Qualitative Data

# Frequency Distribution of Qualitative Data

The **frequency distribution**of a data variable is a summary of the data occurrence in a collection of non-overlapping categories.

#### Example

In the data set painters, the frequency distribution of the School variable is a summary of the number of painters in each school.

#### Problem

Find the frequency distribution of the painter schools in the data set painters.

#### Solution

We apply the table function to compute the frequency distribution of the School variable.

> library(MASS)                 # load the MASS package   
> school = painters$School      # the painter schools   
> school.freq = table(school)   # apply the table function

#### Answer

The frequency distribution of the schools is:

> school.freq   
school   
 A  B  C  D  E  F  G  H   
10  6  6 10  7  4  7  4

#### Enhanced Solution

We apply the cbind function to print the result in column format.

> cbind(school.freq)   
  school.freq   
A          10   
B           6   
C           6   
D          10   
E           7   
F           4   
G           7   
H           4

#### Exercise

1. Find the frequency distribution of the composition scores in painters.
2. Find programmatically the school that has the most painters.

# Relative Frequency Distribution of Qualitative Data

The **relative frequency distribution**of a data variable is a summary of the frequency proportion in a collection of non-overlapping categories.

The relationship of frequency and relative frequency is:

Relative F requency =-Frequency-
                    Sample Size


#### Example

In the data set painters, the relative frequency distribution of the School variable is a summary of the proportion of painters in each school.

#### Problem

Find the relative frequency distribution of the painter schools in the data set painters.

#### Solution

We first apply the table function to compute the frequency distribution of the School variable.

> library(MASS)                 # load the MASS package   
> school = painters$School      # the painter schools   
> school.freq = table(school)   # apply the table function

Then we find the sample size of painters with the nrow function, and divide the frequency distribution with it. Therefore the relative frequency distribution is:

> school.relfreq = school.freq / nrow(painters)

#### Answer

The relative frequency distribution of the schools is:

> school.relfreq   
school   
       A        B        C        D        E        F   
0.185185 0.111111 0.111111 0.185185 0.129630 0.074074   
       G        H   
0.129630 0.074074

#### Enhanced Solution

We can print with fewer digits and make it more readable by setting the digits option.

> old = options(digits=1)   
> school.relfreq   
school   
   A    B    C    D    E    F    G    H   
0.19 0.11 0.11 0.19 0.13 0.07 0.13 0.07   
> options(old)

In addition, we can apply the cbind function to print the result in column format.

> old = options(digits=1)   
> cbind(school.relfreq)   
  school.relfreq   
A           0.19   
B           0.11   
C           0.11   
D           0.19   
E           0.13   
F           0.07   
G           0.13   
H           0.07   
> options(old)    # restore the old option

#### Exercise

Find the relative frequency distribution of the composition scores in painters.

# Bar Graph

A **bar graph**of a qualitative data sample consists of vertical parallel bars that shows the frequency distribution graphically.

#### Example

In the data set painters, the bar graph of the School variable is a collection of vertical bars showing the number of painters in each school.

#### Problem

Find the bar graph of the painter schools in the data set painters.

#### Solution

We first apply the table function to compute the frequency distribution of the School variable.

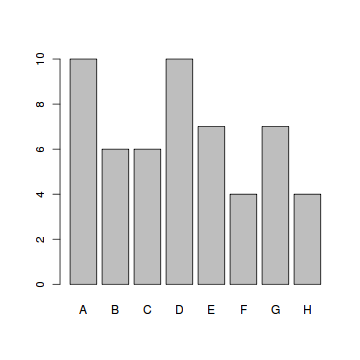
> library(MASS)                 # load the MASS package   
> school = painters$School      # the painter schools   
> school.freq = table(school)   # apply the table function

Then we apply the barplot function to produce its bar graph.

> barplot(school.freq)         # apply the barplot function

#### Answer

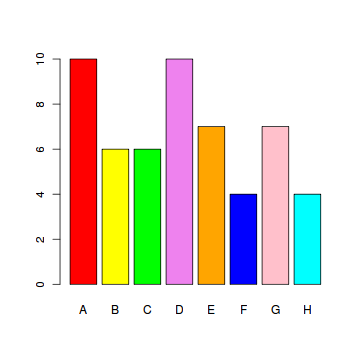
The bar graph of the school variable is:



#### Enhanced Solution

To colorize the bar graph, we select a color palette and set it in the col argument of barplot.

> colors = c("red", "yellow", "green", "violet",   
+   "orange", "blue", "pink", "cyan")   
> barplot(school.freq,         # apply the barplot function   
+   col=colors)                # set the color palette



# Pie Chart

A **pie chart**of a qualitative data sample consists of pizza wedges that shows the frequency distribution graphically.

#### Example

In the data set painters, the pie chart of the School variable is a collection of pizza wedges showing the proportion of painters in each school.

#### Problem

Find the pie chart of the painter schools in the data set painters.

#### Solution

We first apply the table function to produce the frequency distribution of School.

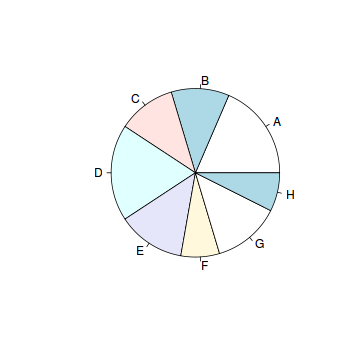
> library(MASS)                 # load the MASS package   
> school = painters$School      # the painter schools   
> school.freq = table(school)   # apply the table function

Then we apply the pie function to produce its pie chart.

> pie(school.freq)              # apply the pie function

#### Answer

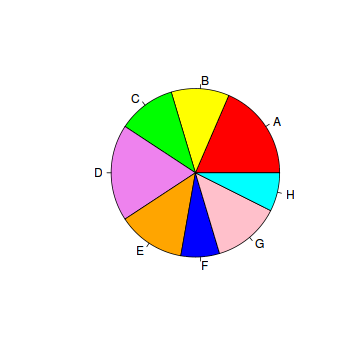
The pie chart of the school variable is:



#### Enhanced Solution

To colorize the pie chart, we select a color palette and set it in the col argument of pie.

> colors = c("red", "yellow", "green", "violet",   
+   "orange", "blue", "pink", "cyan")   
> pie(school.freq,             # apply the pie function   
+   col=colors)                # set the color palette



#### Exercise

Find the pie chart of the composition scores in painters.

# Category Statistics

In the built-in data set painters, the painters are classified according to the schools they belong. Each school can be characterized by its various statistics, such as mean composition, drawing, coloring and expression scores.

Suppose we would like to know which school has the highest mean composition score. We would have to first find out the mean composition score of each school. The following shows how to find the mean composition score of an arbitrarily chosen school.

#### Problem

Find out the mean composition score of school C in the data set painters.

#### Solution

The solution consists of a few steps:

1. Create a logical index vector for school C.

> library(MASS)                 # load the MASS package   
> school = painters$School      # the painter schools   
> c\_school = school == "C"      # the logical index vector

1. Find the child data set of painters for school C. For explanation, please consult the tutorial of *Data Frame Row Slice*.

> c\_painters = painters[c\_school, ]  # child data set

1. Find the mean composition score of school C.

> mean(c\_painters$Composition)   
[1] 13.167

#### Answer

The mean composition score of school C is 13.167.

#### Alternative Solution

Instead of computing the mean composition score manually for each school, use the tapplyfunction to compute them all at once.

> tapply(painters$Composition, painters$School, mean)   
     A      B      C      D      E      F      G      H   
10.400 12.167 13.167  9.100 13.571  7.250 13.857 14.000

#### Exercise

1. Find programmatically the school with the highest composition scores.
2. Find the percentage of painters whose color score is equal to or above 14.

What is Multiple Regression?

Multiple regression is a process that helps to understand the correlation between several independent and one dependent variable. Unfortunately, creating an accurate model with this method can be time consuming because the underlying principles of regression can require a great deal of expertise to understand and implement. The good news is that if you need a quick overview of data relationships, then Tableau can help you create and test hypotheses with ease.

Multiple Regression in Tableau

In this example I am looking at a small data set that contains information on wins, runs, saves, etc. from all 30 Major League Baseball teams in 2008; the idea is to better determine which stats can predict wins.

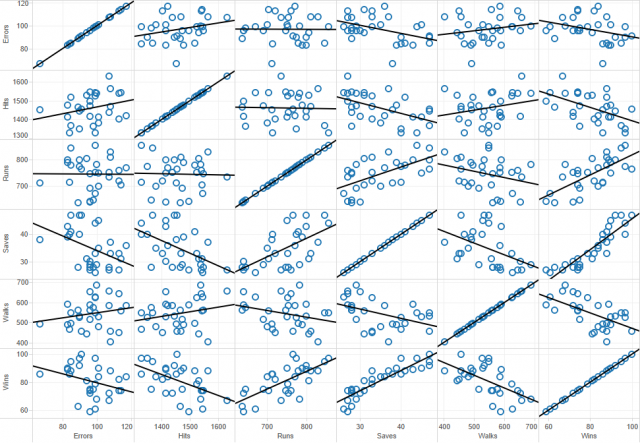
Create a Scatter Plot Matrix

First, we will create a **scatter plot matrix** using all of our variables. The matrix is created by simply dragging each measure into the **Row** and **Column** shelf and setting each variable as a **dimension**.

Drag measures into Row and Column shelf

Once the matrix is populated, use the **Show Trend Lines** option to display.

**NOTE:** In a regression, it is important to observe not just the relationship between your independents and dependent but also the correlation between the independents themselves. If two independent variables are highly correlated, then the information they add to the model is redundant. A scatter plot matrix is a great way to quickly visualize the univariate relationships between all variables in the model.

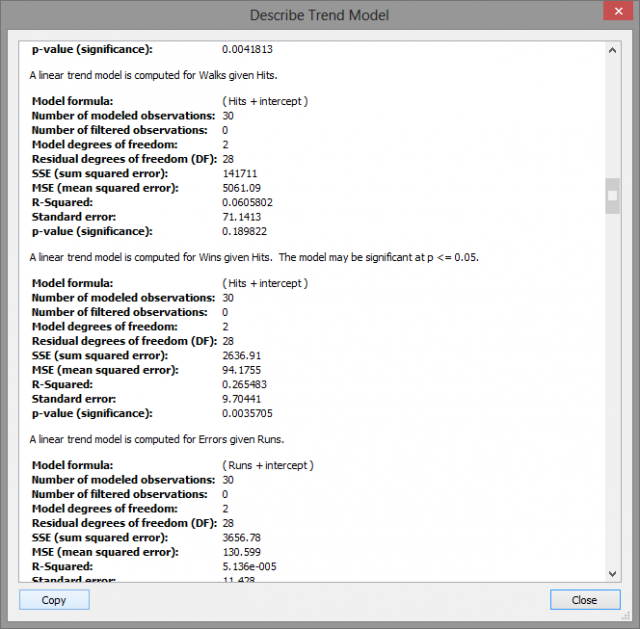


The first thing that is obvious from the matrix above is that every variable except Errors seems to strongly correlate with the dependent variable, Wins. Mathematically, this is not determined by the slope of the trend line but the size of the error. Visually this can be observed by looking at the spread of the data points around the trend line.

Additionally, some of you may have noticed that Wins given Saves seems to be a non-linear relationship. It turns out that, for this model, Saves is ripe for a data transformation.

### Use ANOVA Values to Determine Statistical Significance

So for the moment, all of the work we have done has been casual and non-rigorous. Now that we have some general idea about the story our data is trying to tell we can use ANOVA values to determine statistical significance. Under the **Analysis** menu in Tableau, there is a **Describe Trend Lines** option that gives an output like the one below.



These are the univariate relationships between each variable pair. The best fit trend line is determined by a reduction in the error between the observed and predicted value. This concept is measured objectively by the **R-Squared** value which can be roughly understood as an inverse relation to the error of the trend line.

A higher R-Squared value indicates a lower error and thus a higher correlation between the two variables. Above, the computer R-Squared value for Wins given Hits is .2655 which means that Hits account for 26.55% of the variation in the Win total. A **p-value** lower than .05 indicates that the variable is a statistically significant predictor within the usual 95% confidence interval.

### Final Analysis

When evaluating a model with many potential variables, the number of pairs grows exponentially. Using the scatter plot matrix, you can pinpoint the important factors through visual observation so that you have fewer pairs to analyze later. With a full statistical software package, you could perform a best subset analysis and determine an accurate regression model if it exists for the most significant variables.

It turns out that one of the most accurate models for the given data uses Hits, Runs, Walks, and the Inverse of Saves to predict Wins. This is not too surprising since you have already quickly determined from this simple analysis that Errors was a seemingly weak predictor, and Saves was a candidate for data transformation.

While our method did not provide a comprehensive model, it gave a quick but in-depth understanding of the relationships within the data without the time spent on outlier analysis or sub-setting variables.

**Brinell Hardness Scores**

An engineer measured the Brinell hardness of 25 pieces of ductile iron that were subcritically annealed. The resulting data were:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 170 | 167 | 174 | 179 | 179 | 187 | 179 | 183 | 179 |
| 156 | 163 | 156 | 187 | 156 | 167 | 156 | 174 | 170 |
| 183 | 179 | 174 | 179 | 170 | 159 | 187 |  |  |

The engineer hypothesized that the mean Brinell hardness of all such ductile iron pieces is greater than 170. Therefore, he was interested in testing the hypotheses:

H0 : μ = 170  
HA: μ > 170

The engineer entered his data into Minitab and requested that the "one-sample t-test" be conducted for the above hypotheses. He obtained the following output:

#### Descriptive Statistics

| **N** | **Mean** | **StDev** | **SE Mean** | **95% Lower Bound** |
| --- | --- | --- | --- | --- |
| 25 | 172.52 | 10.31 | 2.06 | 168.99 |

μμ: mean of Brinelli

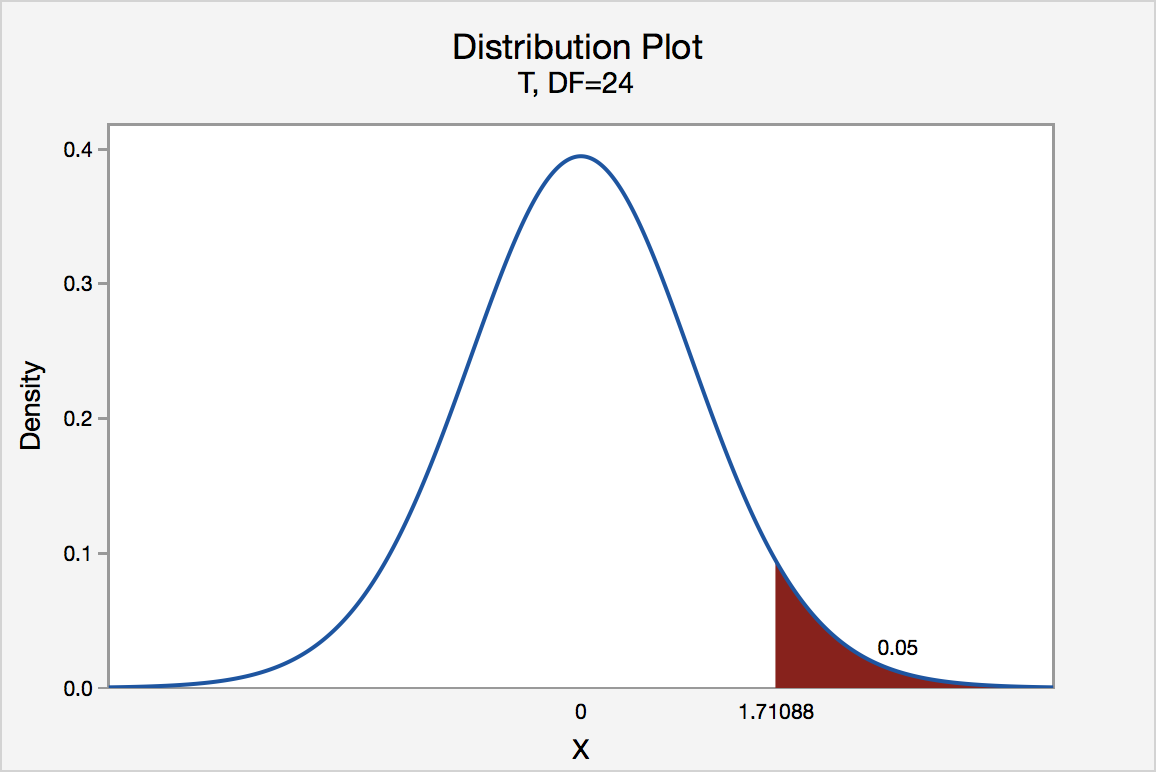
#### Test

Null hypothesis    H₀: μμ = 170  
Alternative hypothesis    H₁: μμ > 170

| **T-Value** | **P-Value** |
| --- | --- |
| 25 | 172.52 |

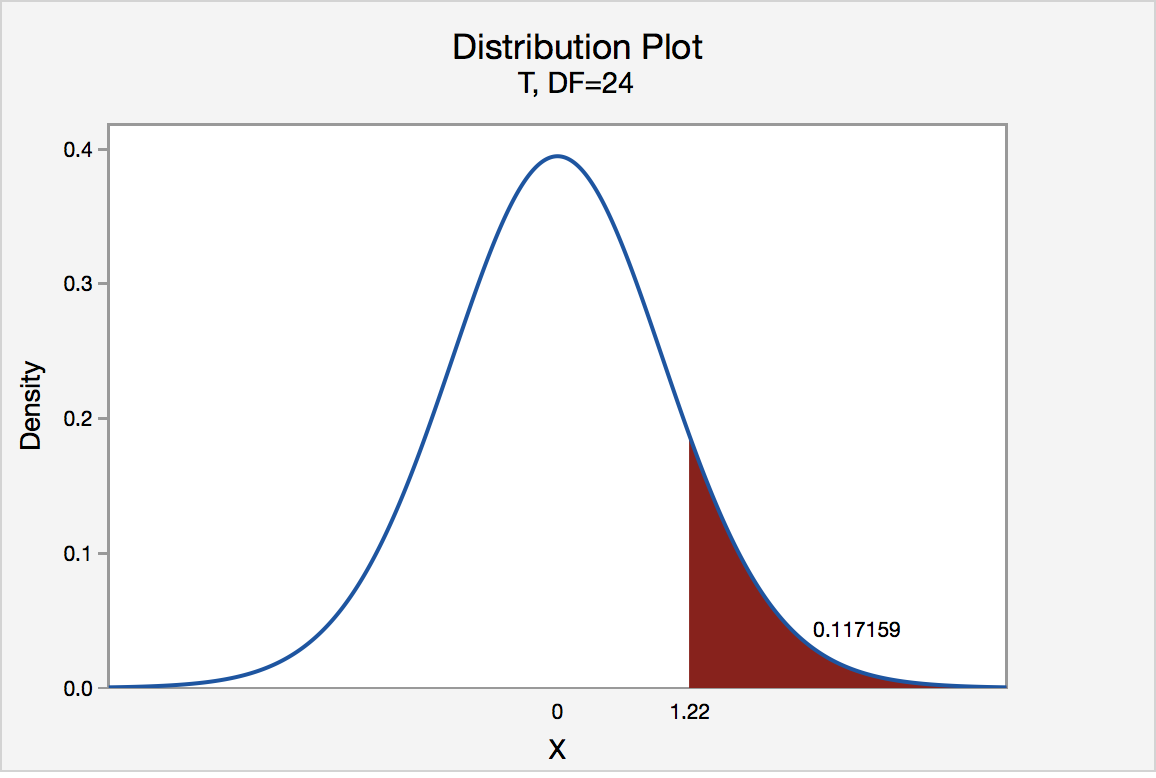
The output tells us that the average Brinell hardness of the n = 25 pieces of ductile iron was 172.52 with a standard deviation of 10.31. (The standard error of the mean "SE Mean", calculated by dividing the standard deviation 10.31 by the square root of n = 25, is 2.06). The test statistic t\* is 1.22, and the P-value is 0.117.

If the engineer set his significance level α at 0.05 and used the critical value approach to conduct his hypothesis test, he would reject the null hypothesis if his test statistic t\* were greater than 1.7109 (determined using statistical software or a t-table):



Since the engineer's test statistic, t\* = 1.22, is not greater than 1.7109, the engineer fails to reject the null hypothesis. That is, the test statistic does not fall in the "critical region." There is insufficient evidence, at the αα = 0.05 level, to conclude that the mean Brinell hardness of all such ductile iron pieces is greater than 170.

If the engineer used the P-value approach to conduct his hypothesis test, he would determine the area under a tn - 1 = t24curve and to the right of the test statistic t\* = 1.22:



In the output above, Minitab reports that the P-value is 0.117. Since the P-value, 0.117, is greater than αα = 0.05, the engineer fails to reject the null hypothesis. There is insufficient evidence, at the αα = 0.05 level, to conclude that the mean Brinell hardness of all such ductile iron pieces is greater than 170

## 1. Hypothesis Testing In R

This blog is all about Hypothesis testing in **R**. It is the assumption made by the researcher about the population of data collected for any experiment. First, we will introduce you with the statistical hypothesis in R, subsequently, it will cover the decision error in R, one and two sample t-test, u-test, Correlation and Covariance in R etc.

## 2. Introduction to Statistical Hypothesis in R

A statistical hypothesis is an assumption made by the researcher about the population of data collected for any experiment. It is not mandatory for this assumption to be true every time. Hypothesis testing is, in a way, the formal way of validating the hypothesis made by the researcher.

In order to validate a hypothesis, it will consider the entire population into account. However, this is not possible practically. Thus, to validate a hypothesis, it will use random samples from a population. On the basis of the result from testing over the sample data, it either selects or rejects the hypothesis.

Statistical Hypothesis can be categorized into 2 types as below:

* **Null Hypothesis –** Hypothesis tests are used to test the validity of a claim that is made about a population. This claim that’s on trial, in essence, is called the null hypothesis. The null hypothesis testing is denoted by H0.
* **Alternative Hypothesis –** The alternative hypothesis is the one you would believe if the null hypothesis is concluded to be untrue. The evidence in the trial is your data and the statistics that go along with it. The alternative hypothesis testing is denoted by H1or Ha.

Let’s take an example of the coin. You want to conclude that a coin is perfectly balanced or not. Since null hypothesis refers to the natural state of an event, thus, according to the null hypothesis, there would an equal number of occurrences of heads and tails, if a coin is tossed several times. On the other hand, alternative hypothesis negates the null hypothesis and refers that the occurrences of heads and tails would have significant differences in number.

## 3. Hypothesis Testing in R

Statisticians use hypothesis testing to formally check whether the hypothesis is accepted or rejected. Hypothesis testing is conducted in the following manner:

* **State the Hypotheses –** This stage involves stating the null and alternative hypotheses.
* **Formulate an Analysis Plan –** This stage involves the construction of an analysis plan.
* **Analyze Sample Data –** This stage involves the calculation and interpretation of the test statistic as described in the analysis plan.
* **Interpret Results –**This stage involves the application of the decision rule described in the analysis plan.

All hypothesis tests ultimately use a p-value to weigh the strength of the evidence or in other words what the data are about the population. The p-value is a number between 0 and 1 and interpreted in the following way:

A small p-value (typically ≤0.05) indicates strong evidence against the null hypothesis, so you reject it. A large p-value (> 0.05) indicates weak evidence against the null hypothesis, so you fail to reject it. A p-value very close to the cutoff (0.05) is considered to be marginal and could go either way.

## 4. Decision Errors in R

2 types of errors can occur from hypothesis test:

* **Type I Error –** Type I error occurs when the researcher rejects a null hypothesis when it is true. The term significance level is used to express the probability of Type I error while testing hypothesis. The significance level is represented by the symbol α(alpha).
* **Type II Error –** Accepting a false null hypothesis H0 is referred as the Type II error. The term power of the test is used to express the probability of Type II error while testing hypothesis. The power of the test is represented by the symbol β(beta).

## 5. Using the Student’s t-test in R

The Student’s t-test is a method for comparing two samples. It can be implemented to determine whether the samples are different. This is a parametric test, and the data should be normally distributed.

R can handle the various versions of t-test using the**t.test()** command. The test can be used to deal with two- and one-sample tests as well as paired tests.

Listed below are the commands used in the Student’s t-test and their explanations:

* **t.test(data.1, data.2)** – The basic method of applying a t-test is to compare two vectors of numeric data.
* **var.equal = FALSE** – If the var.equal instruction is set to TRUE, the variance is considered to be equal and the standard test is carried out. If the instruction is set to FALSE (the default), the variance is considered unequal and the Welch two-sample test is carried out.
* **mu = 0** – If a one-sample test is carried out, mu indicates the mean against which the sample should be tested.
* **alternative = “two.sided”** – It sets the alternative hypothesis. The default is “two.sided” but you can specify “greater” or “less”. You can abbreviate the instruction.
* **conf.level = 0.95** – It sets the confidence level of the interval (default = 0.95).
* **paired = FALSE** – If set to TRUE, a matched pair t-test is carried out.
* **t.test(y ~ x, data, subset)** – The required data can be specified as a formula of the form response ~ predictor. In this case, the data should be named and a subset of the predictor variable can be specified.
* **subset = predictor %in% c(“sample.1”, sample.2”)** – If the data is in the form response ~ predictor, the subset instruction can specify which two samples to select from the predictor column of the data.

## 6. Two-Sample t-test with Unequal Variance

The**t.test()** command is generally used to compare two vectors of numeric values. The vectors can be specified in a variety of ways, depending on how your data objects are set out.

The default form of the t.test() command does not assume that the samples have equal variance. As a result, the two-sample test is carried out unless specified otherwise. The two-sample test can be on any two datasets using the following command:

> t.test(data2, data3)

Welch two-sample t-test

1. data: data2 **and** data3
2. t = -2.8151, df = 24.564, p-value = 0.009462

As per the alternative hypothesis, we can infer that the true difference in means is not equal to 0.

On the basis of 95% confidence interval, the output will be:

-3.5366789 - .5466544

As per samples estimate,

Mean of x = 5.125000

Mean of y = 7.166667

The default clause in the t.test() command can be overridden. To do so, add the var.equal = TRUE instruction to the standard t.test() command. This instruction forces the t.test() command to assume that the variance of the two samples is equal.

The calculation of the t-value uses pooled variance, and the degrees of freedom are unmodified. As a result, the p-value is slightly different from the Welch version. For example:

> t.test(data2, data3, **var**.equal = TRUE)

Two-Sample t-test

1. data: data2 **and** data3
2. t = -2.7908, df = 26, p-value = 0.009718

alternative hypothesis: true difference in means is not equal to 0.

95 percent confidence interval:

-3.5454233 -0.5379101

sample estimates:

1. mean of x mean of y
2. 5.125000 7.166667

## 7. One-Sample t-testing in R

To perform analysis, it collects a large amount of data from various sources and test it on random samples. In several situations when the population of collected data is unknown, researchers test samples to identify the population. The one-sample t-test is one of the useful tests for testing sample’s population.

This test is used for a testing mean of samples. For example, you can use this test to compare that a sample of students from a particular college is identical or different from the sample of general students. In this situation, the hypothesis tests that the sample is from a known population with a known mean (m) or from an unknown population.

To carry out a one-sample t-test in R, the name of a single vector and the mean with which it is compared is supplied.

The mean defaults to 0.

The one-sample t-test can be implemented as follows:

> t.test(data2, mu = 5)

One-Sample t-test

1. data: data2
2. t = 0.2548, df = 15, p-value = 0.8023

alternative hypothesis: true mean is not equal to 5

95% confidence interval

4.079448 6.170552

Sample estimates: mean of x=5.125

## 8. Using Directional Hypotheses in R

You can also specify a “direction” to your hypothesis.

In many cases, you are simply testing to see if the means of two samples are different, but you may want to know if a sample mean is lower or greater than another sample mean. You can use the alternative equal to (=) instruction to switch the emphasis from a two-sided test (the default) to a one-sided test. The choices you have are between ″two.sided″, ″less″, or ″greater″, and the choice can be abbreviated, as shown in the following command:

> t.test(data2, mu = 5, alternative = 'greater')

One-Sample t-test

1. data: data2
2. t = 0.2548, df = 15, p-value = 0.4012

alternative hypothesis: true mean is greater than 5

95 percent confidence interval:4.265067 Inf

sample estimates: mean of x=5.125

## 9. Formula Syntax and Subsetting Samples in the t-test in R

As discussed in the previous sections, the t-test is designed to compare two samples.

So far, you have seen how to carry out the t-test on separate vectors of values; however, your data may be in a more structured form with a column for the response variable and a column for the predictor variable.

When the data is available in a more structured form with a column for the response variable and a column for the predictor variable, the data can be set in a more sensible and flexible manner. You need a new way to deal with the layout.

R deals with the layout by using a formula syntax.

You can create a formula by using the tilde (~) symbol. Essentially, your response variable goes to the left of the ~ and the predictor goes to the right, as shown in the following command:

> t.test(rich ~ graze, data = grass)

If your predictor column contains more than two items, the t-test cannot be used; however, you can still carry out a test by subsetting this predictor column and specifying the two samples you want to compare.

The subset = instruction should be used as a part of the t.test() command, as follows:

**Formula Syntax in R  – The following example illustrates how to do this using the same data as in the previous example:**

> t.test(rich ~ graze, data = grass, subset = graze %in% c('mow‘, 'unmow'))

You first specify the column you want to take your subset from and then type %in%. This tells the command that the list that follows is in the graze column. Note that you have to put the levels in quotes; here you compare ″mow″and ″unmow″and your result is identical to the one you obtained before.

## 10. u-test in R

When you have two samples to compare and your data is nonparametric, you can use the u-test. This goes by various names and may be known as the Mann—Whitney u-test or Wilcoxon sign rank test. The**wilcox.test()** command can carry out the analysis.

The wilcox.test() command can conduct two-sample or one-sample tests, and you can add a variety of instructions to carry out the test.

Given below are the main options available in the wilcox.test() command with their explanations:

* **test(sample.1, sample.2)** – It carries out a basic two-sample u-test on the numerical vectors specified.
* **mu = 0** – If a one-sample test is carried out, mu indicates the value against which the sample should be tested.
* **alternative = “two.sided”** – It sets the alternative hypothesis. The default is “two.sided” but you can specify “greater” or “less”. You can abbreviate the instruction but you still need the quotes.
* **int = FALSE** – It sets whether confidence intervals should be reported.
* **level = 0.95** – It sets the confidence level of the interval (default = 0.95).
* **correct = TRUE** – By default, the continuity correction is applied. You can turn this off by setting it to FALSE.
* **paired = FALSE** – If set to TRUE, a matched pair u-test is carried out.
* **exact = NULL** – It sets whether an exact p-value should be computed. The default is to do so for less than 50 items.
* **test(y ~ x, data, subset)** – The required data can be specified as a formula of the form response ~ predictor. In this case, the data should be named and a subset of the predictor variable can be specified.
* **subset = predictor %in% c(″1″, ″sample.2″)** – If the data is in the form response ~ predictor, the subset instruction can specify which two samples to select from the predictor column of the data

## 11. Two-Sample u-test in R

The basic way of using the wilcox.test()command is to specify the two samples you want to compare as separate vectors, as shown in the following command:

1. > data1 ; data2
2. [1] 3 5 7 5 3 2 6 8 5 6 9
3. [1] 3 5 7 5 3 2 6 8 5 6 9 4 5 7 3 4
4. > wilcox.test(data1, data2)

Wilcoxon rank sum test with continuity correction

1. data: data1 **and** data2
2. W = 94.5, p-value = 0.7639

alternative hypothesis: true location shift is not equal to 0

By default, the confidence intervals are not calculated and the p-value is adjusted using the “continuity correction”; a message tells you that the latter has been used.  In this case, you see a  warning message because you have tied values in the data. If you set exact = FALSE, this message would not be displayed because the p-value would be determined from a normal approximation method

Any doubt yet in Hypothesis Testing In R? Please Comment.

## 12. One-Sample u-test in R

When you specify a single numerical vector, then it carries out a one-sample u-test. The default is to set mu = 0. For example:

> wilcox.test(data3, exact = FALSE)

Wilcoxon signed rank test with continuity correction

1. data: data3
2. V = 78, p-value = 0.002430

alternative hypothesis: true location is not equal to 0

In this case, the p-value is a normal approximation because it uses the exact = FALSE instruction. The command has assumed mu = 0because it is not specified explicitly.

## 13. Formula Syntax and Subsetting Samples in the u-test in R

It is better to have data arranged into a data frame where one column represents the response variable and another represents the predictor variable. In this case, the formula syntax can be used to describe the situation and carry out the **wilcox.test()**command on your data. The method is similar to what is used for the t-test.

The basic form of the command is:

wilcox.test(response ~ predictor, data = my.data)

You can also use additional instructions as you could with the other syntax. If the predictor variable contains more than two samples, you cannot conduct a u-test and must use a subset that contains exactly two samples. The subset instruction works as:

wilcox.test(response ~ predictor, data = my.data, subset = predictor %in% c("sample1", "sample2"))

Notice that in the preceding command, the names of the samples must be specified in quotes in order to group them together. The u-test is one of the most widely used statistical methods, so it is important to be comfortable using the wilcox.test()command. In the following activity, you try conducting a range of u-tests for yourself. The u-test is a useful tool for comparing two samples and is one of the most widely used of all simple statistical tests. Both the t.test()and wilcox.test()commands can also deal with matched-pair data.

## 14. Paired t- and u-tests in R

If you have a situation in which you have paired data, for example, a dataset containing information of marks of students before and after training or weight of pigs, before and after one month. You can use matched pair versions of the t-test and u-test by adding paired = TRUE as an instruction to your command. It does not matter if the data is in two separate sample columns or is a response and predictor as long as you use the appropriate syntax to indicate what it needs. In fact, R will carry out a paired test even if data does not match up as pairs. It is up to you to carry out something sensible. You can use all other standard syntax and instructions.

Here, R will carry out a paired test even if data does not match up as pairs. The command is:

1. > wilcox.test(count ~ trap, data = mpd.s, paired = TRUE, exact = F)
2. > t.test(count ~ trap, data = mpd.s, paired = TRUE, mu = 1, conf. level = 0.99)

Adding paired = TRUE as an instruction to a t.test()or wilcox.test()command carries out a paired version of the test. If the sample vectors are inside a data frame, you must use attach(), with(), or use the $ syntax to allow R to read the variables.

Paired tests are useful and more sensitive than their unpaired cousins. Since paired tests are done by comparing case by case values. However, when using them, make sure it selects the appropriate test since all data in a data frame will appear paired. R will look to see if the length of the vectors is the same. But if you have NA items, by default they will get removed and your result may differ from expectation.

## 15. Correlation and Covariance in R

When you have two continuous variables, you can look for a link between them. This link is called a correlation.

The **cor()** command determines correlations between two vectors, all the columns of a data frame, or two data frames. The **cov()** command examines covariance. The **cor.test()** command carries out a test of significance of the correlation.

You can add a variety of additional instructions to these commands, as given below:

* **cor(x, y = NULL)** – It carries out a basic correlation between x and y. If x is a matrix or data frame, we can omit y. one can correlate any object against any other object as long as the length of the individual vectors matches up.
* **cov(x, y = NULL)** – It determines covariance between x and y. If x is a matrix or data frame, one can omit y.
* **cov2cor(V)** – It takes a covariance matrix V and calculates the correlations.
* **method =** – The default is “pearson”, but “spearman” or “kendall” can be specified as the methods for correlation or covariance. These can be abbreviated but you still need the quotes, and note that they are lowercase.
* **var(x, y = NULL)** – It determines the variance of x. If x is a matrix or data frame or y is specified, It also determines the covariance.
* **test(x, y)** – It carries out a significance test of the correlation between x and y. In this case, you can now specify only two data vectors, but you can use a formula syntax, which makes it easier when the variables are within a data frame or matrix. The Pearson product moment is the default, but it can also use Spearman’s Rho or Kendall’s Tau tests . You can use the subset command to select data on the basis of a grouping variable.
* **alternative = “two.sided”** – The default is for a two-sided test but the alternative hypothesis can be given as “two.sided”, “greater”, or “less” and abbreviations are
* **level = 0.95** – If the method = “pearson” and n > 3, it will show the confidence intervals. This instruction sets the confidence level and defaults to 0.95.

## 16. Simple Correlation in R

Simple correlations are between two continuous variables and use the **cor()**command to obtain a correlation coefficient, as shown in the following command:

1. > count = c(9,25,15,2,14,25,24,47)
2. > speed = c(2,3,5,9,14,24,29,34)
3. > cor(count, speed)
4. [1] .7237206

The default for R is to carry out the Pearson product moment, but you can specify other correlations using the method = instruction, as shown in the following command:

1. > cor(count, speed, method = 'spearman')
2. [1] .5269556

This example used the Spearman Rho correlation but you can also apply kendall’s tau by specifying method = ″kendall″. Note that you can abbreviate this but you still need the quotes. You also have to use lowercase.

If your vectors are within a data frame or some other object, you need to extract them in a different fashion.

## 17. Covariance in R

The cov() command uses syntax similar to the cor() command to examine covariance.

We can use the cov() command as:

> cov(women)

The cov2cor() command determines the correlation from a matrix of covariance, as shown in the following command:

> cov2cor(women.cv)

## 18. Significance Testing in Correlation Tests

You can apply a significance test to your correlations by using the**cor.test()**command. In this case, you can compare only two vectors at a time, as shown in the following command:

> cor.test(women$height, women$weight)

Pearson’s product-moment correlation

1. data: women$height **and** women$weight
2. t = 37.8553, df = 13, p-value = 1.088e-14

alternative hypothesis: true correlation is not equal to 0

95 percent confidence interval: 0.9860970 0.9985447

sample estimates: cor 0.9954948

In the previous example, you can see that the Pearson correlation is between height and weight in the data of women and the result also shows the statistical significance of the correlation.

## 19. Formula Syntax in R

If your data is in a data frame, using the **attach()** or**with()** command is tedious, as is using the $ syntax. A formula syntax is available as an alternative, which provides a neater representation of your data, as shown in the following command:

1. > data(cars)
2. > cor.test(~ speed + dist, data = cars, method = 'spearman', exact = F)

Spearman’s rank correlation rho

1. data: speed **and** dist
2. S = 3532.819, p-value = 8.825e-14

alternative hypothesis: true rho is not equal to 0 sample estimates: rho

0.8303568

Here you examine the data of cars, which comes built into R. The formula is slightly different from the one that you used previously. Here you specify both variables to the right of the ~. You also give the name of the data as a separate instruction. All the additional instructions are available when using the formula syntax as well as the subset instruction. If your data contains a separate grouping column, you can specify the samples to use from it by using an instruction along the following commands:

Subset = grouping %in% .sample

## 20. Tests for Association in R

When you have categorical data, you can look for associations between categories by using the chi-squared test. Routines to achieve this is possible by using the**chisq.test()** command.

The various additional instructions that you can add to the chisq.test() command are:

* **test(x, y = NULL)** – A basic chi-squared test is carried out on a matrix or data frame. If it provides x as a vector, a second vector can be supplied. If x is a single vector and y is not given, a goodness of fit test is carried out.
* **correct = TRUE** – It applies Yates’ correction if the data forms a 2 n 2 **contingency table.**
* **p =** – It is a vector of probabilities for use with a goodness of fit test. If p is not given, the goodness of fit tests that the probabilities are all equal.
* **p = FALSE** – If TRUE, p is rescaled to sum to 1. For use with the goodness of fit tests.
* **p.value = FALSE** – If set to TRUE, a Monte Carlo simulation calculates p-values.
* **B = 2000** – The number of replicates to use in the Monte Carlo simulation.

## 21. Goodness of Fit Tests in R

While fitting a statistical model for observed data, an analyst must identify how accurately the model analysis the data. This is done with the help of chi-square test.  
The chi-square test is a statistical test that identifies the goodness-of-fit by testing whether the observed data is taken from the claimed distribution or not. The two values included in this test are observed value, the frequency of a category from the sample data, and expected frequency that is calculated on the basis of an expected distribution of sample population. The chisq.test() command can be used to carry out a goodness of fit test.

In this case, you must have two vectors of numerical values, one representing the observed values and the other representing the expected ratio of values. The goodness of fit tests the data against the ratios you specified. If you do not specify any, the data is tested against equal probability.

The basic form of the **chisq.test()** command will operate on a matrix or data frame.

By enclosing the command completely within parentheses, you can get the result object to display immediately. The results of many commands are stored as a list containing several elements, and you can see what is available by using the **names()**command and view them by using the **$syntax**.

The p-value can be determined using a Monte Carlo simulation by using the simulate.p.value and B instructions. If the data form a 2 n 2 contingency, then Yates’ correction is automatically applied but only if the Monte Carlo simulation is not used.

To conduct a goodness of fit test, you must specify p, the vector of probabilities; if this does not add to 1, you will get an error unless you use **rescale.p = TRUE**. You can use a Monte Carlo simulation on a goodness of fit test. If a single vector is specified, a goodness of fit test is carried out but the probabilities are assumed to be equal

# Visualize your data and compute one-way ANOVA in R

## Import your data into R

1. **Prepare your data** as specified here: Best practices for preparing your data set for R
2. **Save your data** in an external .txt tab or .csv files
3. **Import your data into R** as follow:

# If .txt tab file, use this

my\_data <- read.delim(file.choose())

# Or, if .csv file, use this

my\_data <- read.csv(file.choose())

Here, we’ll use the built-in R data set named PlantGrowth. It contains the weight of plants obtained under a control and two different treatment conditions.

my\_data <- PlantGrowth

## Check your data

To have an idea of what the data look like, we use the the function **sample\_n**()[in **dplyr** package]. The **sample\_n**() function randomly picks a few of the observations in the data frame to print out:

# Show a random sample

set.seed(1234)

dplyr::sample\_n(my\_data, 10)

weight group

19 4.32 trt1

18 4.89 trt1

29 5.80 trt2

24 5.50 trt2

17 6.03 trt1

1 4.17 ctrl

6 4.61 ctrl

16 3.83 trt1

12 4.17 trt1

15 5.87 trt1

In R terminology, the column “group” is called factor and the different categories (“ctr”, “trt1”, “trt2”) are named factor levels. **The levels are ordered alphabetically**.

# Show the levels

levels(my\_data$group)

[1] "ctrl" "trt1" "trt2"

If the levels are not automatically in the correct order, re-order them as follow:

my\_data$group <- ordered(my\_data$group,

levels = c("ctrl", "trt1", "trt2"))

It’s possible to compute summary statistics (mean and sd) by groups using the **dplyr** package.

* Compute summary statistics by groups - count, mean, sd:

**library**(dplyr)

group\_by(my\_data, group) %>%

summarise(

count = n(),

mean = mean(weight, na.rm = TRUE),

sd = sd(weight, na.rm = TRUE)

)

Source: local data frame [3 x 4]

group count mean sd

(fctr) (int) (dbl) (dbl)

1 ctrl 10 5.032 0.5830914

2 trt1 10 4.661 0.7936757

3 trt2 10 5.526 0.4425733

## Visualize your data

* To use R base graphs read this: R base graphs. Here, we’ll use the **ggpubr** R package for an easy ggplot2-based data visualization.
* Install the latest version of ggpubr from GitHub as follow (recommended):

# Install

**if**(!**require**(devtools)) install.packages("devtools")

devtools::install\_github("kassambara/ggpubr")

* Or, install from CRAN as follow:

install.packages("ggpubr")

* Visualize your data with ggpubr:

# Box plots

# ++++++++++++++++++++

# Plot weight by group and color by group

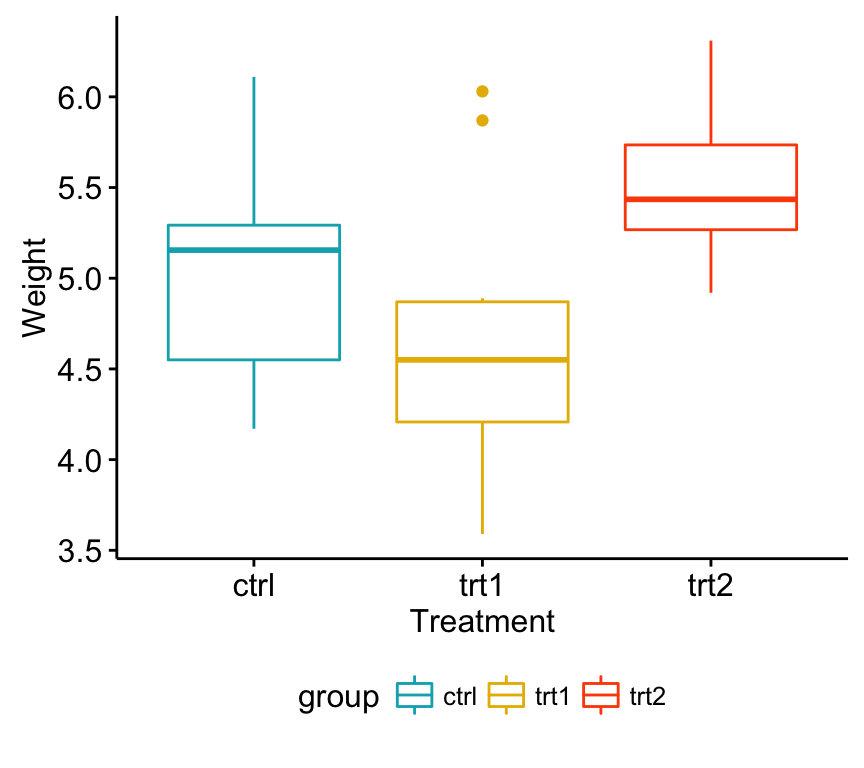
**library**("ggpubr")

ggboxplot(my\_data, x = "group", y = "weight",

color = "group", palette = c("#00AFBB", "#E7B800", "#FC4E07"),

order = c("ctrl", "trt1", "trt2"),

ylab = "Weight", xlab = "Treatment")



# Mean plots

# ++++++++++++++++++++

# Plot weight by group

# Add error bars: mean\_se

# (other values include: mean\_sd, mean\_ci, median\_iqr, ....)

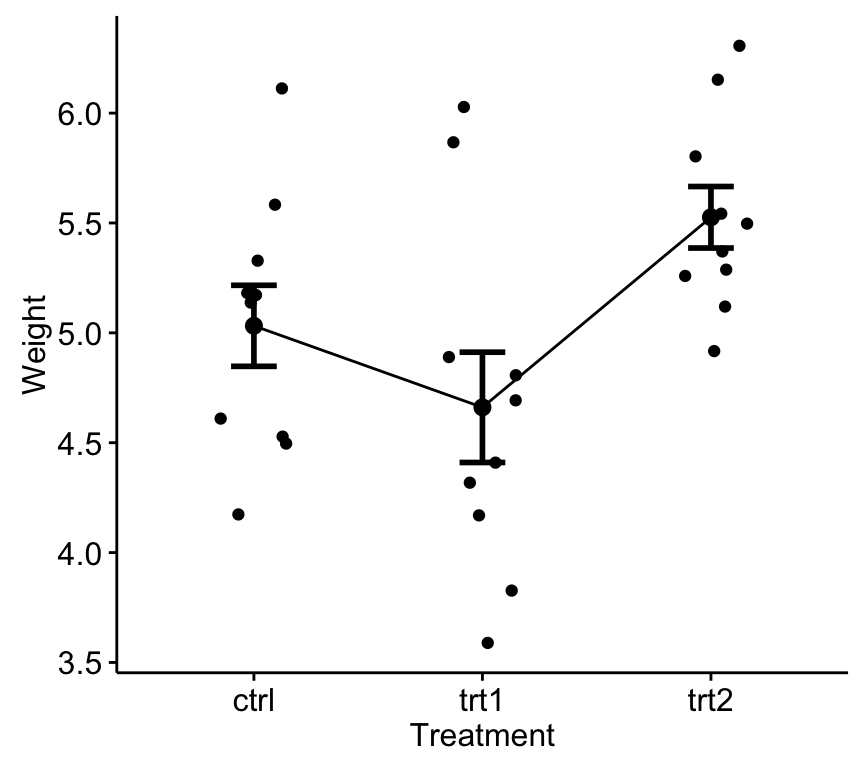
**library**("ggpubr")

ggline(my\_data, x = "group", y = "weight",

add = c("mean\_se", "jitter"),

order = c("ctrl", "trt1", "trt2"),

ylab = "Weight", xlab = "Treatment")



If you still want to use R base graphs, type the following scripts:

# Box plot

boxplot(weight ~ group, data = my\_data,

xlab = "Treatment", ylab = "Weight",

frame = FALSE, col = c("#00AFBB", "#E7B800", "#FC4E07"))

# plotmeans

**library**("gplots")

plotmeans(weight ~ group, data = my\_data, frame = FALSE,

xlab = "Treatment", ylab = "Weight",

main="Mean Plot with 95% CI")

## Compute one-way ANOVA test

We want to know if there is any significant difference between the average weights of plants in the 3 experimental conditions.

The R function **aov**() can be used to answer to this question. The function **summary.aov**() is used to summarize the analysis of variance model.

# Compute the analysis of variance

res.aov <- aov(weight ~ group, data = my\_data)

# Summary of the analysis

summary(res.aov)

Df Sum Sq Mean Sq F value Pr(>F)

group 2 3.766 1.8832 4.846 0.0159 \*

Residuals 27 10.492 0.3886

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

The output includes the columns F value and Pr(>F) corresponding to the p-value of the test.

## Interpret the result of one-way ANOVA tests

As the p-value is less than the significance level 0.05, we can conclude that there are significant differences between the groups highlighted with “\*" in the model summary.

## Multiple pairwise-comparison between the means of groups

In one-way ANOVA test, a significant p-value indicates that some of the group means are different, but we don’t know which pairs of groups are different.

It’s possible to perform multiple pairwise-comparison, to determine if the mean difference between specific pairs of group are statistically significant.

### Tukey multiple pairwise-comparisons

As the ANOVA test is significant, we can compute **Tukey HSD** (Tukey Honest Significant Differences, R function: **TukeyHSD**()) for performing multiple pairwise-comparison between the means of groups.

The function **TukeyHD**() takes the fitted ANOVA as an argument.

TukeyHSD(res.aov)

Tukey multiple comparisons of means

95% family-wise confidence level

Fit: aov(formula = weight ~ group, data = my\_data)

$group

diff lwr upr p adj

trt1-ctrl -0.371 -1.0622161 0.3202161 0.3908711

trt2-ctrl 0.494 -0.1972161 1.1852161 0.1979960

trt2-trt1 0.865 0.1737839 1.5562161 0.0120064

* **diff**: difference between means of the two groups
* **lwr**, **upr**: the lower and the upper end point of the confidence interval at 95% (default)
* **p adj**: p-value after adjustment for the multiple comparisons.

It can be seen from the output, that only the difference between trt2 and trt1 is significant with an adjusted p-value of 0.012.

### Multiple comparisons using multcomp package

It’s possible to use the function **glht**() [in **multcomp** package] to perform multiple comparison procedures for an ANOVA. **glht**stands for general linear hypothesis tests. The simplified format is as follow:

glht(model, lincft)

* **model**: a fitted model, for example an object returned by **aov**().
* **lincft**(): a specification of the linear hypotheses to be tested. Multiple comparisons in ANOVA models are specified by objects returned from the function **mcp**().

Use glht() to perform multiple pairwise-comparisons for a one-way ANOVA:

**library**(multcomp)

summary(glht(res.aov, linfct = mcp(group = "Tukey")))

Simultaneous Tests for General Linear Hypotheses

Multiple Comparisons of Means: Tukey Contrasts

Fit: aov(formula = weight ~ group, data = my\_data)

Linear Hypotheses:

Estimate Std. Error t value Pr(>|t|)

trt1 - ctrl == 0 -0.3710 0.2788 -1.331 0.391

trt2 - ctrl == 0 0.4940 0.2788 1.772 0.198

trt2 - trt1 == 0 0.8650 0.2788 3.103 0.012 \*

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Adjusted p values reported -- single-step method)

### Pairewise t-test

The function **pairewise.t.test**() can be also used to calculate pairwise comparisons between group levels with corrections for multiple testing.

pairwise.t.test(my\_data$weight, my\_data$group,

p.adjust.method = "BH")

Pairwise comparisons using t tests with pooled SD

data: my\_data$weight and my\_data$group

ctrl trt1

trt1 0.194 -

trt2 0.132 0.013

P value adjustment method: BH

The result is a table of p-values for the pairwise comparisons. Here, the p-values have been adjusted by the Benjamini-Hochberg method.

## Check ANOVA assumptions: test validity?

The ANOVA test assumes that, the data are normally distributed and the variance across groups are homogeneous. We can check that with some diagnostic plots.

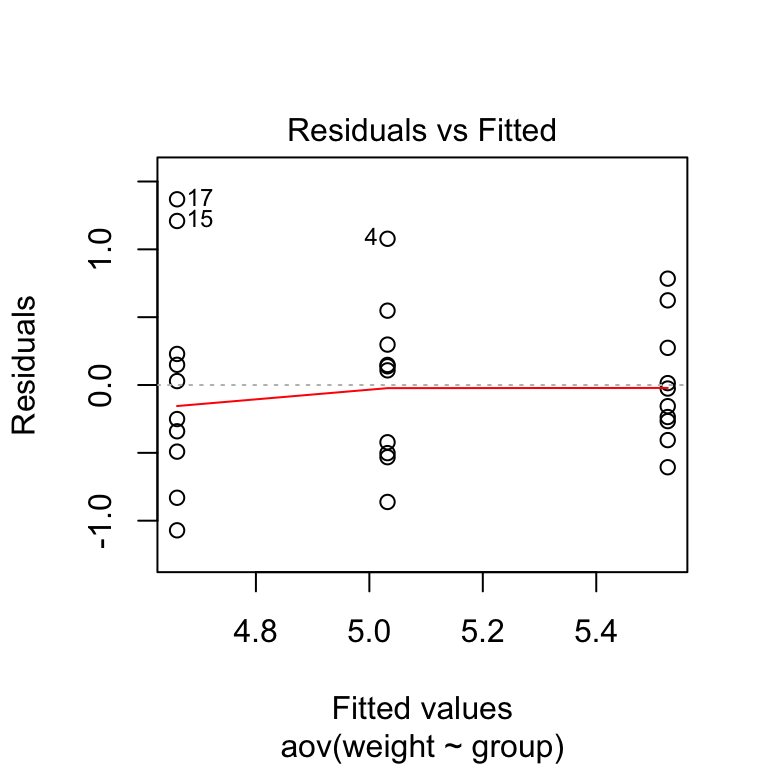
### Check the homogeneity of variance assumption

The **residuals versus fits plot** can be used to check the homogeneity of variances.

In the plot below, there is no evident relationships between residuals and fitted values (the mean of each groups), which is good. So, we can assume the homogeneity of variances.

# 1. Homogeneity of variances

plot(res.aov, 1)



Points 17, 15, 4 are detected as outliers, which can severely affect normality and homogeneity of variance. It can be useful to remove outliers to meet the test assumptions.

It’s also possible to use **Bartlett’s test** or **Levene’s test** to check the **homogeneity of variances**.

We recommend **Levene’s test**, which is less sensitive to departures from normal distribution. The function **leveneTest**() [in **car** package] will be used:

**library**(car)

leveneTest(weight ~ group, data = my\_data)

Levene's Test for Homogeneity of Variance (center = median)

Df F value Pr(>F)

group 2 1.1192 0.3412

27

From the output above we can see that the p-value is not less than the significance level of 0.05. This means that there is no evidence to suggest that the variance across groups is statistically significantly different. Therefore, we can assume the homogeneity of variances in the different treatment groups.

### Relaxing the homogeneity of variance assumption

The classical one-way ANOVA test requires an assumption of equal variances for all groups. In our example, the homogeneity of variance assumption turned out to be fine: the Levene test is not significant.

How do we save our ANOVA test, in a situation where the homogeneity of variance assumption is violated?

An alternative procedure (i.e.: **Welch one-way test**), that does not require that assumption have been implemented in the function **oneway.test**().

* **ANOVA test with no assumption of equal variances**

oneway.test(weight ~ group, data = my\_data)

* **Pairwise t-tests with no assumption of equal variances**

pairwise.t.test(my\_data$weight, my\_data$group,

p.adjust.method = "BH", pool.sd = FALSE)

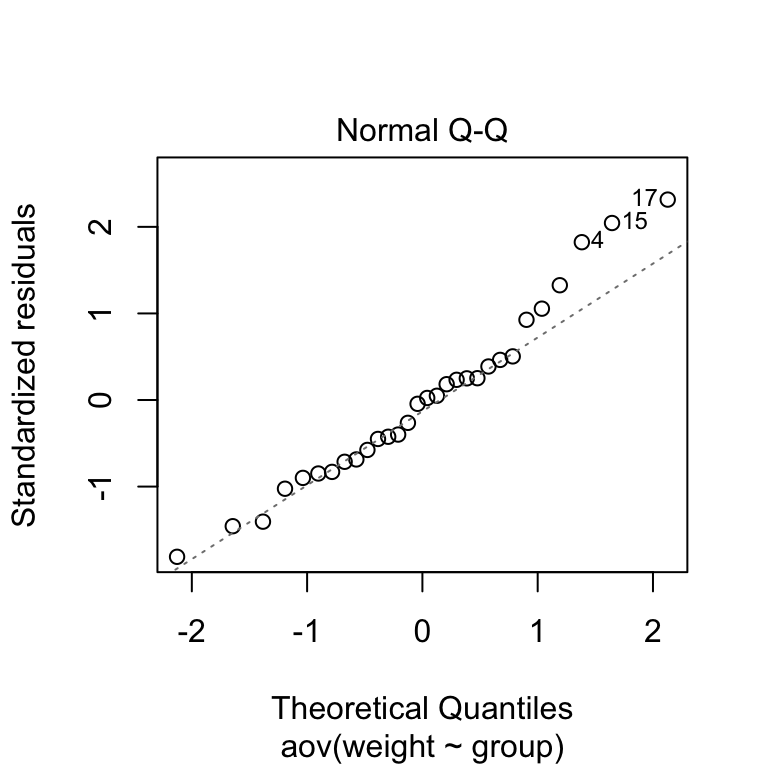
### Check the normality assumption

**Normality plot of residuals**. In the plot below, the quantiles of the residuals are plotted against the quantiles of the normal distribution. A 45-degree reference line is also plotted.

The normal probability plot of residuals is used to check the assumption that the residuals are normally distributed. It should approximately follow a straight line.

# 2. Normality

plot(res.aov, 2)



1. SCRIPT\_REAL(
2. "
3. a = aov(.arg1~.arg2, data=NULL)
4. summary(a)[[1]][[""Pr(>F)""]][1]
5. ",
6. SUM([Expression]), ATTR([Subtype])
7. )

If we wanted to calculate the correlation on a row-level, we'd need either

1) a unique identifier on each row and add this identifier to the view or

2) to select from the top menu bar deselect Analysis > Aggregate Measures.

In this case SUM() == MIN() == MAX() == ATTR(), since the aggregate is being done at the row level.

**> data <- read.table(pipe("pbpaste"), sep="\t", header=T)**

the above is for OSX; **for Windows** try:

data <- read.table("clipboard", sep="\t", header=T)

**> a = aov(data$Expression~data$Subtype, data=NULL)**

**> summary(a)**

              Df Sum Sq Mean Sq F value Pr(>F)

data$Subtype   2   4.75  2.3769   3.991 0.0196 \*

Residuals    278 165.59  0.5956

---

Signif. codes:  0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

**> summary(a)[[1]][["Pr(>F)"]][1]**

[1] 0.01955886