**Data Types**

* **Categorical**
  + **Dichotomous**
  + **Ordinal**
  + **Nominal**
* **Numerical**
  + **Continuous**
  + **Interval**

**Categorical data** represents data characteristics or qualitative descriptions. Generally, categorical data is any data that is not measured, also known as qualitative data. Categorical data can be collected in the form of strings, true/false Boolean values, or even encoded numbers as categories (such as one for red, two for blue, three for green, etc.). Several statistical tests use categorical data to inform which groups to compare. Categorical data has three subtypes: dichotomous, ordinal, and nominal

**Dichotomous data** is collected from either one of two categories. For example, an online survey might collect member/non-member or demographic information. Dichotomous data can be collected in the form of true/false Boolean values, 0 or 1 binary values, or two strings. Later in the module, we'll use dichotomous data to help perform many of our comparative statistical tests.

**Ordinal data** has a ranked order. Although ordinal data has a sequence, we don't necessarily know the value between each ordinal data point. Data that is collected on a value scale (e.g., movie rankings, survey results, and the [Likert scale (Links to an external site.)](https://en.wikipedia.org/wiki/Likert_scale)) are common forms of ordinal data. Ordinal data combines the qualitative properties of labels to the quantitative properties of scale to allow for comparative analyses. Ordinal data is very popular with research and survey groups because it allows for quantitative analysis without the need of machinery and tools to obtain measurements.

**NOTE**

There are statistical tests such as the Mann-Whitney U test and the Kruskal-Wallis H test that compare ordinal datasets. These statistical tests are more advanced versions of basic comparative statistics tests and are outside the scope of this course. However, once you master the basics of statistical testing, it is not difficult to apply more advanced statistical models based on your specific data needs. Remember—Google is your best friend!

**Nominal data** is data used as labels or names for other measures. Nominal data can be as individual as an identification number or can be as general as a list of three options. Unlike ordinal data, nominal data has no ranking. Therefore, nominal data is often used with a more quantitative data type to perform an analysis. Often nominal data will be transformed using a grouping function to decrease the complexity of the data.

**Numerical Data**

Typically, **numerical data** is obtained by taking a measurement from an instrument (such as a ruler, measuring scale, sensor, etc.) or by counting. In statistics, numerical data is used to perform quantitative analysis that can produce the probability of an outcome or quantify the relationship between categories. Within numerical data there are two primary data types to consider: continuous and interval.

**Continuous data** can be subdivided infinitely. For example, if you want to describe the thickness of window glass, you could measure it in x number of centimeters, millimeters, nanometers, picometers, and so on. Continuous data is typically recorded with decimal places to match the precision of the measurement. Almost all statistical tests and models use continuous data to generate precise results.

**Interval data** is spaced out evenly on a scale. Also known as integer data, interval data does not use decimal places and can't be subdivided. Interval data also can't be multiplied or divided. Because interval data is spaced out evenly, it can be grouped together or bucketed easily. For example, a set of integers 15, 4, 18, 10, 3 , and 5 could be collected as a group that is less than 20. Due to this property, interval data can be treated as a numerical data type or transformed into a nominal data type.

Additionally, interval data can be generated through rounding continuous data at the cost of losing precision of the measurement. Therefore, interval data can be used by most statistical models as either a quantitative or qualitative variable, depending on the use case.

Now that we understand the different statistical data types and how to identify each by its characteristics, we should be able to classify any tabular dataset.

**Getting Oriented with Data**

The easiest means of orienting ourselves on a new dataset in R is to use the head()function, which shows us the first few rows of our data frame. At any point when looking at the first few rows, we can use bracket notation (or the $ operator) to select an individual column to explore.

Alternatively, if we're using RStudio, we can explore any data frame by clicking on it in our environment pane. By navigating through each column and classifying each data type, we can determine which columns provide measurement results, and which columns provide characteristics about our subjects.

If we're fortunate to have context provided for a given dataset via documentation or from the data collector, we should be able to identify columns and metrics of interest. However, we have not finished characterizing our data just yet—we still need to understand how values in our data are distributed.

**Skew**

When dealing with relatively smaller sample sizes, our data distributions are often asymmetrical. Compared to the normal distribution, where each tail of the distribution (on either side of the mean μ) mirrors one another, the asymmetrical distribution has one distribution tail that is longer than the other. This asymmetrical distribution is commonly referred to as a **skewed distribution** and there are two types—left skew and right skew.

**Left Skew**

A data distribution is considered to be **left skewed,** or negative skewed, if the left tail is longer than the right, as shown below.

When data is skewed left, from the center of the distribution curve, there is a higher probability that extreme negative values exist within our dataset. When this occurs, the mean may no longer accurately reflect the central tendency of the data. Instead, we would use the median to describe the central tendency of the data. This skew is called negative skewed.

**Right Skew**

A data distribution is considered to be **right skewed**, or positive skewed, if the right tail is longer than the left, as shown below.

When data is skewed right, from the center of the distribution curve, there is a higher probability that extreme positive values exist within our dataset. Once again, if this occurs, we would use the median to describe the central tendency of the data. This skew is called positive skewed.

**Manage Skewness**

As with most problems in data analytics, we must approach skewness on a case-by-case basis. Depending on the severity of the skewness and the size of the dataset, there are multiple means of dealing (or not dealing) with skewness.

If our dataset is large, or the skewness is very subtle, we would simply point out that our data distribution shows signs of skew during reporting or presentation. In these cases, our mean and median will be roughly the same value, and there should be minimal impact to any downstream analysis.

If our dataset is smaller, or the skewness does impact the overall shape of our distribution, more action is needed. There are a few different things we can try:

* If possible, add more data points to our dataset to alleviate the effect of skew. However, this might not be possible or might not improve the distribution.
* Resample or regenerate data if we think that the data might not be representative of the original conditions or dataset.
* Transform our data values by normalization, using another numerical variable, or by transforming the data using an operator. The concept of transforming skewed data is very popular with scientists who deal with datasets where values can differ by orders of magnitude. One of the easiest means of transforming data is using a log-transform, where each value in the numeric dataset is transformed taking either natural log, or log10. By using a log-transformation, the effects of extreme values are reduced, and this transformation can help make each distribution tail more symmetrical.

**IMPORTANT**

No matter what approach is used to help reduce the skewness of a dataset, it's good practice to disclose this information in a report or to use annotations on your results. This will help the reader understand the context surrounding any results, and it will make your analysis more credible.

Hypothesis Testing

One of the largest and most critical concepts in statistics is hypothesis testing. In data science, we use statistical hypothesis testing to determine the probability of an event (or set of observations) under particular assumptions. In other words, we use statistical hypothesis testing to determine which of our hypotheses are most likely to be true. There are two types of statistical hypothesis:

* The **null hypothesis** is also known as H0 and is generally the hypothesis that can be explained by random chance.
* The **alternate hypothesis** is also known as Ha and is generally the hypothesis that is influenced by non-random events.

By the end of this section, we should be able to generate a set of hypotheses and interpret the outcome of a statistical test. These concepts are universal and will apply to any statistical test, dataset, or analytical result.

## The Importance of Hypothesis Testing

Although data collection and research are important, the backbone of the scientific method is **hypothesis testing**. Hypotheses are utilized by the scientific method to help narrow the scope of research and testing as well as provide a clear outcome of our results. Without generating a set of hypotheses, it becomes exponentially more difficult to quantify results and provide measurable outcomes to our analyses. As data analysts, it's our job to match a set of hypotheses to an appropriate statistical test to ensure that results are interpreted correctly.

## Hypothesis Testing in Five Steps

Regardless of the complexity of the dataset or the proposed question, hypothesis testing uses the same five steps:

1. Generate a null hypothesis, its corresponding alternate hypothesis, and the significance level.
2. Identify a statistical analysis to assess the truth of the null hypothesis.
3. Compute the p-value using statistical analysis.
4. Compare p-value to the significance level.
5. Reject (or fail to reject) the null hypothesis and generate the conclusion.

Keep in mind that the null and alternate hypotheses are used to explain one of two outcomes from our proposed question, and both are mutually exclusive and exhaustive. In other words, no matter what, one of these statements must be used to explain our analysis results.

For example, perhaps we wanted to solve the question: "Is flipping a specific coin fair and balanced?" Given this question, our null hypothesis could be that the likelihood of flipping heads is the same as flipping tails. In other words, the likelihood of heads or tails can be totally explained by random chance. Our alternative hypothesis might be that the likelihood of flipping heads is not the same as flipping tails. If we were to represent our hypotheses using mathematical symbols, it would be expressed as:

H0 : PH = 0.5

Ha : PH ≠ 0.5

Where PH represents the probability of flipping heads.

**IMPORTANT**

Notice that our null hypothesis represents the scenario that our results can be explained by random chance without any outside influence. In contrast, our alternate hypothesis represents any other scenario that our results could yield.

Once we established our null and alternate hypothesis, we would then need to identify a statistical analysis to assess if our null hypothesis is true. In this example, we could test our null hypothesis by flipping our own coin 50 times and then by calculating the probability of flipping heads.

**IMPORTANT**

Depending on the complexity of the question and null hypothesis, calculating probability might be insufficient, and we might need to rely on more traditional statistical tests. Fret not, we'll cover many of the most popular statistical tests later in this module.

After determining which statistical analysis is most appropriate and analyzing our data, we must quantify our statistical results using probability. In our example, we are calculating probability directly; however, most statistical tests will produce probability in the form of a p-value.

The **p-value**, or probability value, tells us the likelihood that we would see similar results if we tested our data again, if the null hypothesis is true. Therefore, we use the p-value to provide quantitative evidence as to which of our hypotheses are true.

To determine which hypothesis is most likely to be true, we compare the p-value against a significance level. A **significance level** (also denoted as alpha or ɑ) is a predetermined cutoff for our hypothesis test. When designing our hypothesis, we would determine the significance level based on the importance of our findings.

In most cases, a significance level of 0.05 is sufficient, but if our hypotheses are being used for critical decision-making (such as the performance of a drug or the durability of a helmet), we might want to use smaller cutoffs such as 0.01 or 0.001. Regardless of what significance level we select, we want to predetermine our cutoff prior to computing the p-value as to not introduce bias into our results. Refer to the following chart:

|  |  |  |
| --- | --- | --- |
| **Importance of Findings** | **Significance Level** | **Probability of Being Wrong** |
| Low | 0.1 | 1 in 10 |
| Normal | 0.05 | 5 in 100 |
| High | 0.01 | 1 in 100 |
| Very high | 0.001 | 1 in 1,000 |
| Extreme | 0.0001 | 1 in 10,000 |

In addition to determining a significance level based on the importance of our findings, we must ensure our hypotheses and statistical tests are either one-tailed or two-tailed. The tails of our hypotheses or statistical tests are referring to the distribution of measurements or observations used in the analysis.

When it comes to hypothesis testing, a **one-tailed hypothesis** is only describing one side of the distribution. One-tailed hypotheses use descriptions such as "x is greater than y" or "x is less than or equal to y." Alternatively, **two-tailed hypotheses** describe both sides of the distribution and use descriptions such as "equal to" or "not equal to."

When it comes to checking our statistical tests, the documentation will tell us if our statistical test is one-tailed or two-tailed. Once we have determined the number of tails considered for both our hypotheses and statistical test, we can determine if we need to adjust our p-value:

* If our hypotheses and statistical test are both two-tailed, use the statistical test p-value as is.
* If our hypotheses are one-tailed, but our statistical test is two-tailed, divide the statistical test p-value by 2.

Once we have determined the significance level and the calculated p-value, we can complete our statistical analysis. If our calculated p-value is smaller than our significance level, we would state that there is sufficient statistical evidence that our null hypothesis is not true, and therefore we would reject our null hypothesis. Alternatively, if our calculated p-value is larger than our significance level, we would state that we do not have sufficient evidence to reject our null hypothesis, and therefore we fail to reject our null hypothesis.

**NOTE**

Outside of statistics, you may see others refer to this process as "accepting or rejecting" the null hypothesis. While this is not entirely untrue, many statisticians believe that there is no way of making a definitive choice between either hypothesis without an infinitely large dataset. Therefore, we use p-values and significance levels to determine the probability that our observations were obtained assuming the null hypothesis.

After we have determined which hypothesis is most likely to be true, we must conclude our statistical findings by relating our results back to the initial dataset or proposed question.

[Previous](https://courses.bootcampspot.com/courses/482/modules/items/125382)[Next](https://courses.bootcampspot.com/courses/482/modules/items/125397)

Errors

In an ideal world, we would be able to definitively decide if our null hypothesis or alternative hypothesis was true by collecting all possible measurements or data points. But since this is often impossible, we must approximate truth using a subset of data.

In most cases, our approximations and hypothesis selection will be correct and represent the true real-world results. But due to the variability of data, at some point our hypothesis selection will be incorrect. Our incorrect hypothesis selection can fall into two categories:

* **Type I error** (also known as a **false positive** error)—an error in which we reject the null hypothesis when it is actually true. In other words, the observations and measurements used in our statistical test should have been attributed to random chance, but we attributed them to something else.
* **Type II error** (also known as a **false negative** error)—an error in which we fail to reject the null hypothesis when it is actually false. In other words, our analysis demonstrates that the observations were due to random chance, but they were not. The observations and measurements used in our statistical test failed to reflect an external force or influence to our problem.

While selecting the wrong hypothesis is never ideal, depending on our field of research or the importance of our proposed problem, one error type may be more problematic than the other.

When it comes to limiting our type I and type II errors, there are two basic methods:

* A **type I error** can be limited by making your significance level smaller. A smaller significance level makes it harder to accidentally reject the null hypothesis when the data was truly random. This is also why our significance level (alpha or ɑ) is sometimes referred to as our false positive rate.
* A **type II error** can be limited by increasing the power of the analysis. Although performing a power analysis is outside the scope of the course, power can be increased by adding additional measurements or observations to our analysis.

Now that we understand what a hypothesis test is, how to generate our hypothesis, and how to use the p-value to provide results and interpretations, we're ready to dive into our first statistical test.

You are in charge of testing whether or not there are statistical differences between two manufacturing lots of brake pads. Which type of statistical error would be considered more problematic?

* 

Type I error

Type I error

* 

Type II error

Type II error

Correct. Nice work! If there were statistical differences between lots that were not identified, there is a potential risk of injury or death to the consumer.

[Skip to navigation](https://learnosityplayer.atomicjoltapps.com/lti_launches/BvwEYKqNaNb8hZL3E5ti9xbq#lrn_assess_next_btn)

### **Question**

You must test whether or not the new company logo significantly increases monthly profit. Which type of statistical error would be considered more problematic?

* 

Type I error

Type I error

* 

Type II error

Type II error

Correct. Nice work! If the company changes the logo and profits do not change (or even decrease), this mistake could cost the company a substantial amount of money.

[Skip to navigation](https://learnosityplayer.atomicjoltapps.com/lti_launches/BvwEYKqNaNb8hZL3E5ti9xbq#lrn_assess_next_btn)

Finish

Sample vs pop

In data analysis and statistics, an ideal dataset is one that contains measurements and results from every possible outcome, condition, or consideration. These datasets are known as a **population** **dataset** and contain all possible elements of a study or experiment.

Often, such an exhaustive dataset is prohibitively expensive or logistically impossible to generate. In this case, we must use a **sample** or subset of the population dataset, where not all elements of a study or experiment are collected or measured.

**CAUTION**

In data science, the concept of sample versus population does not strictly apply to people or animals. Any comprehensive dataset is considered a population, and any dataset that is a subset of a larger dataset is considered a sample.

Since a sample dataset is just that—a sample—we must be clear about how a sample dataset represents the corresponding population data. One of the most straightforward ways to characterize a sample versus its population data is to compare the mean and standard deviation of both datasets. Ideally, a sample dataset will have a similar distribution to the population data, and therefore the mean and standard deviation would be about equal.

To produce a sample dataset that has a similar distribution to the population data, most statisticians suggest using random sampling. **Random sampling** is a technique in data science in which every subject or data point has an equal chance of being included in the sample. This technique increases the likelihood that even a small sample size will include individuals from each "group" within the population.

If performed using functions such as the built-in sample()function in R, or sample\_n() function from dplyr, the resulting sample distributions should be similar to the input population data. When selecting sample data from a numerical vector, we should use the built-in sample() function. However, in most cases we will want to use the sample\_n() function to select sample data from a two-dimensional data object.

Type the following code into the R console to look at the sample\_n() documentation in the Help pane, listed under the subhead "Usage" in the image below:

>?sample\_n()

Using the sample\_n()function only requires two arguments:

* **tbl** is the name of the input table, which is typically the name of a data frame. Optionally, we can use a dplyr pipe (%>%) to provide the data frame object directly, in which case, this argument is optional.
* **size** is the number of rows to return. As noted in the documentation, if we are providing a data frame that was grouped using the group\_by()function, the **size** argument is the number of groups to return.

To practice generating samples using random sampling, download the [used vehicle dataset (Links to an external site.)](https://2u-data-curriculum-team.s3.amazonaws.com/dataviz-online/module_15/used_car_data.csv) dataset contains market data on more than 300 used vehicles. If we want to visualize the distribution of driven miles for our entire population dataset, we can use the geom\_density()function from ggplot2:

> population\_table <- read.csv('used\_car\_data.csv',check.names = F,stringsAsFactors = F) #import used car dataset

> plt <- ggplot(population\_table,aes(x=log10(Miles\_Driven))) #import dataset into ggplot2

> plt + geom\_density() #visualize distribution using density plot

**IMPORTANT**

In this example, we want to transform our miles driven using a log10 transformation. This is because the distribution of raw mileage is right skewed—a few used vehicles have more than 100,000 miles, while the majority of used vehicles have less than 50,000 miles. The log10 transformation makes our mileage data more normal.

Now that we characterized our population data using our density plot, we'll create a sample dataset using dplyr's sample\_n()function. Type the following code in the R console:

> sample\_table <- population\_table %>% sample\_n(50) #randomly sample 50 data points

> plt <- ggplot(sample\_table,aes(x=log10(Miles\_Driven))) #import dataset into ggplot2

> plt + geom\_density() #visualize distribution using density plot

By using dplyr's sample\_n() function, we can create a random sample dataset from our population data that contains minimal bias and (ideally) represents the population data.

Depending on the size of the population data, we may need to also adjust the size argument in our sample\_n() function to ensure that our sample data is representative of the underlying population data. There are two basic ways to check that our sample data is representative of the underlying population: a qualitative assessment of each density plot or a quantitative statistical test such as the one-sample t-test.

One-sided T-test

The **Student's t-test** (most commonly referred to as **t-test**) is one of the most basic and popular statistical tests in the world. In statistics, we use a t-test to compare the mean of one dataset to another under a few assumptions.

There are two main forms of the t-test that we use: the **one-sample t-test** and the **two-sample t-test**. The one-sample t-test is used to determine whether there is a statistical difference between the means of a sample dataset and a hypothesized, potential population dataset. In other words, a one-sample t-test is used to test the following hypotheses:

* H0 : There is **no statistical difference** between the observed sample mean and its presumed population mean.
* Ha : There is **a statistical difference** between the observed sample mean and its presumed population mean.

**NOTE**

We can also use a one-sided t-test by changing our alternative hypothesis to state that our sample mean is **significantly less** or **significantly more** than our presumed population mean.

Before we can apply any statistical test to our data, we must check if there are any assumptions regarding our input dataset. When it comes to our one-sample t-test there are five assumptions about our input data:

1. The input data is numerical and continuous. This is because we are testing the distribution of two datasets.
2. The sample data was selected randomly from its population data.
3. The input data is considered to be normally distributed.
4. The sample size is reasonably large. Generally speaking, this means that the sample data distribution should be similar to its population data distribution.
5. The variance of the input data should be very similar.

As long as our input data satisfies (or mostly satisfies) the above assumptions, we can use the one-sample t-test to assert the similarities or differences in our data.

In R, we can implement a one-sample t-test using the built-in stats package t.test()function. Type the following code into the R console to look at the t.test()documentation in the Help pane:

> ?t.test()

To use the t.test()function to perform our one-sample t-test, we have to use a few arguments:

* **x**
* **mu**
* **alternative**

By setting all three of these arguments, the t.test()function should produce our test statistic "t" along with our p-value, which we can use to evaluate our null hypothesis.

For example, if we want to test if the miles driven from our previous sample dataset is statistically different from the miles driven in our population data, we would use our t.test()function as follows:

>t.test(log10(sample\_table$Miles\_Driven),mu=mean(log10(population\_table$Miles\_Driven))) #compare sample versus population means

There are a number of metrics produced from the t.test()function, but for now we will only concern ourselves with the calculated p-value. Assuming our significance level was the common 0.05 percent, our p-value is above our significance level. Therefore, we do not have sufficient evidence to reject the null hypothesis, and we would state that the two means are statistically similar.

**IMPORTANT**

Due to random sampling, your sample dataset may differ from our example and thus your calculations may be different. Therefore, you'll need to compare your calculated p-value to your own significance level. If your p-value is lower than the significance level, you would have sufficient evidence to reject the null hypothesis and state that the two means are statistically different.

Looking at the R documentation for the t.test() function, match the following arguments to their corresponding descriptions:

Put responses in the correct input to answer the question. Select a response, navigate to the desired input and insert the response. Responses can be selected and inserted using the space bar, enter key, left mouse button or touchpad. Responses can also be moved by dragging with a mouse.

1. x is the numeric vector of sample data.
2. mu is the calculated mean of the population data.
3. alternative tells the t.test() function if the hypothesis is one-sided (one-tailed) or two-sided (two-tailed). The options for the alternative argument are “two.sided,” “less,” or “greater.” By default, the t.test() function assumes a two-sided t-test.

2-sided t-test

The second main form of the t-Test is a two-sample t-Test. Instead of testing whether a sample mean is statistically different from its population mean, the two-sample t-Test determines whether the means of two samples are statistically different. In other words, a two-sample t-Test is used to test the following hypotheses:

* H0 : There is **no statistical difference** between the two observed sample means.
* Ha : There is **a statistical difference** between the two observed sample means.

There are also five assumptions regarding our input data when using the two-sample t-Test, which are the same as the one-sample t-Test:

1. The input data is numerical and continuous.
2. Each sample data was selected randomly from the population data.
3. The input data is considered to be normally distributed.
4. Each sample size is reasonably large. Generally speaking, this means that the sample data distribution should be similar to its population data distribution.
5. The variance of the input data should be very similar.

In R, we use the same t.test() function to calculate both a one-sample t-Test and two-sample t-Test. However, the two-sample t-Test arguments are slightly different:

* ***x*** is the first numeric vector of sample data.
* ***y*** is the second numeric vector of sample data.
* **alternative** tells the t.test() function if the hypothesis is one-sided (one-tailed) or two-sided (two-tailed). The options for the alternative argument are "two.sided,""less," or "greater." By default, the t.test() function assumes a two-sided t-Test.

Once we have provided the necessary numeric vectors for each sample, the t.test() function will calculate our two-sample t-Test and return the same output as before. As practice, let's test whether the mean miles driven of two samples from our used car dataset are statistically different.

First, we produce our two samples using the following R statements:

> sample\_table <- population\_table %>% sample\_n(50) #generate 50 randomly sampled data points

> sample\_table2 <- population\_table %>% sample\_n(50) #generate another 50 randomly sampled data points

Because our samples should not contain bias, we would expect our null hypothesis to be true—our samples should not be statistically different. To confirm, we'll use the t.test() function as follows:

> t.test(log10(sample\_table$Miles\_Driven),log10(sample\_table2$Miles\_Driven)) #compare means of two samples

Anova

When dealing with large real-world numerical data, we're often interested in comparing the means across more than two samples or groups. The most straightforward way to do this is to use the **analysis of variance (ANOVA) test,** which is used to compare the means of a continuous numerical variable across a number of groups (or factors in R).

Depending on your dataset and questions you wish to answer, an ANOVA can be used in multiple ways. For the purposes of this course, we'll concentrate on two different types of ANOVA tests:

* A **one-way ANOVA** is used to test the means of a single dependent variable across a single independent variable with multiple groups. (e.g., fuel efficiency of different cars based on vehicle class).
* A **two-way ANOVA** does the same thing, but for two different independent variables (e.g., vehicle braking distance based on weather conditions and transmission type).

Regardless of whichever type of ANOVA test we use, the statistical hypotheses of an ANOVA test are the same:

H0 : The means of all groups are equal, or µ1 = µ2 = … = µn.

Ha : At least one of the means is different from all other groups.

Additionally, both ANOVA tests have assumptions about the input data that must be validated prior to using the statistical test:

1. The dependent variable is numerical and continuous, and the independent variables are categorical.
2. The dependent variable is considered to be normally distributed.
3. The variance among each group should be very similar.

In R, we can use the aov() function to perform both the one-way and two-way ANOVA test. Type the following code into the R console to look at the aov() documentation in the Help pane:

>?aov()

To perform an ANOVA test in R, we have to provide the aov()function two arguments:

* **formula**
* **data**

Unlike the t.test() function, where each group was a separate numeric vector, the aov() function expects that all of the observations and grouping information are contained within a single data frame. Once we have our cleaned and labeled data frame, we're ready to perform our ANOVA test using the aov()function.

To practice our one-way ANOVA, return to the mtcars dataset. For this statistical test, we'll answer the question, "Is there any statistical difference in the horsepower of a vehicle based on its engine type?"

In this case, we will use the "hp" and "cyl" columns from our mtcars dataset:

* horsepower (the "hp" column) will be our dependent, measured variable
* number of cylinders (the "cyl" column) will be our independent, categorical variable.

However, in the mtcars dataset, the cyl is considered a numerical interval vector, not a categorical vector. Therefore, we must clean our data before we begin, using the following code:

> mtcars\_filt <- mtcars[,c("hp","cyl")] #filter columns from mtcars dataset

> mtcars\_filt$cyl <- factor(mtcars\_filt$cyl) #convert numeric column to factor

Now that we have our cleaned dataset, we can use our aov()function as follows:

> aov(hp ~ cyl,data=mtcars\_filt) #compare means across multiple levels

Due to the fact that the ANOVA model is used in many forms, the initial output of our aov() function does not contain our p-values. To retrieve our p-values, we have to wrap our aov()function in a summary() function as follows:

> summary(aov(hp ~ cyl,data=mtcars\_filt))

When using the formula statement, each independent variable will be shown as a separate row, with an additional "Residuals" row that tells us what the residual error is for our ANOVA model. For our purposes, we are only concerned with the "Pr(>F)" column, which is the same as our p-value statistic.

Depending on how small our p-value is, there may be symbols on the right side that indicate which significance level the p-value is below. In this case, our p-value is 1.32 ✕ 10^-8^, which is much smaller than our assumed 0.05 percent significance level. Therefore, we would state that there is sufficient evidence to reject the null hypothesis and accept that there is a significant difference in horsepower between at least one engine type and the others.

Now that you have learned the differences between our t-tests and ANOVA tests, you're ready to analyze data and perform statistical tests when comparing means. Feel free to explore more datasets and practice implementing analysis of means on your own.

Correlation

In data analytics, we'll often ask the question "is there any relationship between variable A and variable B?" This concept is known in statistics as correlation. **Correlation analysis** is a statistical technique that identifies how strongly (or weakly) two variables are related.

Correlation is quantified by calculating a **correlation coefficient**, and the most common correlation coefficient is the Pearson correlation coefficient. The **Pearson correlation coefficient** is denoted as "r" in mathematics and is used to quantify a linear relationship between two numeric variables. The Pearson correlation coefficient ranges between -1 and 1, depending on the direction of the linear relationship.

The following image is an example of an **ideal positive correlation** where r = 1. When two variables are positively correlated, they move in the same direction. In other words, when the variable on the x-axis increases, the variable on the y-axis increases as well:

The following image is an example of an **ideal negative correlation** where r = -1. When two variables are negatively correlated, they move in opposite directions. In other words, when the variable on the x-axis increases, the variable on the y-axis decreases.

The following image is an example of two variables with **no correlation** where r ≈ 0. When two variables are not correlated, their values are completely independent between one another.

For real-world data, it can be very difficult to determine if two variables are correlated, so we must use the Pearson correlation coefficient to calculate the correlation strength. Refer to the table below.

|  |  |
| --- | --- |
| Absolute Value of r | Strength of Correlation |
| r < 0.3 | None or very weak |
| 0.3 ≤ r < 0.5 | Weak |
| 0.5 ≤ r < 0.7 | Moderate |
| r ≥ 0.7 | Strong |

In R, we can use our geom\_point() plotting function combined with the cor() function to quantify the correlation between variables. Type the following code into the R console to look at the cor() documentation in the Help pane:

>?cor()

To use the cor() function to perform a correlation analysis between two numeric variables, we need to provide the following arguments:

* **x** is the first variable, which would be plotted on the x-axis.
* **y** is the second variable, which would be plotted on the y-axis.

As long as we are using two numeric variables, there are no other assumptions regarding our input data. To practice calculating the Pearson correlation coefficient, we'll use the mtcars dataset. Type the following in the R console:

> head(mtcars)

In the mtcars dataset, there are a number of numeric columns that we can use to test for correlation such as mpg, disp, hp, drat, wt, and qsec. For our example, we'll test whether or not horsepower (hp) is correlated with quarter-mile race time (qsec).

First, let's plot our two variables using the geom\_point() function as follows:

> plt <- ggplot(mtcars,aes(x=hp,y=qsec)) #import dataset into ggplot2

> plt + geom\_point() #create scatter plot

Looking at our plot, it appears that the quarter-mile time is negatively correlated with horsepower. In other words, as vehicle horsepower increases, vehicle quarter-mile time decreases.

Next, we'll use our cor() function to quantify the strength of the correlation between our two variables:

> cor(mtcars$hp,mtcars$qsec) #calculate correlation coefficient

From our correlation analysis, we have determined that the r-value between horsepower and quarter-mile time is -0.71, which is a strong negative correlation.

For another example, let's reuse our used\_cars dataset:

> used\_cars <- read.csv('used\_car\_data.csv',stringsAsFactors = F) #read in dataset

> head(used\_cars)

For this example, we'll test whether or not vehicle miles driven and selling price are correlated. Once again, we'll plot our two variables using the geom\_point() function:

> plt <- ggplot(used\_cars,aes(x=Miles\_Driven,y=Selling\_Price)) #import dataset into ggplot2

> plt + geom\_point() #create a scatter plot

Compared to our previous example, our scatter plot did not help us determine whether or not our two variables are correlated. However, let's see what happens if we calculate the Pearson correlation coefficient using the cor() function:

> cor(used\_cars$Miles\_Driven,used\_cars$Selling\_Price) #calculate correlation coefficient

Our calculated r-value is 0.02, which means that there is a negligible correlation between miles driven and selling price in this dataset.

In most cases, we'll use correlation analysis as a means of exploring data and looking for trends. Although we can calculate the correlation of each pair of numerical variables in a dataset, this process can be highly time-consuming.

Instead of computing each pairwise correlation, we can use the cor() function to produce a correlation matrix. A **correlation matrix** is a lookup table where the variable names of a data frame are stored as rows and columns, and the intersection of each variable is the corresponding Pearson correlation coefficient. We can use the cor() function to produce a correlation matrix by providing a matrix of numeric vectors.

For example, if we want to produce a correlation matrix for our used\_cars dataset, we would first need to select our numeric columns from our data frame and convert to a matrix. Then we can provide our numeric matrix to the cor() function as follows:

> used\_matrix <- as.matrix(used\_cars[,c("Selling\_Price","Present\_Price","Miles\_Driven")]) #convert data frame into numeric matrix

> cor(used\_matrix)

If we look at the correlation matrix using either rows or columns, we can identify pairs of variables with strong correlation (such as selling price versus present price), or no correlation (like our previous example of miles driven versus selling price).

The correlation matrix is a very powerful data exploration tool that allows an analyst to scan large numerical datasets for variables of interest. Once the variables of interest have been identified, the analyst can move on to more rigorous data analysis and hypothesis testing.

Linear regression

Earlier we learned about linear regression and how it can be used to determine our dependent *y* variable from an independent *x* variable. In a more formal definition, **linear regression** is a statistical model that is used to predict a continuous dependent variable based on one or more independent variables fitted to the equation of a line.

In school, most students learned that the equation of a line is written as *y* = *mx* + *b*:

The job of a linear regression analysis is to calculate the slope and *y* intercept values (also known as coefficients) that minimize the overall distance between each data point from the linear model. There are two basic types of linear regression:

* **Simple linear regression** builds a linear regression model with one independent variable.
* **Multiple linear regression** builds a linear regression model with two or more independent variables.

Linear regression is popular in data science because it has multiple applications. First and foremost, linear regression can be used as a predictive modeling tool where future observations and measurements can be predicted and extrapolated from a linear model. Linear regression can also be used as an exploratory tool to quantify and measure the variability of two correlated variables.

A good linear regression model should approximate most data points accurately if two variables are strongly correlated. In other words, linear regression can be used as an extension of correlation analysis. In contrast to correlation analysis, which asks whether a relationship exists between variables A and B, linear regression asks if we can predict values for variable A using a linear model and values from variable B.

To answer this question, linear regression tests the following hypotheses:

H0 : The slope of the linear model is zero, or m = 0

Ha : The slope of the linear model is not zero, or m ≠ 0

If there is no significant linear relationship, each dependent value would be determined by random chance and error. Therefore, our linear model would be a flat line with a slope of 0.

To quantify how well our linear model can be used to predict future observations, our linear regression functions will calculate an r-squared value. The **r-squared (r2) value** is also known as the coefficient of determination and represents how well the regression model approximates real-world data points. In most cases, the r-squared value will range between 0 and 1 and can be used as a probability metric to determine the likelihood that future data points will fit the linear model.

When using a simple linear regression model, the r-squared metric can be approximated by calculating the square of the Pearson correlation coefficient between the two variables of interest.

By combining the p-value of our hypothesis test with the r-squared value, the linear regression model becomes a powerful statistics tool that both quantifies a relationship between variables and provides a meaningful model to be used in any decision-making process.

Although the interpretation of a simple linear regression is different from a multiple linear regression, their model implementation is the same. In R, we'll build our linear models using the built-in lm()function. Type the following code into the R console to look at the lm() documentation in the Help pane:

>?lm()

Even though there are many optional arguments for the lm()function, the lm() function only requires us to provide two arguments:

* **formula**
* **data**

Similar to our t-test analysis, there are a few assumptions about our input data that must be met before we perform our statistical analysis:

1. The input data is numerical and continuous.
2. The input data should follow a linear pattern.
3. There is variability in the independent *x* variable*.* This means that there must be more than one observation in the x-axis and they must be different values.
4. The residual error (the distance from each data point to the line) should be normally distributed.

**IMPORTANT**

Validating the fourth assumption is outside the scope of this course as it involves more robust statistical methods. However, in most real-world cases, we can expect our data to meet the fourth assumption.

Once we have our data in a single data frame that meets the assumptions of our linear regression analysis, we're ready to implement the lm() function.

For practice, let's revisit our correlation example using the mtcars dataset. Using our simple linear regression model, we'll test whether or not quarter-mile race time (qsec) can be predicted using a linear model and horsepower (hp).

Remember from our correlation example that our Pearson correlation coefficient's r-value was -0.71, which means there is a strong negative correlation between our variables. Therefore, we anticipate that the linear model will perform well.

To create a linear regression model, our R statement would be as follows:

> lm(qsec ~ hp,mtcars) #create linear model

The output of the lm() function will be the metrics from our model. Specifically, the lm() function returns our y intercept (Intercept) and slope (hp) coefficients. Therefore, the linear regression model for our dataset would be qsec = -0.02hp + 20.56.

To determine our p-value and our r-squared value for a simple linear regression model, we'll use the summary() function:

> summary(lm(qsec~hp,mtcars)) #summarize linear model

Although there are a number of quantitative metrics produced by the summary(lm()) function, we are only concerned with the r-squared and p-value metrics at the bottom of the output.

From our linear regression model, the r-squared value is 0.50, which means that roughly 50% of all quarter mile time predictions will be correct when using this linear model. Compared to the Pearson correlation coefficient between quarter-mile race time and horsepower of -0.71, we can confirm our r-squared value is approximately the square of our r-value.

In addition, the p-value of our linear regression analysis is 5.77 x 10-6, which is much smaller than our assumed significance level of 0.05%. Therefore, we can state that there is sufficient evidence to reject our null hypothesis, which means that the slope of our linear model is not zero.

Once we have calculated our linear regression model, we can visualize the fitted line against our dataset using ggplot2.

First, we need to calculate the data points to use for our line plot using our lm(qsec ~ hp,mtcars) coefficients as follows:

> model <- lm(qsec ~ hp,mtcars) #create linear model

> yvals <- model$coefficients['hp']\*mtcars$hp +

model$coefficients['(Intercept)'] #determine y-axis values from linear model

Once we have calculated our line plot data points, we can plot the linear model over our scatter plot:

> plt <- ggplot(mtcars,aes(x=hp,y=qsec)) #import dataset into ggplot2

> plt + geom\_point() + geom\_line(aes(y=yvals), color = "red") #plot scatter and linear model

Using our visualization in combination with our calculated p-value and r-squared value, we have determined that there is a significant relationship between horsepower and quarter-mile time.

Although the relationship between both variables is statistically significant, this linear model is not ideal. According to the calculated r-squared value, using only quarter-mile time to predict horsepower is roughly as accurate as guessing using a coin toss. In other words, the variability we observed within our horsepower data must come from multiple sources of variance. To accurately predict future horsepower observations, we need to use a more robust model.

Multiple linear regression

**Multiple linear regression** is a statistical model that extends the scope and flexibility of a simple linear regression model. Instead of using a single independent variable to account for all variability observed in the dependent variable, a multiple linear regression uses multiple independent variables to account for parts of the total variance observed in the dependent variable.

As a result, the linear regression equation is no longer *y = mx + b*. Instead, the multiple linear regression equation becomes *y = m1x1 + m2x2 + … + mnxn + b*, for all independent *x* variables and their *m* coefficients.

In actuality, a multiple linear regression is a simple linear regression in disguise—all of the assumptions, hypotheses, and outputs are the same. The only difference between multiple linear regression and simple linear regression is how we will evaluate the outputs.

When it comes to multiple linear regression, we'll look at each independent variable to determine if there is a significant relationship with the dependent variable. Once we have evaluated each independent variable, we'll evaluate the r-squared value of the model to determine if the model sufficiently predicts our dependent variable.

To practice multiple linear regression, let's revisit our mtcars dataset. From our last example, we determined that quarter-mile time was not adequately predicted from just horsepower. To better predict the quarter-mile time (qsec) dependent variable, we can add other variables of interest such as fuel efficiency (mpg), engine size (disp), rear axle ratio (drat), vehicle weight (wt), and horsepower (hp) as independent variables to our multiple linear regression model.

In R, our multiple linear regression statement is as follows:

> lm(qsec ~ mpg + disp + drat + wt + hp,data=mtcars) #generate multiple linear regression model

Similar to the simple linear regression, the output of multiple linear regression using the lm() function produces the coefficients for each variable in the linear equation.

**NOTE**

Because multiple linear regression models use multiple variables and dimensions, they are almost impossible to plot and visualize.

Now that we have our multiple linear regression model, we need to obtain our statistical metrics using the summary()function. In your R console, use the following statement:

>summary(lm(qsec ~ mpg + disp + drat + wt + hp,data=mtcars)) #generate summary statistics

In addition to overall model fit and the statistical test for slope, most data scientists would be curious about the contribution of each variable to the multiple linear regression model. To determine which variables provide a significant contribution to the linear model, we must look at the individual variable p-values.

In the summary output, each Pr(>|t|) value represents the probability that each coefficient contributes a random amount of variance to the linear model. According to our results, vehicle weight and horsepower (as well as intercept) are statistically unlikely to provide random amounts of variance to the linear model. In other words the vehicle weight and horsepower have a significant impact on quarter-mile race time. When an intercept is statistically significant, it means there are other variables and factors that contribute to the variation in quarter-mile time that have not been included in our model. These variables may or may not be within our dataset and may still need to be collected or observed.

Despite the number of significant variables, the multiple linear regression model outperformed the simple linear regression. According to the summary output, the r-squared value has increased from 0.50 in the simple linear regression model to 0.71 in our multiple linear regression model while the p-value remained significant.

**CAUTION**

Although the multiple linear regression model is far better at predicting our current dataset, the lack of significant variables is evidence of overfitting. Overfitting means that the performance of a model performs well with a current dataset, but fails to generalize and predict future data correctly. Later in this course we'll learn more about overfitting and ways to avoid it.

Depending on the dataset, the questions being asked, and the audience, a simple linear regression model may be more appropriate than a multiple linear regression model. However, the amount of information that can be obtained and analyzed will be far greater using a multiple linear regression.

As with any data model, it takes practice to learn how to identify variables of interest, select an appropriate model, and refine a model to increase performance. Before moving to the next section, take some time to perform correlation analysis on our previous datasets. Then use the correlation analysis to identify potential variables of interest. Once you have variables of interest, practice generating simple and multiple linear regression models to try and create accurate predictive models.

Chi Square Test

As we learned previously, categorical data is generally any data that is not measured, or qualitative data. Even though categorical data may not require an instrument to measure, it can be just as informative as numerical data.

One common form of categorical data is **frequency data**, where we record how often something was observed within a single variable. For example, in the mpg dataset, if we were to count up the number of vehicles for each vehicle class, the output would be a form of frequency data.

In data science, we'll often compare frequency data across another dichotomous factor such as gender, A/B groups, member/non-member, and so on. In these cases, we may ask ourselves, "Is there a difference in frequency between our first and second groups?" To test this question, we can perform a chi-squared test.

The **chi-squared test** is used to compare the distribution of frequencies across two groups and tests the following hypotheses:

H0 : There **is no difference** in frequency distribution between both groups.

Ha : There **is a difference** in frequency distribution between both groups

Before we can perform our chi-squared analysis, we must ensure that our dataset meets the assumptions of the statistical test:

1. Each subject within a group contributes to only one frequency. In other words, the sum of all frequencies equals the total number of subjects in a dataset.
2. Each unique value has an equal probability of being observed.
3. There is a minimum of five observed instances for every unique value for a 2x2 chi-squared table.
4. For a larger chi-squared table, there is at least one observation for every unique value and at least 80% of all unique values have five or more observations.

Once we have confirmed our categorical dataset meets all of the assumptions of the chi-square analysis, we can perform our chi-squared test.

In R, we'll compute our chi-squared test using the chisq.test() function. Type the following code into the R console to look at the chisq.test() documentation in the Help pane.

>?chisq.test()

Depending on the structure of your dataset, you can implement the chisq.test() function in multiple ways using the optional arguments. The most straightforward implementation of chisq.test() function is passing the function to a contingency table. A **contingency table** is another name for a frequency table produced using R's table() function. R's table() function does all the heavy lifting for us by calculating frequencies across factors.

For example, if we want to test whether there is a statistical difference in the distributions of vehicle class across 1999 and 2008 from our mpg dataset, we would first need to build our contingency table as follows:

> table(mpg$class,mpg$year) #generate contingency table

Then, pass the contingency table to the chisq.test()function:

> tbl <- table(mpg$class,mpg$year) #generate contingency table

> chisq.test(tbl) #compare categorical distributions

**IMPORTANT**

The chi-squared warning message is due to the small sample size. Because the p-value is so large, we are not too concerned that our interpretation may be incorrect.

Despite having no quantitative input, the chi-squared test enables data scientists to quantify the distribution of categorical variables. Although this test can be applied to more groups and larger datasets, it does have a limit. Increasing the number of groups also increases the likelihood that insignificant changes will incorrectly be considered significant. Therefore, it's important to keep the number of unique values and groups relatively low. A good rule of thumb is to keep the number of unique values and groups lower than 20, which means the degrees of freedom (df in the output) is less than or equal to 19.

Take some additional time to practice implementing contingency tables and chi-squared tests using categorical data from our previous datasets. Feel free to tweak the frequency values in the contingency tables tp see what happens to the chi-squared and p-value metrics.

A/B Testing

Often when performing analysis and testing on a well-established product, website, or software, making changes can be difficult. Well-established products typically have a large consumer base and reliable sales and usage metrics, and are highly valued by their company. As a result, it's too risky to implement changes directly to the product without proper evaluation of the consequences.

To properly evaluate potential product changes, companies can use a technique called A/B testing. **A/B testing** is a randomized controlled experiment that uses a control (unchanged) and experimental (changed) group to test potential changes using a success metric. A/B testing is used to test whether or not the distribution of the success metric increases in the experiment group instead of the control group; we would not want to make changes to the product that would cause a decrease in the success metric.

Although A/B testing has been around for almost a century, giant software and tech companies such as Google and Amazon have popularized the practice by providing in-depth analytic metrics for their Google AdSense and AWS platforms.

Regardless of the industry or product, the process of A/B testing is the same. First, we must decide what changes will be made to the experimental group. Typically, the number of changes will be very limited to ensure comparisons are equal; however, more substantial changes can also be tested using an A/B framework.

Once a consensus has been made on the changes to be made to the experimental group, a success metric should be determined. The success metric can vary widely, depending on what is being tested. For example, a website might use consumer engagement as a success metric (e.g., number of visitors, clicked links, or time spent on a page). Alternatively, an automotive design team might want to know how performance changes after a slight design change to a vehicle's form factor, so the team's success metric might be mpg fuel efficiency.

Once we have decided on our experimental changes and the success metric, we must determine which statistical test is most appropriate. In this course, we'll only concern ourselves with normally distributed data and categorical data, which limits the number of statistical tests we'll need. However, if the A/B test groups are disproportionately uneven, or if the success metric distribution is non-normal, more elaborate statistical analysis may be required.

For our purposes, we can apply the following logic to determine the most appropriate statistical test:

* If the success metric is **numerical** and the **sample size is small**, a **z-score summary** **statistic** can be sufficient to compare the mean and variability of both groups.
* If the success metric is **numerical** and the **sample size is large**, a **two-sample t-test** should be used to compare the distribution of both groups.
* If the success metric is **categorical**, you may use a **chi-squared test** to compare the distribution of categorical values between both groups.

After determining the testing conditions and statistical test, the next consideration in A/B testing is sample size. It's important to collect a sufficient number of data points for each group to ensure that the A/B test results are meaningful.

There are multiple ways to determine optimal sample size, such as quantitative power analyses, but often a qualitative estimate is sufficient. If the changes made to the experimental group are expected to have a strong effect on the success metric (often referred to in data science as an **effect size**), fewer data points are necessary for the test. In contrast, if the effect size is small, a larger sample size will be necessary for meaningful statistical findings.

For example, if we were testing purchase rates on an experiment group that receives a pop-up notification when visiting the AutosRUs website, we may use historical purchase rates as an indicator of effect size. If in general people who visit the site are likely to purchase a vehicle, our A/B test sample size can be small. However, if most people who visit the site are not likely to purchase a vehicle, we would need a large number of data points to confirm if the pop-up notifications make a statistical difference.

**NOTE**

Using a quantitative power analysis can be helpful to determine sample size when effect size is unknown or resources are limited. Although performing a power analysis is outside the scope of this course, there is robust documentation online regarding [implementation (Links to an external site.)](http://www.statsoft.com/Textbook/Power-Analysis) and [interpretation (Links to an external site.)](https://www.statisticssolutions.com/statistical-power-analysis/).

Due to its simple design and flexible application, the A/B testing framework is quickly becoming a go-to standard in the data science industry and one of the most highly desired data skills for Fortune 500 companies. Regardless, if you have experience in product design or optimization, you can use A/B testing to make informed design changes and confident development decisions.

Deciding on an analysis

When using data to make informed decisions in a professional environment, implementing a statistics function is not the biggest challenge. Rather, it's determining what questions to ask.

In data science, researchers use **retrospective analysis** to analyze and interpret a previously generated dataset where the outcome is already known. Retrospective analyses are helpful because there are no upfront costs to generate data and statistical results can be compared to the known outcomes. Depending on the dataset and input variables, there is a (potentially) limitless number of statistical questions that can be asked from the data:

* Are two groups statistically different? Use a t-test with one dichotomous independent variable and one continuous dependent variable.
* Can one continuous dependent variable be predicted using another independent variable? What about multiple independent variables and one dependent variable? Use regression analysis.
* Are there multiple categorical variables tightly linked in a dataset? Are the distributions of the different categorical variables equal? We can test with chi-squared.

In contrast, researchers can **design their own study** to answer their own specific questions. In this case, the data types and size of the dataset will be directly reflective of how complicated their hypotheses are, and what statistical analyses are required to answer the question.

For example, if we want to verify that a car battery ages at an appropriate rate, we would need to test our question with a regression model. If we were to use a multiple linear regression model, we would need to collect numerical variables, such as number of uses, time, battery capacity, tire tread, and engine horsepower. Once we select the variables to collect, we would estimate sample size based on how low of a significance level is necessary and how sensitive the measurements are.

Regardless, if we collect and measure the data ourselves, or if the data has been curated from a previous dataset, statistical tests can help us provide quantitative interpretation to the results.