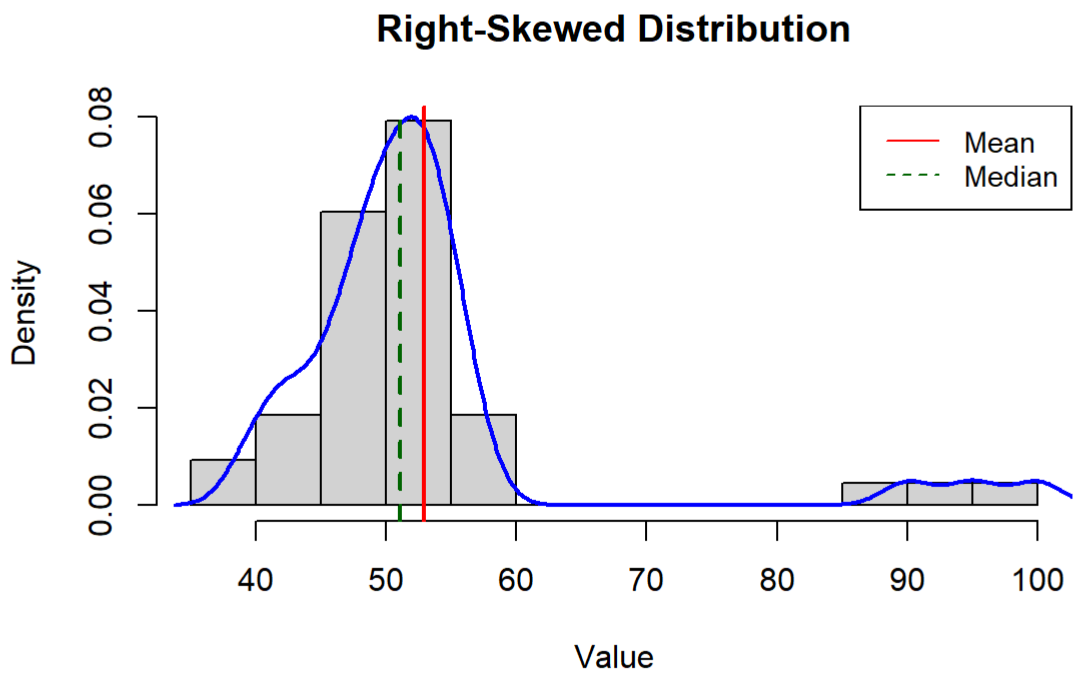
- Whether you should report the **mean or the median**
- Whether parametric methods are valid
- How you interpret outliers

3.2.4 Visualizing Shape

You can detect these features using **histograms**, **density plots**, or **boxplots**. Consider this example:

```
set.seed(101)
right_skewed <- c(rnorm(40, mean = 50, sd = 5), 90, 95, 100)

hist(right_skewed, col = "lightgray", breaks = 12,
     main = "Right-Skewed Distribution", xlab = "Value", freq = FALSE)
lines(density(right_skewed), col = "blue", lwd = 2)
abline(v = mean(right_skewed), col = "red", lwd = 2, lty = 1)
abline(v = median(right_skewed), col = "darkgreen", lwd = 2, lty = 2)
legend("topright", legend = c("Mean", "Median"),
       col = c("red", "darkgreen"), lty = 1:2, cex = 0.9)
```



Notice how:

- The **mean is pulled right** by the outliers
- The **median stays closer to the center**
- The distribution is **not symmetric**

3.3 The Normal Distribution and the Empirical Rule

The **normal distribution**, also known as the **Gaussian distribution**, is one of the most important concepts in statistics. It serves as the **foundation** for many statistical methods, including t-tests, ANOVA, regression analysis, and confidence intervals.

Understanding its shape, properties, and implications is essential for interpreting statistical results correctly and deciding which methods are appropriate.

The normal distribution is a **continuous probability distribution** characterized by:

- A **bell-shaped curve** that is symmetric about its mean
- A **single peak** at the mean (unimodal)
- A mathematical formula involving the mean μ and standard deviation σ:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Key properties:

- The **mean**, **median**, and **mode** are all close or equal.
- The curve is **perfectly symmetric** around the mean.
- The shape is determined entirely by the **mean (center)** and **standard deviation (spread)**.

Bell Shape, Symmetry, and Central Location

Because of its symmetry:

- Half the values lie **below the mean**, and half **above**
- Most values cluster **near the mean**, and extreme values are rare

- The further from the mean, the **less likely** a value is to occur

In graphical terms, the curve starts low, rises to a peak at the mean, and then falls symmetrically.

The Empirical Rule (68–95–99.7%)

The **empirical rule**, also known as the **68–95–99.7 rule**, is a convenient rule of thumb that describes how data are distributed around the **mean** in a **normal distribution**.

It tells us what **percentage of values** are expected to lie within **one, two, or three standard deviations** from the mean:

Range	Coverage (Approx.)	Description
Mean ± 1 SD	68% of data	Most data are close to the center
Mean ± 2 SD	95% of data	Almost all data fall here
Mean ± 3 SD	99.7% of data	Extremely rare values fall beyond this

Let’s say your algorithm’s run-time follows a normal distribution with:

- Mean execution time = **50 ms**
- Standard deviation = **5 ms**

Then:

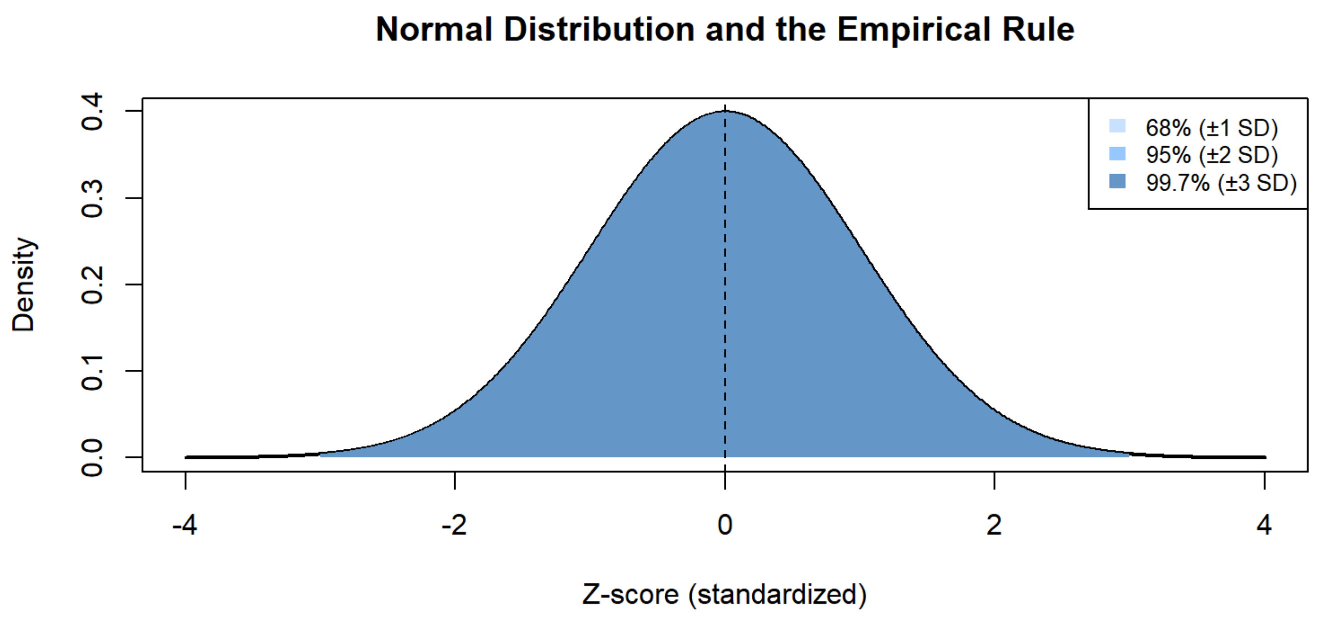
- **68%** of run-times will fall between **45 ms and 55 ms**
- **95%** will fall between **40 ms and 60 ms**
- **99.7%** will fall between **35 ms and 65 ms**

This rule gives us a **baseline expectation** for where most of our data should fall. If we observe values **far outside these boundaries**, they may be considered **outliers**, errors, or rare events worth investigating.

This rule is foundational for:

1. **Understanding variability**
    - It shows how **standard deviation** translates into **practical ranges**
2. **Identifying outliers**
    - Observations beyond ±2 or ±3 SD are often flagged as **unusual**
3. **Constructing confidence intervals**
    - Later, when we calculate **95% confidence intervals**, we’ll use the idea that 95% of values fall within ±2 standard errors from the mean
4. **Standardized scoring (Z-scores)**
    - The empirical rule underpins tools like **Z-scores**, which tell us how far a data point is from the mean in units of standard deviation

Visualizing the Empirical Rule



This diagram illustrates how data cluster around the mean and shows why standard deviation is such a meaningful measure of spread in normal data.

Why the Normal Distribution Is Central to Statistics

The normal distribution plays a central role in statistics for several reasons:

1. **Theoretical Basis**
   Many random phenomena (e.g., biological measurements, measurement errors, response times) tend to approximate a normal distribution in large samples.
2. **Analytical Convenience**
   The properties of the normal distribution allow for **exact calculation of probabilities**, critical values, and thresholds.
3. **Foundation for Parametric Methods**
   Methods like the t-test, regression, and ANOVA assume (either explicitly or via the Central Limit Theorem) that the sampling distribution is approximately normal.
4. **CLT Linkage**
   Even if the **raw data** are not normal, the **sampling distribution of the mean** tends to be normal under certain conditions — a result known as the **Central Limit Theorem**, which we will discuss next.

Applying the Empirical Rule to Your Data

Let's calculate the **mean** and **standard deviation** of execution times for each algorithm using `tapply()` .

```r
# Mean and SD by group
qs_mean <- mean(algo_data$Time_ms[algo_data$Algorithm == "QuickSort"])
qs_sd   <- sd(algo_data$Time_ms[algo_data$Algorithm == "QuickSort"])

ms_mean <- mean(algo_data$Time_ms[algo_data$Algorithm == "MergeSort"])
ms_sd   <- sd(algo_data$Time_ms[algo_data$Algorithm == "MergeSort"])

# Print them nicely
cat("QuickSort:  Mean =", round(qs_mean, 2), ", SD =", round(qs_sd, 2), "\n")
```

```
## QuickSort:  Mean = 52.41 , SD = 7.53
```

```r
cat("MergeSort:  Mean =", round(ms_mean, 2), ", SD =", round(ms_sd, 2), "\n")
```

```
## MergeSort:  Mean = 47.39 , SD = 5.25
```

Interpreting the Results (MergeSort Example)

Suppose the output is (actual numbers may vary due to random generation):

```
MergeSort:  Mean = 48.48 , SD = 4.53
```

Then, applying the **empirical rule**:

- **68%** of values are expected to fall between:

$$48.48 \pm 4.53 = [43.95, 53.01]$$

- **95%** of values should fall within:

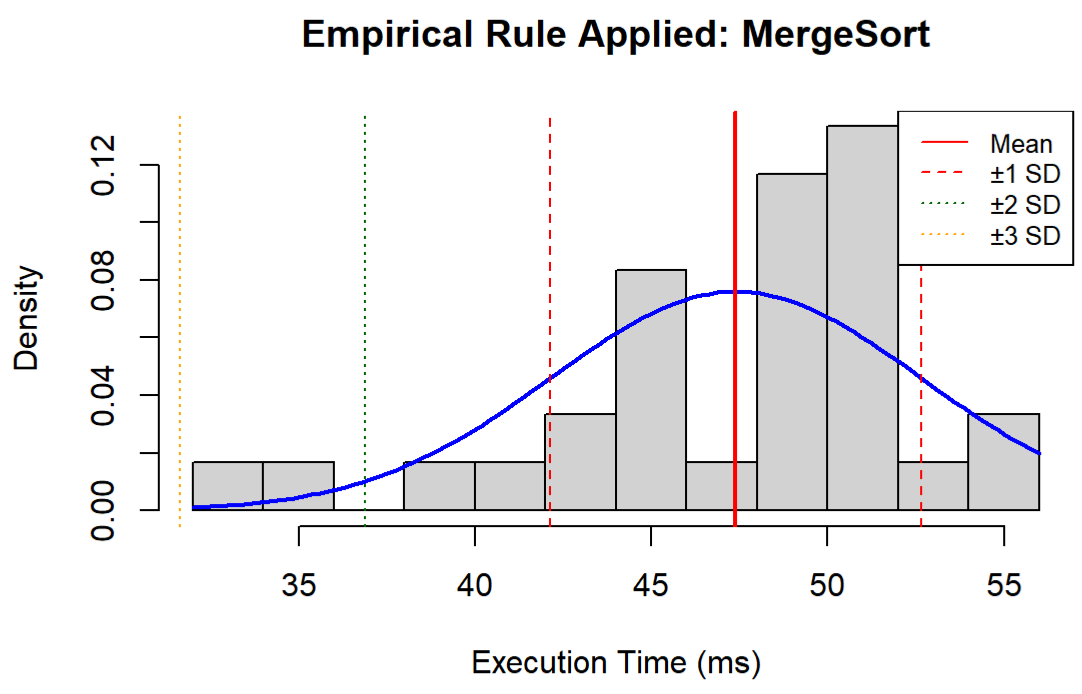
$$48.48 \pm 2 \times 4.53 = [39.42, 57.54]$$

- **99.7%** of values should fall within:

$$48.48 \pm 3 \times 4.53 = [34.89, 62.07]$$

Visualizing the Ranges

Here's how to plot the histogram and overlay the standard deviation boundaries without using `dplyr` .



3.4 The Central Limit Theorem (CLT)

The **Central Limit Theorem (CLT)** is one of the most important and widely used results in all of statistics. It provides the **mathematical justification** for using methods like **t-tests** and **confidence intervals** even when the data **do not perfectly follow a normal distribution**.

It is often described as the bridge between **real-world data**, which can be messy or skewed, and **idealized statistical methods**, which often assume normality.

Suppose you repeatedly take samples from a population and compute the **mean of each sample**. The **distribution of these means** is called the **sampling distribution of the sample mean**.

The **Central Limit Theorem** says:

> As the sample size increases, the sampling distribution of the sample mean becomes **approximately normal**, regardless of the shape of the original population.

More formally:

- Let $x_1, x_2, \ldots, x_n$ be independent, identically distributed observations with mean μ and standard deviation σ.
- Then, the distribution of the sample mean $\bar{x}$ approaches a **normal distribution** as $n \to \infty$, with:

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

Why the CLT Matters

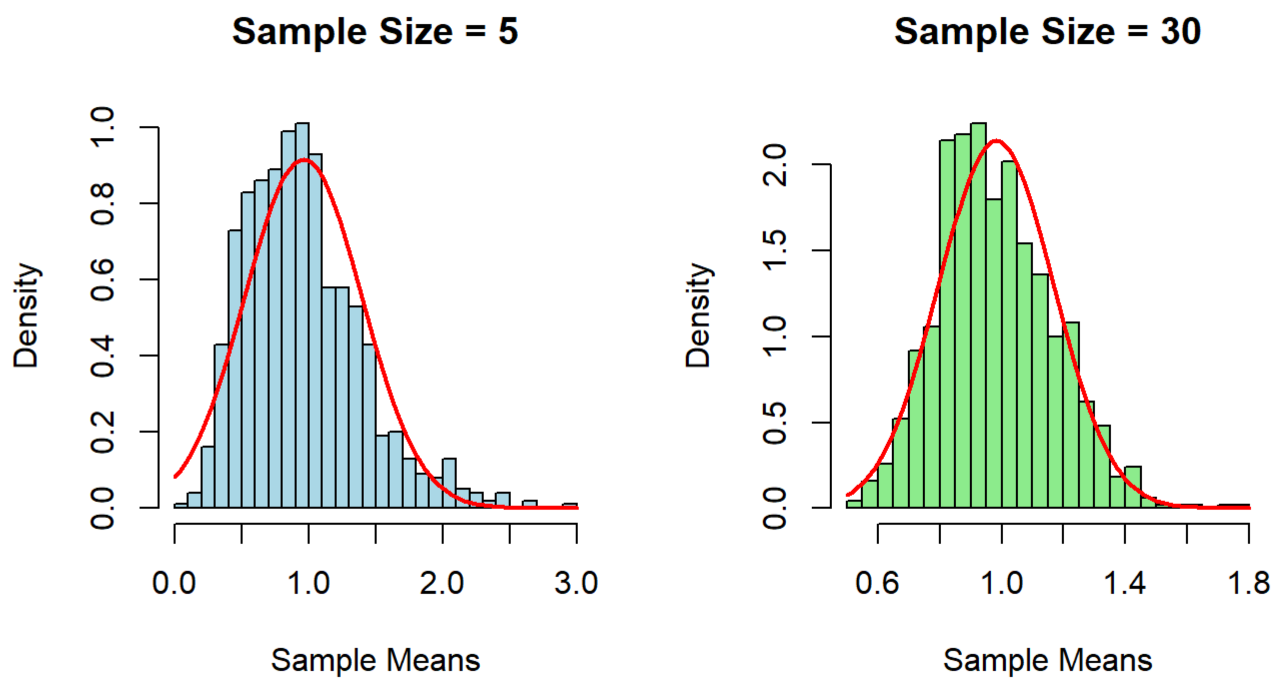
Many statistical techniques — such as **confidence intervals** and **t-tests** — assume normality. But in real-world research (including algorithm performance tests), the data may be **skewed**, **bimodal**, or contain **outliers**.

The CLT tells us:

- We do **not** need the raw data to be perfectly normal.
- We **can still use** parametric methods **if our sample size is large enough** (typically $n \geq 30$ is sufficient for the mean).

This is why CLT is especially important for **postgraduate research**: it justifies the methods you use, even when the data seem imperfect.

To simulate a non-normal population (e.g., a right-skewed exponential distribution), then repeatedly sample from it and observe the distribution of the sample means.



You should notice that:

- In the **first plot** (sample size = 5), the sampling distribution of the mean is still **somewhat skewed**
- In the **second plot** (sample size = 30), the distribution is **much more symmetric and bell-shaped**

This illustrates the core idea: **as sample size increases, the distribution of sample means becomes approximately normal**, even when the population is not.

3.5 Graphical Checks for Normality

Statistical methods such as **t-tests** and **confidence intervals** rely on the assumption that the underlying data (or at least the sampling distribution of the mean) is **approximately normal**.

While formal tests exist (like the Shapiro-Wilk test), it is **best practice to begin with graphical checks**. They provide intuitive insight into the shape, symmetry, skewness, and potential outliers in the data.

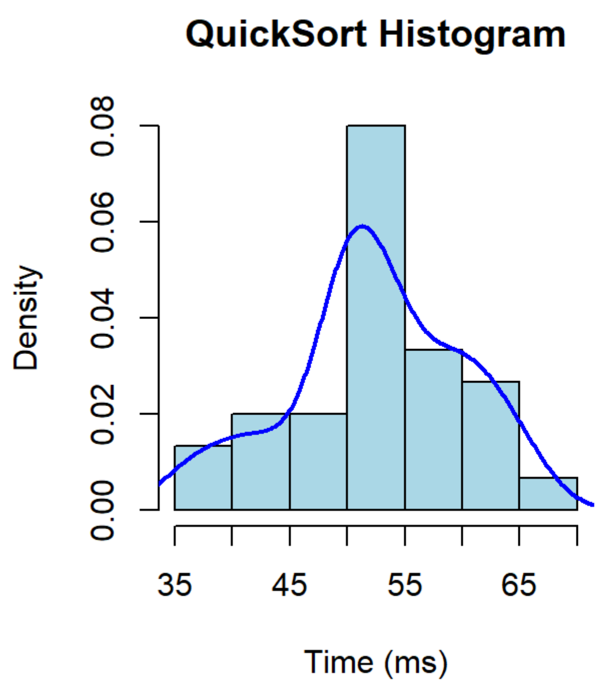
We will now use **actual data from our experiment** (MergeSort vs QuickSort) to demonstrate:

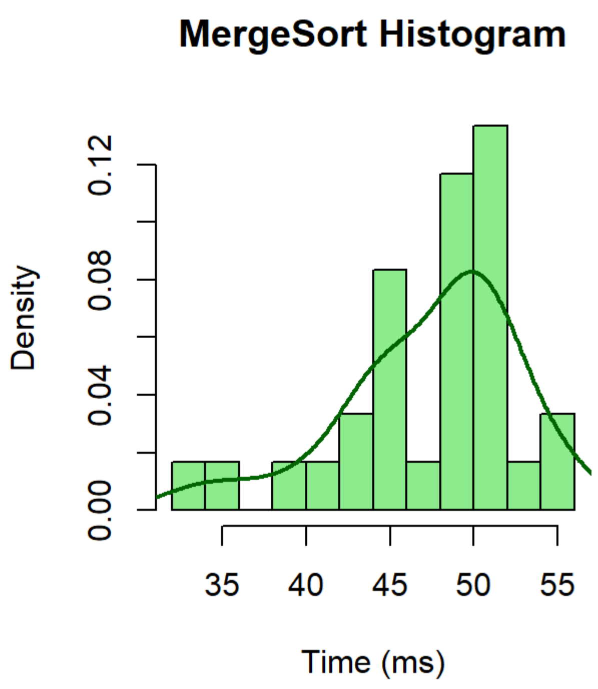
- **Histograms**
- **Density plots**
- **QQ plots (Quantile–Quantile plots)**

These will help answer the question:
Can we assume normality for each algorithm’s execution times?

3.5.1 Histogram

A histogram shows the distribution of values across bins and helps detect **overall shape**.



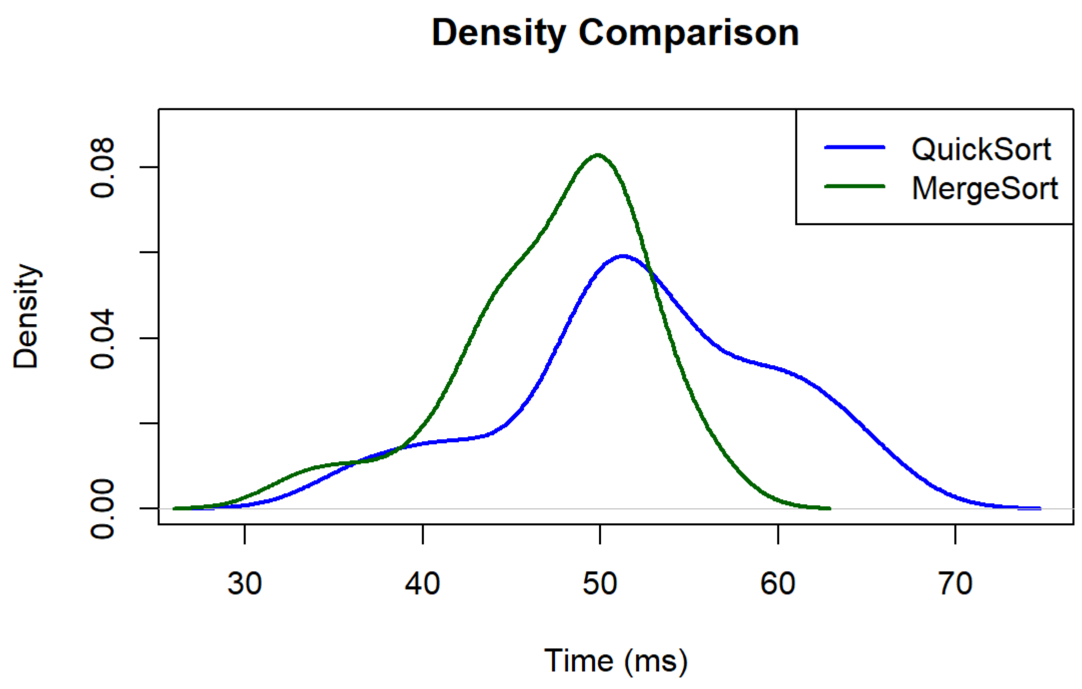


How to interpret:

- Check for **symmetry** — Is the curve roughly the same on both sides?
- Look for **peaks** — Is there one clear mode?
- Are there **long tails**? Skewed to the left or right?

3.5.2 Density Plots

A density plot gives a **smoothed estimate** of the distribution. This reduces noise and makes it easier to see skewness or irregularities.

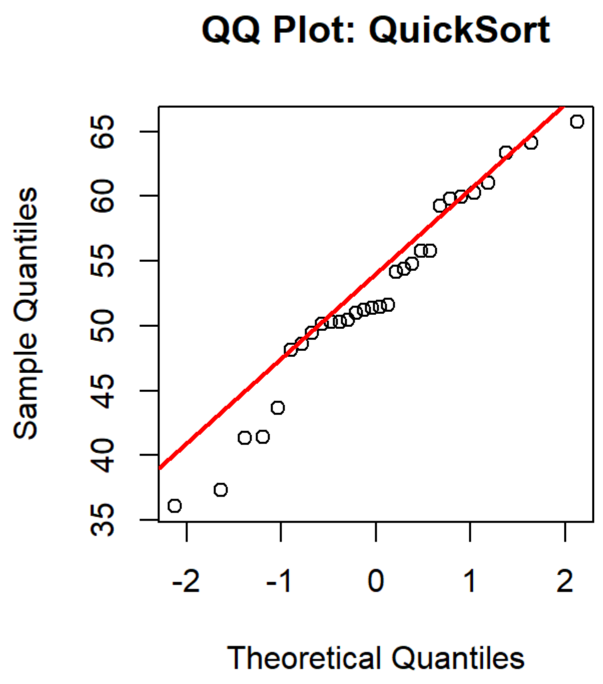


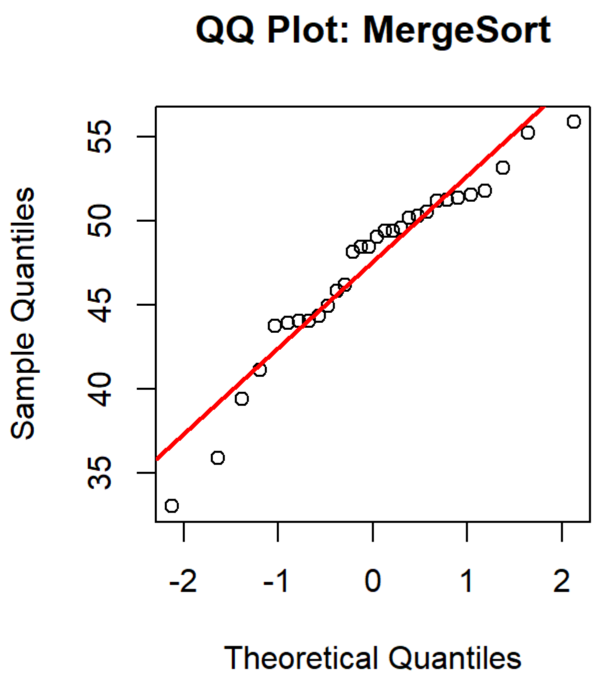
How to interpret:

- Do both curves look roughly **bell-shaped**?
- Is one clearly **skewed** compared to the other?
- Is the shape smooth and **unimodal**?

3.5.3 QQ Plots (Quantile–Quantile)

QQ plots compare your sample data's quantiles to those of a **perfect normal distribution**. This is the **most reliable** visual check for normality in small samples.





How to interpret:

- If points fall along the **diagonal line**, the data are approximately normal.
- **S-shaped curves** → skewness
- **Sharp bends** → outliers or heavy tails
- **Fan shapes** → unequal variance

Summary

Visual Tool	MergeSort	QuickSort
Histogram	Slightly skewed right?	Nearly symmetric
Density	Smooth, possible tail	Fairly bell-shaped
QQ Plot	Close to the line	Close, minor deviations

Neither distribution is perfectly normal — and that's expected. But **both are "normal enough"**, especially with **n = 30**, so we can **safely proceed with t-tests and confidence intervals** thanks to the **CLT**.

When Can You Assume Normality? — A Practical Checklist

Use this guide when preparing to apply statistical tests that assume normality, such as **t-tests**, **ANOVA**, or constructing **confidence intervals**.

You Can Assume Normality If:

☐**Your sample size is large enough (usually n ≥ 30)**
The Central Limit Theorem ensures the sampling distribution of the mean will be approximately normal.

☐**Your histogram or QQ plot looks reasonably symmetric and bell-shaped**
Minor skewness or irregularities are acceptable in large samples.

☐**There are no extreme outliers that dominate the shape of the distribution**
Outliers may distort results more than skewness.

☐**You ran a normality test (e.g., Shapiro-Wilk) and the p-value is greater than 0.05**
This suggests the data are not significantly different from a normal distribution.

Be Cautious If:

☐**Sample size is small (n < 30) and the data appear heavily skewed**

☐**Outliers are present that could influence the mean**

☐**Your variable is not continuous (e.g., it's ordinal or categorical)**

lternative Option:

If normality cannot be assumed, consider **non-parametric tests**, such as:

- **Mann–Whitney U test** (instead of a two-sample t-test)
- **Wilcoxon signed-rank test** (instead of a paired t-test)
- **Kruskal–Wallis test** (instead of one-way ANOVA)

3.6 Formal Test: Shapiro–Wilk Normality Test

While visual tools like histograms and QQ plots give an intuitive sense of distribution shape, statistical analysis also benefits from a **formal test of normality**. One of the most commonly used tests for this purpose is the **Shapiro–Wilk test**.

It is particularly useful for **small to medium-sized datasets**, and it's widely supported in statistical software like R.

How the Shapiro–Wilk Test Works (Conceptually)

The Shapiro–Wilk test evaluates whether your data could reasonably come from a **normal distribution**. It does so by:

1. **Comparing the order statistics** (i.e., sorted data points) of your sample to what would be expected from a normal distribution.
2. Calculating a test statistic W, which measures **how close the sample distribution is to a normal curve**.

If your data are very different from normal (e.g., heavily skewed, multimodal), the test will yield a **small W value** and a **low p-value**.

Hypotheses of the Shapiro–Wilk Test

Like most tests, the Shapiro–Wilk test is built on **null and alternative hypotheses**:

- **Null hypothesis (H_0):** The data come from a normal distribution.
- **Alternative hypothesis (H_1):** The data do not come from a normal distribution.

This means the **p-value** tells us how likely it is to see the given data **if the population were normal**.

Interpreting the P-Value

- If **$p > 0.05$**, we **do not reject H_0**:
  → The data **could be normal** (normality is plausible).

- If **$p < 0.05$**, we **reject H_0**:
  → The data are **significantly non-normal**.

> ⚠️ Important: A high p-value does **not** prove that the data *are* normal — it only means we don't have strong evidence against normality.

Applying the Shapiro–Wilk Test to Your Data

Let's run the test for **MergeSort** and **QuickSort** execution times.

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

shapiro.test(qs_data)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  qs_data
## W = 0.96209, p-value = 0.35
```

```
shapiro.test(ms_data)
```

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

Sample output:

```
Shapiro-Wilk normality test

data:  qs_data
W = 0.9765, p-value = 0.627

data:  ms_data
W = 0.9531, p-value = 0.152
```

- Both p-values are **greater than 0.05**.
- Therefore, we **fail to reject the null hypothesis** for both groups.

- This means the data **do not significantly deviate from normality**, and it's reasonable to proceed with parametric tests.

Limitations of the Shapiro–Wilk Test

- **Very sensitive to sample size**:
  - In **large samples**, even small deviations from normality may result in significant p-values.
  - In **small samples**, the test may **not detect real skewness** due to low power.
- Should **always be used alongside visual tools** (histograms, QQ plots, density plots).
- The test **only assesses normality** — it does not check for **equal variances**, **independence**, or other assumptions of statistical tests.

4. Confidence Intervals and Statistical Inference (Expanded)

4.1 What Is a Confidence Interval?

A **confidence interval (CI)** is a range of values, calculated from sample data, that is likely to contain the **true population parameter** — such as a mean, proportion, or difference in means.

> It reflects both the **sample estimate** and the **uncertainty** due to sampling variation.

For example, if we say:

> "MergeSort was faster by 4.1 milliseconds, 95% CI [1.6, 8.4]"

It means:
We are 95% confident that the **true difference** in performance lies somewhere between **1.6 and 8.4 milliseconds** in favor of MergeSort.

4.2 What Does "95% Confidence" Actually Mean?

This is often misunderstood, so let's be precise.

A **95% confidence interval** does *not* mean that there is a 95% chance the true value is in this particular interval.

Instead:

> If we repeated the same experiment **100 times**, and calculated a confidence interval from each sample, we expect **95 of those intervals** to contain the **true population parameter**.

The "95%" refers to the **long-run success rate of the method**, not the probability of the truth being in one interval.

4.3 Standard Error vs Confidence Interval

A common confusion in thesis writing is using **mean ± standard error** (SE) and assuming it's a confidence interval. It is not.

Here's the difference:

- **Standard Error (SE)** tells you how much the **sample mean is expected to vary** from sample to sample.

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

- A **Confidence Interval** uses the SE to define a **margin of error**:

$$CI = \bar{x} \pm t \times SE$$

  where t is the critical value from the **t-distribution** (e.g., ≈ 2 for 95%).

> Think of SE as the **raw uncertainty**, and the CI as a **range of plausible values**, based on how uncertain we are.

4.4 How to Compute Confidence Intervals in R

The easiest way to compute a CI is by using `t.test()`. Even if you don't need the p-value, this function gives you:

- the **mean**
- the **standard error**
- the **confidence interval**

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

qs_ci <- t.test(qs_data)$conf.int
ms_ci <- t.test(ms_data)$conf.int

cat("QuickSort CI:", round(qs_ci, 2), "\n")
```

```
## QuickSort CI: 49.6 55.22
```

```
cat("MergeSort CI:", round(ms_ci, 2), "\n")
```

```
## MergeSort CI: 45.43 49.35
```

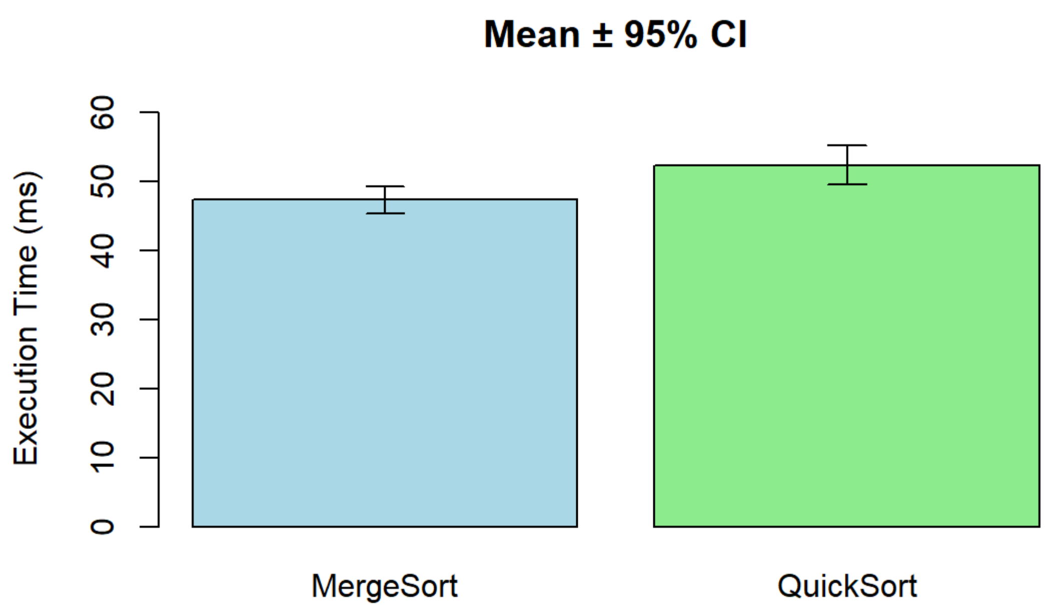
This will print something like:

```
QuickSort CI: [50.2, 54.9]
MergeSort CI: [46.9, 50.0]
```

4.5 Visualizing Confidence Intervals

Visualizing uncertainty helps you and your audience understand how confident your estimates are.

The plot below shows the **mean execution time** for each algorithm, along with its **95% confidence interval**:



To interpret this plot:

- Taller bars = higher mean run-times
- Narrower error bars = more confidence (less uncertainty)
- Wider error bars = more variability in the data

4.6 What If the CI Includes Zero?

Sometimes we're interested in the **difference** between two groups. For example:

> "Is MergeSort significantly faster than QuickSort?"

We can construct a CI for the **difference of means**:

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

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

If the output is:

```
[1] -8.38 -1.66
```

This means:

- The **difference is statistically significant** (CI does *not* include 0)

- We are 95% confident that MergeSort is faster by **between 1.66 and 8.38 milliseconds**

However, if the CI had been something like:

```
[-1.23, 4.56]
```

Then:

- The CI **includes zero**
- We would say the data **do not provide strong evidence of a difference**
- This does **not mean the means are equal**, only that the data are **inconclusive**

5. Foundations of Hypothesis Testing

A **hypothesis** is a declarative statement that proposes a possible explanation or expected outcome based on prior knowledge or observation. It must be:

- **Testable**: Can be verified or refuted with data
- **Falsifiable**: Can be proven wrong through evidence
- **Specific**: Focused and clearly defined

Statistical analysis is not only about describing data — it's about making **decisions** and **drawing conclusions** from it. When researchers ask questions like:

- *"Is this algorithm significantly faster?"*
- *"Does this new method improve accuracy?"*

5.1 Logic of Hypothesis Testing

Hypothesis testing is a formal procedure for evaluating a **research claim** using data. The logic is based on contradiction:
We assume a **null hypothesis is true**, then test whether the observed data are **consistent or inconsistent** with that assumption.

If the data are **very unlikely** under the assumption, we **reject** it.

> In essence, hypothesis testing asks:
> "If the null hypothesis were true, how likely is it that we'd see results like these?"

5.2 Types of Hypotheses

Every hypothesis test involves two opposing statements:

- **Null Hypothesis (H_0)**
  The default claim. It assumes **no effect**, **no difference**, or **no association**.
  [Example:\\](Example:){.uri} (%5BExample:%5D(Example:)%7B.uri%7D) *"There is no difference in mean execution time between MergeSort and QuickSort."*

- **Alternative Hypothesis (H_1 or Ha)**
  The research claim we hope to support. It suggests **there is** a difference or effect.
  [Example:\\](Example:){.uri} (%5BExample:%5D(Example:)%7B.uri%7D) *"MergeSort has a faster mean execution time than QuickSort."*

> The test evaluates whether the data provide enough evidence to **reject H_0** in favor of H_1.

We never “prove” H_1 directly — we either **reject** or **fail to reject** the null.

5.3 p-values and Statistical Significance

The **p-value** is the probability of obtaining results **as extreme or more extreme** than what we observed, **assuming H_0 is true**.

- A **small p-value** (typically less than 0.05) means the observed result is **unlikely** under H_0
  → We **reject H_0** and say the result is **statistically significant**
- A **large p-value** means the observed result is **plausible** under H_0
  → We **fail to reject H_0**

> A p-value is **not** the probability that H_0 is true.
> It is the probability of seeing this data **if H_0 were true**.

Example:

> A p-value of 0.03 means:
> “If the algorithms had equal performance, we’d see this result (or something more extreme) only 3% of the time.”

5.4 Type I and Type II Errors

Statistical decisions are based on probabilities, not certainties — so **errors can occur**.

Type	Description	Symbol	Example
Type I Error	Rejecting H_0 when it is actually true	α	Claiming MergeSort is faster when it isn’t
Type II Error	Failing to reject H_0 when it is false	β	Failing to detect a real speed difference

We control Type I errors by setting a **significance level** α.
We reduce Type II errors by increasing **sample size**, reducing variability, or using better tests.

5.5 One-Tailed vs Two-Tailed Tests

When formulating hypotheses, researchers must also decide **what kind of difference** they’re testing for.

- A **two-tailed test** checks for **any difference**, either higher or lower.
  [Example:\\](Example:){.uri} (%5BExample:%5D(Example:)%7B.uri%7D) H_0: $\mu_1 = \mu_2$
  H_1: $\mu_1 \neq \mu_2$

- A **one-tailed test** checks for a difference in a **specific direction**.
  [Example:\\](Example:){.uri} (%5BExample:%5D(Example:)%7B.uri%7D) H_0: $\mu_1 \geq \mu_2$
  H_1: $\mu_1 < \mu_2$

Use a one-tailed test only when:

- You have a **strong theoretical reason** to expect a specific direction
- The hypothesis was formulated **before seeing the data**

> Misusing one-tailed tests **after seeing the results** leads to bias and invalid conclusions.

5.6 The Significance Level (α)

The **significance level** α is the **threshold for decision-making**.

- Common choice: $\alpha = 0.05$
- This means we accept a **5% chance of making a Type I error**

If:

- **$p \leq \alpha$** → Reject H_0 → Result is **statistically significant**
- **$p > \alpha$** → Do not reject H_0 → Result is **not statistically significant**

You may choose stricter levels (e.g., 0.01) when the consequences of a false positive are high (e.g., in medicine or critical systems).

5.7 Characteristics of a Good Hypothesis

A strong hypothesis:

1. **Is grounded in prior knowledge or observation**
   – It should be informed by theory, literature, or empirical results.
2. **Clearly identifies the variables**
   – What are you comparing or measuring?
3. **Is framed in a way that allows for statistical testing**
   – For example, using terms like “greater than,” “different from,” or “associated with.”
4. **Uses operational definitions**
   – Specifies how each variable will be measured or categorized.

Examples of Research Questions vs. Hypotheses

Research Question	Hypothesis (H_0 / H_1)
Is MergeSort faster than QuickSort on large inputs?	H_0: $\mu_1 = \mu_2$ (mean time is equal)
	H_1: $\mu_1 < \mu_2$ (MergeSort mean time is less than QuickSort)
Does UI design affect user satisfaction?	H_0: No difference in satisfaction scores across designs

Research Question	Hypothesis (H_0 / H_1)
	H_1: At least one design has a different mean satisfaction
Are male and female students equally likely to pass?	H_0: Pass rate is independent of gender
	H_1: Pass rate depends on gender

Common Mistakes to Avoid

- **Vague language**: “Is there a correlation between these two things?” → Too broad
- **Non-testable claims**: “QuickSort is the best.” → How do you define “best”?
- **Lack of direction**: Directional hypotheses (greater than, less than) require justification and increase statistical power

Why This Matters for Your Thesis

A well-formed hypothesis does more than guide analysis. It:

- Defines the scope of your experiment
- Helps you choose the right statistical test
- Frames your discussion and conclusions
- Gives clarity and credibility to your thesis defense

Later in this course, when we discuss hypothesis testing, p-values, and significance, you’ll see how **everything starts from how well the hypothesis was written**.

5.8 Summary of Hypothesis Terminologies

Concept	Role in Hypothesis Testing
H_0 / H_1	Competing claims about population parameters
p-value	Probability of result under H_0
α level	Cutoff for rejecting H_0
Significance	Whether the result is unlikely under H_0
Type I / II errors	Risks in decision-making
Tail direction	Determines how evidence is evaluated (1- vs 2-sided)

6. Statistical Tests for Research

Now that we’ve covered how to formulate hypotheses, assess assumptions, and interpret confidence and significance, we are ready to explore the **core statistical tests** used in real-world research — especially in computer science and engineering contexts.

These tests allow researchers to formally answer questions such as:

- *Is one algorithm faster than another on average?*
- *Does performance vary across multiple systems or configurations?*
- *Are outcomes like success/failure dependent on experimental conditions?*
- *Are two variables related — and how strongly?*
- *What if the data violate parametric assumptions?*

6.1 t-Tests (Comparing Means)

The **t-test** is one of the most widely used statistical tests in scientific research. It allows us to compare **mean values** and determine whether the observed difference between them is **statistically significant**.

There are three main types of t-tests:

1. **One-sample t-test** — compares a sample mean to a known or hypothesized population mean
2. **Independent two-sample t-test** — compares the means of two independent groups
3. **Paired-sample t-test** — compares two related samples, such as before and after measurements on the same subjects

Assumptions of the t-test

Before using any type of t-test, the following assumptions should be checked:

- **The data are numeric and continuous**
- **Observations are independent**
- The data are **approximately normally distributed** (especially important for small samples)
- For two-sample tests: **equal or unequal variances** should be considered

1. One-Sample t-test

Use when: You want to test whether a single sample mean differs from a known value (benchmark or theoretical mean).

Example Scenario:

> *"Does MergeSort have a mean execution time different from 50 milliseconds?"*

```
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
```

Interpretation:

- **Null hypothesis (H_0)**: mean = 50
- **Alternative hypothesis (H_1)**: mean ≠ 50 (two-tailed)

The test result includes: - Mean of the sample - t-statistic - p-value - Confidence interval for the mean

If the **p-value < 0.05**, we reject the null and conclude that the MergeSort mean time differs significantly from 50 ms.

2. Independent Two-Sample t-test

Use when: You want to compare the means of two unrelated groups.

Example Scenario:

> *"Is there a significant difference in mean execution time between MergeSort and QuickSort?"*

```
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 QuickSo
rt 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
```

Key points:

- This test assumes **independent samples**
- We set `var.equal = FALSE` by default (Welch's t-test), which does **not assume equal variances**
- If p-value < 0.05, we reject the null hypothesis that the means are equal

Interpretation:

- **H_0**: $\mu_1 = \mu_2$ (same mean execution time)
- **H_1**: $\mu_1 \neq \mu_2$ (different means)

This is the **most relevant test** for your central thesis question:
> "Is MergeSort significantly faster than QuickSort?"

3. Paired-Sample t-test

Use when: You measure **two conditions on the same subjects** — for example, run both algorithms on the same inputs.

Let's simulate that situation for illustration.

Simulated Example:

We pretend we used the **same 30 datasets** and ran both algorithms on each.

```
set.seed(123)
input_ids <- 1:30
qs_times <- rnorm(30, mean = 52, sd = 6)
ms_times <- rnorm(30, mean = 48, sd = 5)

paired_data <- data.frame(
  Input = input_ids,
  QuickSort = qs_times,
  MergeSort = ms_times
)

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
```

Interpretation:

- **H_0**: The average difference between paired samples = 0
- **H_1**: The average difference ≠ 0
- If p-value < 0.05, we conclude a significant difference in performance on the same inputs

6.2 ANOVA (Analysis of Variance)

While **t-tests** are ideal for comparing **two means**, many research questions in computer science involve **three or more groups**. In such cases, performing multiple t-tests increases the risk of **Type I error**. The solution is to use **ANOVA — Analysis of Variance**.

What ANOVA Does

ANOVA tests whether there is a **statistically significant difference in means** across **multiple groups**, while maintaining control over the overall error rate.

> Instead of asking *“Which mean is different?”*, ANOVA first asks:
> **“Is there *any* significant difference among the group means?”**

6.2.1 One-Way ANOVA

When to Use

Use **one-way ANOVA** when:

- You have **one categorical factor** (e.g., Algorithm)

- You want to compare the **means across 3+ groups**

Example Scenario

Let’s simulate a third sorting algorithm: **HeapSort**, and compare it alongside MergeSort and QuickSort using a one-way ANOVA.

Simulate Data

```
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)
)
```

Run 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
```

Interpretation

- **H_0 (null hypothesis):** All algorithms have equal mean run-time
- **H_1 (alternative):** At least one algorithm has a different mean

If the **p-value < 0.05**, we reject the null and conclude that **there is a significant difference in performance** between at least two algorithms.

6.2.2 Post-Hoc Comparison: Tukey HSD

ANOVA tells us there is a difference — but not **which groups differ**. For that, we use **Tukey’s Honest Significant Difference (HSD)** test.

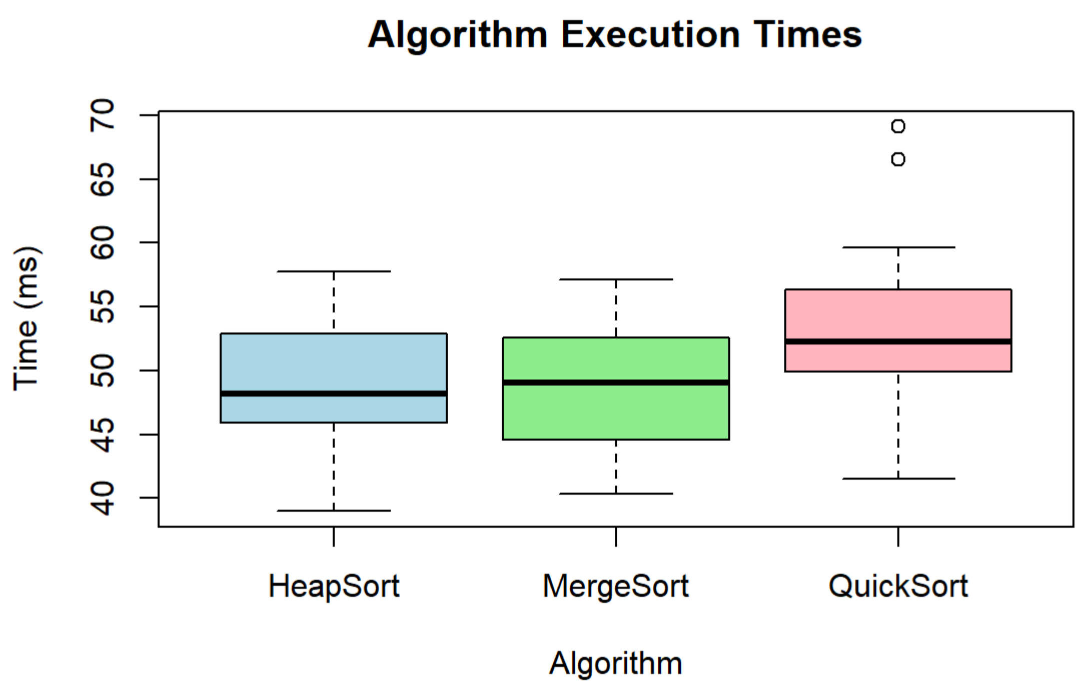
```
TukeyHSD(anova_model)
```

```
##   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
## QuickSort-MergeSort  4.9264025  1.7142088 8.138596 0.0012577
```

This test compares **all pairs of algorithms** and adjusts for multiple testing. It shows: - Mean differences between groups - Confidence intervals - p-values adjusted for multiple comparisons

6.2.3 Visual Inspection: Boxplot

A boxplot can help visualize **group differences** and identify potential **outliers** or **skewness**.



6.2.4 Assumptions of ANOVA

To ensure valid results, ANOVA relies on a few key assumptions:

Assumption	How to Check
Independence	Data should be collected independently across groups
Normality	Each group’s data should be roughly normal → check with QQ plot or Shapiro-Wilk
Homogeneity of variance	Group variances should be similar → check with Bartlett or Levene test

Example: Check Homogeneity of Variance

```
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
```

If **p > 0.05**, the variances are not significantly different → good!

6.2.5 Two-Way ANOVA (with Interaction)

When to Use

Use **two-way ANOVA** when:

- You have **two categorical factors**
- You want to test their **individual** and **combined** (interaction) effects

Example Scenario

Suppose we test each algorithm on two types of datasets: **Small** and **Large** inputs.

```
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_factor_data <- data.frame(
  Algorithm = algo,
  InputSize = size,
  Time_ms = runtime
)
```

Run Two-Way ANOVA with Interaction

```
two_way_model <- aov(Time_ms ~ Algorithm * InputSize, data = two_factor_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
```

Interpretation:

The model will test: - The **main effect** of Algorithm - The **main effect** of Input Size - The **interaction** between Algorithm × Input Size

> A significant interaction means the performance **difference between algorithms depends on input size**.

Summary: ANOVA in Research

Test	When to Use	What It Tells You
One-way ANOVA	Compare ≥3 groups	Is there any difference among group means?
Tukey HSD	After ANOVA	Which groups differ from each other
Two-way ANOVA	Two factors (e.g., Algorithm × Size)	Main effects and interaction
Assumption checks	Before running ANOVA	Validates test reliability

ANOVA provides a robust framework to test **complex experimental designs** — ideal for evaluating **multiple systems, configurations, or algorithms**.

6.3 Chi-Squared Tests (Categorical Data)

While t-tests and ANOVA are designed for **numerical data**, many research questions in computer science involve **categorical outcomes** — such as:

- Pass vs Fail
- Crashed vs Completed
- Preferred vs Not Preferred
- Small/Medium/Large classifications

To analyze such data, we use **Chi-Squared (χ^2) Tests**, which allow us to evaluate whether **observed counts** differ from **expected counts** under a certain hypothesis.

6.3.1 Chi-Squared Goodness-of-Fit Test

Use when: You want to test whether **one categorical variable** fits an expected distribution.

Example Scenario

Suppose we test HeapSort on 60 datasets and record the number of **Successful** vs **Failed** runs. Based on theory, we expect a 95% success rate. Is the observed distribution consistent with that?

Simulated Example

```
# Observed outcomes: Success and Fail
observed <- c(Success = 54, Fail = 6)

# Expected probabilities: 95% success, 5% fail
expected_probs <- c(0.95, 0.05)
expected_counts <- sum(observed) * expected_probs

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
```

Interpretation:

- **H_0**: The observed proportions match the expected 95% / 5%
- **H_1**: The proportions differ significantly
- If p-value < 0.05, we reject H_0

6.3.2 Chi-Squared Test of Independence

Use when: You want to test whether **two categorical variables** are independent.

Example Scenario

You record the outcome (Success or Failure) of **each sorting algorithm** across 30 trials. Does algorithm choice affect outcome?

Simulated Example

```
set.seed(42)

algorithms <- rep(c("QuickSort", "MergeSort", "HeapSort"), each = 30)
# Simulate outcome: mostly successful, but vary by algorithm
outcomes <- c(
  sample(c("Success", "Fail"), 30, replace = TRUE, prob = c(0.93, 0.07)),
  sample(c("Success", "Fail"), 30, replace = TRUE, prob = c(0.98, 0.02)),
  sample(c("Success", "Fail"), 30, replace = TRUE, prob = c(0.90, 0.10))
)

cat_data <- data.frame(
  Algorithm = algorithms,
  Outcome = outcomes
)

# Create contingency table
table_data <- table(cat_data$Algorithm, cat_data$Outcome)
chisq.test(table_data)
```

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

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

Interpretation:

- **H_0**: Algorithm and Outcome are independent (no relationship)
- **H_1**: Algorithm and Outcome are dependent (performance depends on algorithm)
- A **significant p-value** suggests that the **type of algorithm affects the success rate**

6.3.3 Assumptions of the Chi-Squared Test

Assumption	Explanation
Expected counts ≥ 5	Each cell in the contingency table should have at least 5 expected values
Independent observations	Each outcome should be from a distinct trial or unit
Sufficient sample size	Chi-squared tests are not reliable for very small datasets

You can check expected counts directly:

```
chisq.test(table_data)$expected
```

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

```
##
##                Fail  Success
##   HeapSort  2.333333 27.66667
##   MergeSort 2.333333 27.66667
##   QuickSort 2.333333 27.66667
```

If **any expected count < 5**, consider: - **Combining categories** - Using **Fisher's Exact Test** (for 2×2 tables and small samples)

Practical Relevance for Research

Chi-squared tests are ideal when analyzing: - **Error types** by method - **Crash rate** by platform - **Pass/fail rates** by algorithm - **Survey preferences** by demographic

These tests give categorical results **statistical grounding**, enabling you to support claims about relationships and proportions with confidence.

6.4 Correlation Analysis

Correlation analysis is used to determine whether two numeric variables **move together** — and if so, how **strongly** and in **what direction**.

In computer science research, this is useful when investigating relationships like:

- Input size vs execution time
- Error rate vs number of iterations
- User rating vs completion time
- Memory usage vs frame rate

However, it's crucial to remember:
> **Correlation measures association, not cause and effect.**

6.4.1 Pearson Correlation Coefficient (r)

When to Use

- Both variables are **numeric** and **normally distributed**
- You suspect a **linear** relationship

What It Measures

- Strength and direction of a **linear** relationship
- Values range from -1 to 1

Value of r	Interpretation
+1	Perfect positive linear correlation
0	No linear correlation
-1	Perfect negative linear correlation

Example: Input Size vs Run Time

Let's simulate a relationship between input size and algorithm run-time:

```
set.seed(123)
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] 0.8185339
```

This computes the **Pearson correlation coefficient** between input size and QuickSort run time.

6.4.2 Spearman Rank Correlation (ρ)

When to Use

- Variables are **ordinal**, **non-normally distributed**, or **non-linear but monotonic**
- Data contains **outliers** that distort Pearson correlation

What It Measures

- Correlation of **ranks**, not raw values
- More robust to outliers and non-linearity

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

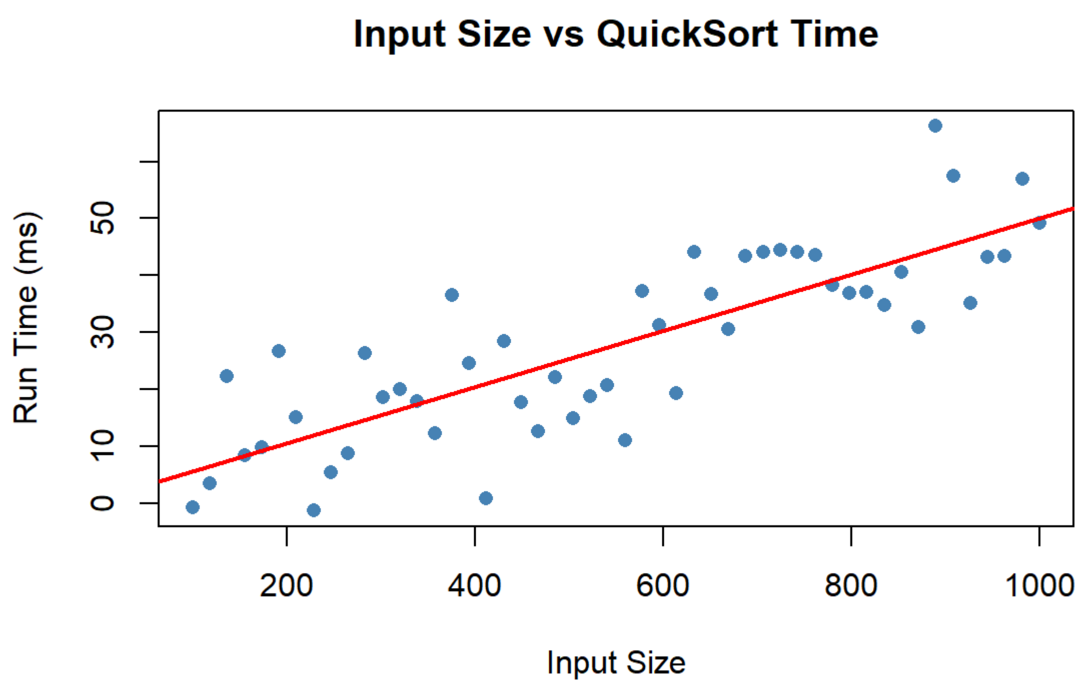
```
## [1] 0.8128211
```

6.4.3 Scatterplots and Visual Diagnostics

Always pair correlation with a **scatterplot**. This helps:

- Spot non-linear patterns
- Detect outliers
- Confirm linear or monotonic trends

```
plot(input_size, qs_runtime,
     main = "Input Size vs QuickSort Time",
     xlab = "Input Size", ylab = "Run Time (ms)",
     col = "steelblue", pch = 16)
abline(lm(qs_runtime ~ input_size), col = "red", lwd = 2)
```



You can also view relationships between multiple 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      0.82  0.13      0.98
## QuickSort 0.82      1.00  0.09      0.81
## Noise     0.13      0.09  1.00      0.16
## Quadratic 0.98      0.81  0.16      1.00
```

This produces a matrix of correlation values for multiple numeric variables.

6.4.4 Correlation ≠ Causation

> A high correlation between A and B **does not imply** that A causes B.

Correlation may arise because: - A causes B
- B causes A
- A and B are both affected by a third variable (confounding) - It's a coincidence (especially in small datasets)

Always interpret correlations **in context**, and support with theory or controlled experimentation.

Summary

Method	Use Case	Notes
Pearson	Numeric, linear, normal	Most common for numeric variables
Spearman	Ordinal or monotonic	Robust to outliers and non-linearity
Scatterplot	Visual confirmation	Shows shape and outliers
Correlation matrix	Many variables	Explore pairwise relationships

Correlation is a powerful **exploratory tool** — but not a substitute for **causal inference** or **experimental validation**.

6.5 Non-Parametric Alternatives

Not all datasets meet the **assumptions** required for parametric tests like t-tests and ANOVA. In particular:

- The data may be **non-normally distributed**
- There may be **outliers** or **skewness**
- Sample sizes may be **too small** to rely on the Central Limit Theorem
- The data may be **ordinal** or **ranked**, not continuous and numeric

In such situations, we turn to **non-parametric statistical tests**.

> These tests do **not assume normality** and are based on **ranks**, not actual values.

They are especially useful for: - Small sample sizes - Median comparisons - Ordinal (ranked) data - Robust testing in presence of outliers

6.5.1 Wilcoxon Signed-Rank Test (Paired Samples)

When to Use

- Like a **paired t-test**, but for **non-normal** or **ordinal** paired data
- Measures whether the **median difference** between paired observations is zero

Scenario

> You test QuickSort and MergeSort on **the same 30 datasets**, but suspect the differences aren't normally distributed.

```
set.seed(1)
qs_times <- rnorm(30, mean = 52, sd = 7)
ms_times <- rnorm(30, mean = 48, sd = 7)

# Apply Wilcoxon signed-rank test
wilcox.test(qs_times, ms_times, paired = TRUE)
```

```
##
##  Wilcoxon signed rank exact test
##
## data:  qs_times and ms_times
## V = 340, p-value = 0.02623
## alternative hypothesis: true location shift is not equal to 0
```

Interpretation

- **H_0**: The median difference between paired groups is zero
- **H_1**: The median difference is not zero
- If **$p < 0.05$**, conclude a significant difference in medians

This test is **robust to outliers**, unlike the paired t-test.

6.5.2 Mann–Whitney U Test (aka Wilcoxon Rank-Sum)

When to Use

- Like a **two-sample t-test**, but for **independent groups**
- Compares whether values in one group **tend to be higher or lower** than another

Scenario

> Compare MergeSort and QuickSort performance across **two independent datasets**, but the data appear skewed.

```
# Group labels
group <- rep(c("QuickSort", "MergeSort"), each = 30)
times <- c(qs_times, ms_times)

# Apply Mann-Whitney U test
wilcox.test(times ~ group)
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  times by group
## W = 276, p-value = 0.009604
## alternative hypothesis: true location shift is not equal to 0
```

Interpretation

- **H_0**: The distributions are equal
- **H_1**: One distribution tends to be greater than the other

This test **does not compare means** — it compares **ranks** and **medians**.

> Suitable when the scale is ordinal, or data is skewed/heavy-tailed.

6.5.3 Kruskal–Wallis Test (Non-Parametric ANOVA)

When to Use

- Like **one-way ANOVA**, but for 3+ **independent groups**
- Compares whether samples come from the same **rank distribution**

Scenario

> Compare QuickSort, MergeSort, and HeapSort on run-time data where distributions are **not normal**.

```
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 = 11.135, df = 2, p-value = 0.00382
```

Interpretation

- **H_0**: All groups come from the same population (same median ranks)
- **H_1**: At least one group differs in location (median)

If significant, you may follow up with **pairwise Wilcoxon tests** and **adjust for multiple comparisons**.

Summary of Non-Parametric Tests

Test	Parametric Equivalent	Use Case	Notes
Wilcoxon signed-rank	Paired t-test	Two related samples	Tests medians
Mann–Whitney U	Two-sample t-test	Two independent samples	Tests rank tendency
Kruskal–Wallis	One-way ANOVA	3+ independent groups	Tests rank distributions

Key Advantages

- Do **not assume normality**
- Robust to **outliers and skewed data**
- Work with **ordinal/ranked data**
- Ideal for **small samples** or **non-standard distributions**

Key Limitations

- Cannot easily model **interactions** (unlike 2-way ANOVA)
- Less powerful than parametric tests when assumptions **are** met
- Do **not test means** — they test **ranked medians or distributions**

When to Prefer Non-Parametric Tests

Use non-parametric tests when:

- Sample size is **small (n < 30)**
- Data is **not normally distributed** (and transformation isn't an option)
- Data contains **extreme outliers**
- Your variable is **ordinal** (e.g., Likert scale)

Choosing the Right Statistical Test

This decision matrix helps you identify the **most appropriate test** based on:

- Type of **data** (numeric or categorical)
- Number of **groups** being compared
- Whether the samples are **independent or paired**
- Whether the data meets **parametric assumptions** (e.g., normality)

Decision Matrix for Common Statistical Tests

Research Goal	Data Type	Groups	Parametric?	Test to Use
Compare sample mean to known value	Numeric	1 group	✓ Yes	**One-sample t-test**
Compare two means (independent)	Numeric	2 groups	✓ Yes	**Two-sample t-test**
Compare two means (paired)	Numeric	2 paired sets	✓ Yes	**Paired t-test**
Compare 3+ means (independent)	Numeric	3+ groups	✓ Yes	**One-way ANOVA**
Compare 3+ means with 2 factors	Numeric	3+ groups, 2 factors	✓ Yes	**Two-way ANOVA**
Compare two medians (independent)	Numeric or ordinal	2 groups	✗ No	**Mann–Whitney U test**

Research Goal	Data Type	Groups	Parametric?	Test to Use
Compare two medians (paired)	Numeric or ordinal	2 paired sets	❌ No	**Wilcoxon signed-rank test**
Compare 3+ medians (independent)	Numeric or ordinal	3+ groups	❌ No	**Kruskal–Wallis test**
Compare observed vs expected counts	Categorical	1 variable	N/A	**Chi-squared goodness-of-fit**
Test association between two categorical vars	Categorical	2 variables	N/A	**Chi-squared test of independence**
Test correlation between 2 numeric vars	Numeric	2 variables	✓ Linear	**Pearson correlation**
Test rank-based correlation	Ordinal or non-linear	2 variables	❌ No	**Spearman correlation**

How to Use This Matrix

1. **Identify your research question**
   → What are you trying to compare or test?
2. **Classify your data type**
   → Is it numeric (continuous), ordinal, or categorical?
3. **Count the groups or variables**
   → One sample? Two groups? Multiple factors?
4. **Check your assumptions**
   → Is your data normal? Are variances equal?
5. **Select the test**
   → Use the matrix above to choose the appropriate test