StepByStepStatitics

Iko koevoets

14 July 2016

This documents guides people through a (simple to a bit more complex) statistical analysis using R. It includes the script that you will use in R and some guidance in how to interpret the data. Some of my own example datasets are used to illustrate how to use the data.

Set Working directory
The first step when starting with R is ALWAYS setting your working directory. This is the location where your data is saved. You can do this by hand or you can also use the menu in R studio in the right to browse your folders and select the workingdirectory (click More, select workingdirectory).

```
setwd("~/Iko/PhD/Scripts/StepByStepStatistics_examples")
```

Load required packages
R has a basic set of packages which is installed and loaded when it starts up, but a lot of additional and very useful packages can be installed and loaded when you need them. The code below shows how to load the packages required for the code presented in this document. It is possible that you still need to install (some of) these packages when you use these for the first time. Fore example, if you need to install the multcomp package, you can use the code: install.packages("multcomp")

```
require(nlme)
require(multcomp)
require(plyr)
require(lme4)
```

Load your data
Ofcourse, the next step is to load your data into R (it should be in the folder you just chose as your working directory). Make sure that all factors (categorical variables) are factors and otherwise change them to factors (as shown for Treatment and genotypes). In the same formula you can also arrange the order by putting the levels in the order you would like them to be displayed when you are for example showing a graph. Below I have loaded one of the example datasets.

```
Data_Angle <- read.csv("Data_RootAngle.csv", sep=",")
Data_Angle$Treatment <- factor(Data_Angle$Treatment, c("Control", "Salt", "Sorbitol"))
Data_Angle$genotypes <- factor(Data_Angle$genotypes)
head(Data_Angle)
```

```
##   ID Plate Treatment genotypes Root DAG     Angle
## 1  1     1   Control         C    0  10 -1.268203
## 2  2     1   Control         C    1  10 -8.403155
## 3  3     1   Control         A    2  10  9.248349
## 4  4     1   Control         A    3  10 -5.858582
## 5  9     3      Salt         B    0  10 14.106252
## 6 10     3      Salt         B    1  10 -8.238575
```

NOTE: It is important that your data is in the right format. It should be in a csv file, which is ordered as "long" format. This means you have a column for sample name, a column for treatment, a column for genotype and a column for each trait. See the example.

Choose the appropriate model
All statistics is based on modeling. The idea is that we try to find a statistical model, which approximates reality as good as possible and can be used to make predictions about reality. The model is a mathematical equation which holds the parameters explaining the variation observed in the data. For statistics, several standardized models are available, which all have their own assumptions. When choosing a model it is important to consider the purpose of the model and whether the underlying assumptions are met. The first step to choosing your model is to consider what kind of data you have:
Is your response variable (trait of interest) continuous (e.g. main root length, lipid concentration)
Is your response variable (trait of interest) discrete (e.g. number of open stomata, survivalrates)

Continous Data

When considering continuous data, there are two most commonly used models: linear models and linear mixed models. The biggest difference between these models lies in the fact that linear mixed models includes random factors: these are factors that are not controlled by the researchers, but that are explaining part of the variation in your response variable (trait of interest). For example, this can be in which tray your plant was (when using different trays). A well known example of a random factor is "Block", when using a randomised block design. We use a special type of linear mixed models for timeseries, because time is actually correlated (differences between timepoint 1 and 2 are often smaller then between 1 and 4). Below, linear models, linear mixed models and timeseries analysis are discussed.

linear models
Linear models are based on a formula of fixed factors, with a continuous response variable (trait of interest) and one or more explanatory variables (regressors) *Linear regression*: ALL explanatory variables are quantitative (eg regression between area and dryweight of plant) *ANOVA*: All explanatory variables are qualitative (eg Genotype, treatment etc) *ANCOVA*: Combination of qualitative and quantitative All types are analysed in a similar way, only for post-hoc comparison and graphs we need to take these differences into account. We use the lm function from the stats package (always loaded). When defining the model, the formula is based on: trait ~ explanatory variable1 * explanatory variable2. Add all variables and possible interactions (except when you have a reason why a certain interaction is not interesting).

```
Model <- lm(formula = Angle ~ Treatment * genotypes, data=Data_Angle)
anova(Model)
```

```
## Analysis of Variance Table
##
## Response: Angle
##                       Df Sum Sq Mean Sq F value Pr(>F)
## Treatment              2 4813.1 2406.54 69.0269 <2e-16 ***
## genotypes              3  434.8  144.92  4.1568 0.0069 **
## Treatment:genotypes    6 4110.5  685.08 19.6501 <2e-16 ***
## Residuals            212 7391.1   34.86
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In this case, all explanatory variables are significant and we can continue with further analysis. However, sometimes certain variables are not significantly explaining the variation in the trait of interest. It is in that case best to discard these variables from the formula and rerun the model. The idea is to use a model that is as simple as possible, but sufficient. Putting more variables in will always increase the amount of variation explained, but when not significant, it also causes noise.

After defining your model, the next step is always to check the model assumptions, see chapter "checking model assumptions".

linear mixed models
Whereas linear models only consider fixed factors explaining variation, linear mixed models also consider random factors. The variation due to random factors is not controlled by researched, but does influence your outcome. If you have knowledge on these kind of factors influencing your dataset, you can implement them by using a linear mixed model, which corrects for these factors. For example, if you use different trays to grow your plants in, correct for the tray as for example amount of water/soil/light could influence the outcome of your experiment.

To define a linear mixed model, we use the lme function from the nlme package. We will again use the Data_Angle dataset, but now correct for the plate the plants were growing on.

NOTE:Although a standard ANOVA is not possible for linear mixed models, we can use the same comment in R to get the desired p-values. What R does for us, is to drop a value of the model and compare it with the full model (so for example compare Angle~Treatment + genotypes with Angle~Treatment*genotypes to get the output for interaction). It is best to define these models yourself and compare them (by using anova(model1, model2)), but for the sake of simplicity, the anova function directly on the full model is enough for this "tutorial".

```
Model_lme <- lme(fixed=Angle ~ Treatment * genotypes, data=Data_Angle, random = ~1|Plate)
anova(Model_lme)
```

```
##                       numDF denDF  F-value p-value
## (Intercept)               1   156 32.25465  <.0001
## Treatment                 2    56 58.33485  <.0001
## genotypes                 3   156  4.07366  0.0081
## Treatment:genotypes       6   156 19.67645  <.0001
```

In this case, not much has changed in the output (compare both tables) and there are several methods to get more information about how much influence your random factor has on your dataset. In the case of plate, I have tested it several times and it actually does not significantly influence the output. However, for random factors, I would advice including them if you have knowledge about them, because it is a source of variation you like to exclude.
Next, check the model assumptions before continuing to further analysis and interpretation.

A special case of lmm: timeseries
Above, no correlation between different levels of (random) factors is considered, but when analysing a timeseries, this is not correct. In timeseries something called "autocorrelation" exists. This has to do with the fact that output for day 1 and day 2 will be more similar then for day 1 and day 10 (for example in survival on salt stress). To include this, we use a linear mixed model, in which we define the type of correlation (in this case autoregressive process of order 1, the most used type, but if going further with this, read up on other options).

First we load the data.

```
Data_timeseries <- read.csv("Data_timeseries.csv")
head(Data_timeseries)
```

```
##   ID Root Plate Genotype Sucrose NaCl DAG     MRLd
## 1 51    2     5    Col-0 Sucrose 0 mM   5 1.112146
## 2 51    2     5    Col-0 Sucrose 0 mM   8 3.681645
## 3 51    2     5    Col-0 Sucrose 0 mM   9 5.053461
## 4 51    2     5    Col-0 Sucrose 0 mM  10 6.579656
## 5 51    2     5    Col-0 Sucrose 0 mM  11 7.750533
## 6 51    2     5    Col-0 Sucrose 0 mM  12 8.899135
```

Next we run the model. My question is in this case whether sucrose effects the response of my plant to salt stress. I am not specifically interested in the effect of the day on this response, because I know the MRL increases with day. I do want to correct for the timeseries and by doing so I compare the timeseries of sucrose with and without NaCl (at least that is the interaction). If you are interested on the effect of time, you can include it both as random factor and as fixed to include the autocorrelation structure and for the main factors also correct for the day.

```
Model_ts <- lme(fixed=MRLd ~ Sucrose*NaCl, data=Data_timeseries,
                correlation=corAR1(), random = ~1|ID/DAG)
anova(Model_ts)
```

```
##              numDF denDF   F-value p-value
## (Intercept)      1   291 1625.9437  <.0001
## Sucrose          1    54    2.0386  0.1591
## NaCl             2    54   91.6612  <.0001
## Sucrose:NaCl     2    54    1.7355  0.1860
```

As you see, sucrose actually has no main effect, because its effects depends on the level of NaCl, as is reflected in the clear interaction effect of NaCl and sucrose (which makes sense, as sucrose does not affect growth in salt stress and it does without salt). There is a main effect of NaCl, which means that this effect is also apparent without taken sucrose in account (as the effect of NaCl is also much stronger then sucrose and is apparent with and without sucrose). Next, check model assumptions before continuing with analysis and interpretation

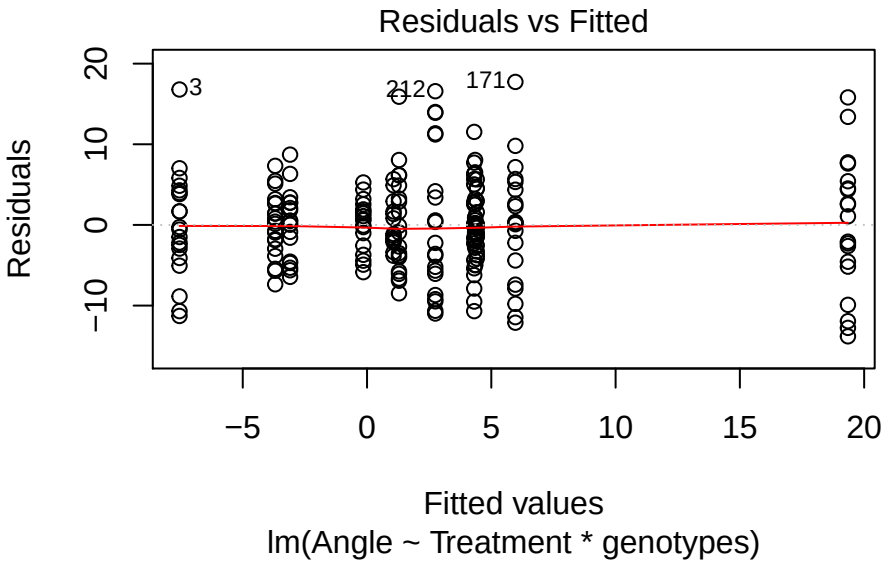
Check model Assumptions

Before further analysing your data and drawing conclusions, make sure you check the model assumptions. Basic linear models have several assumptions, with the most important being: the samples are independent (no repeated samples, timeseries etc. If you do have dependence, go to linear mixed models), the residuals (difference between observed value and value predicted by the model) have a linear pattern, a constant variance and the residuals are normally distributed.

The last two assumptions should be checked after defining the model and before continuing further analysis, if the assumptions are not met, the output is not reliable and we should either transform the data (e.g. log transformation) so the assumptions are met or use a different model. The simplest way to check the assumptions is to use the plot function on your model. The plot function outputs 4 graphs.

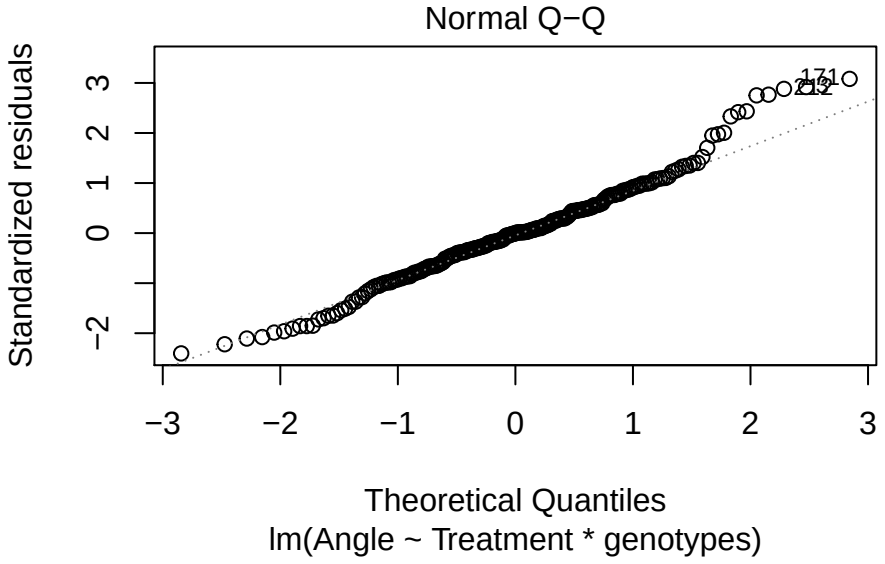
```
plot(Model)
```

Graph 1: Residuals vs Fitted
Graph 1 shows the residuals of your model vs the fitted data. The fitted values are the values predicted based on the model. As residuals=predicted-observed, a deviation from the 0 line means the value has either been predicted too low (if above) or too high (if below). If the data point is on the 0 line, the real residuals are the same as the predicted (fitted). This plot is very useful, because it gives you immediate insight in whether or not you have a linear pattern and your residual variance is constant. The amount of points above and below the line (and their values) should be similar and this should also be similar for each fitted values. On first sight, no obvious pattern in the datapoints should be visisble. For this dataset we see no pattern and approximately equal variances, so this assumption has been met. Three datapoints are pointed out (3,212,171), because they have a very high (and not comparable with other datapoints) residual. These points could be outliers, which wil be shown in next plots.



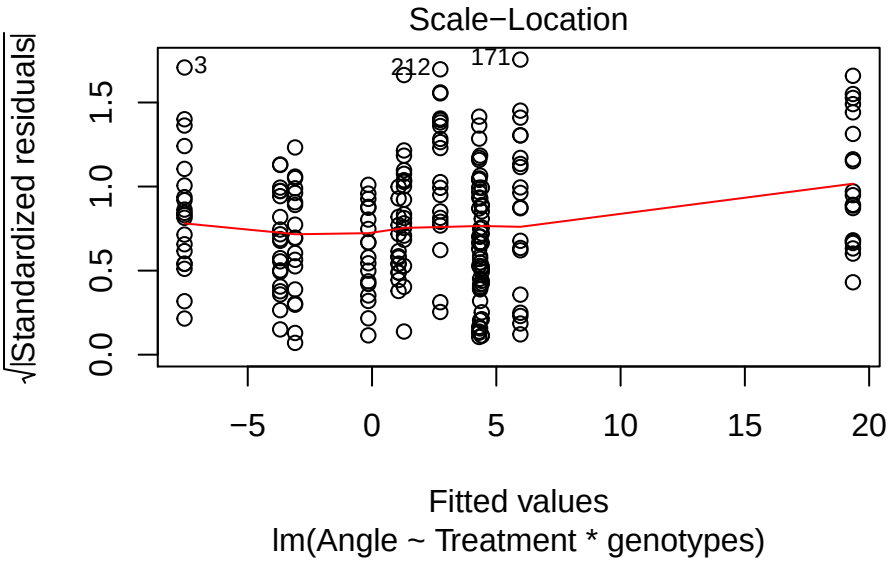
Graph 2: QQplot

Graph 2 shows the QQplot, the most used plot to determine whether the residuals follow a normal distribution, the second assumption. This plot plots the standardized residuals as a function of the theoretical quantiles. A quantile is the fraction of points below the given value. If the dataset follows a normal distribution, its quantiles (standardized residuals) should be similar to the theoretical quantiles (based on a normal distribution). This would thus follow a straight (45 degrees) line. As we can see our datapoints roughly follow this line (indicated by the dotted line), although there are some deviations at the right tail (again the same outliers are indicated). Based on this QQplot we can say that the assumption of the normal distribution is met and we can continue with the current model. If your data clearly deviates from the line (especially in the middle), then the model might not be suited or you should use some transformation (e.g log transformation), to make sure your data is following a normal distribution. Ask for help if stuck.



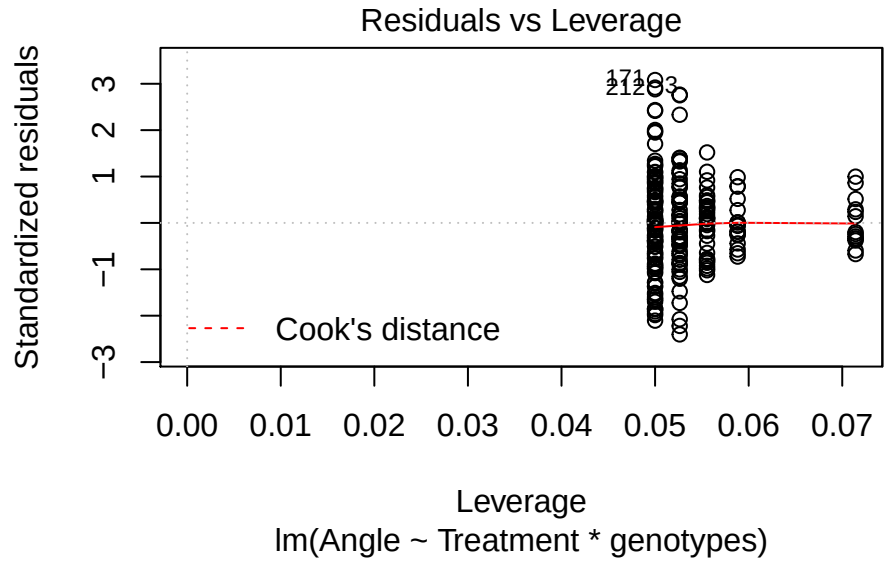
Graph 3: Scale-location

Graph 3 shows the squareroot of the residuals vs the fitted data. It is and addition to the first plot and helps you to have a better view on whether your variance is similar. The red line is the average of the datapoint for each fitted value. If this line is approximately linear your data has equal variance, as can be observed in this plot. If there is a strong relation between fitted values and the average, this means variance changes depending on the range of predictors (for example more variance in 1 genotype then the other).



Graph 4: Residuals vs Leverage

Graph 4 shows the residuals of your model vs the leverage. Leverage is value which represents how much influence the datapoint has on the model. If the leverage is high, this means that the point has a very strong influence and above a certain threshold we can consider these point as outliers. Again, 3 datapoints are regarded as outliers (the same as shown earlier). At this point, it is important to check where this could come from: maybe you made a typo in the data or maybe you can explain this because this plant has been damaged during transfer. If you have no good reason to discard the datapoint, you should not discard it. However, you should keep it in mind for further analysis.

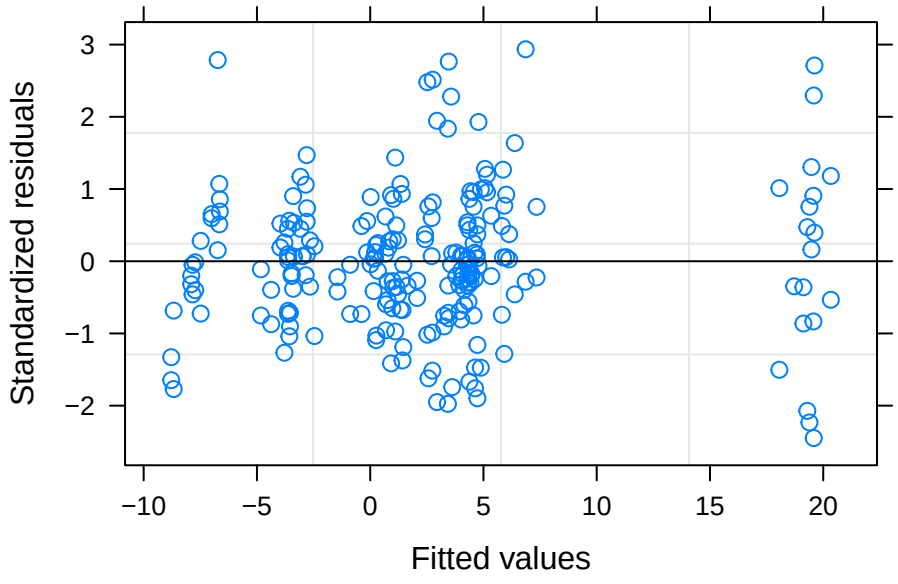


Visual checks of your data based on the above graphs and on plotting the data (use plot function or look further into the ggplot2 package if you like using R) are necessary before drawing any conclusions based on the model. It can be quite tricky to interpret these graphs and especially in the beginning it can be good to either consult other or to look for examples of these graphs on the internet. There are loads of (extreme) examples which show when assumptions are not met. If you doubt whether the used model is suited for analysis based on the presented graphs, make sure you first find out what causes the deviation: did you choose the wrong model, do you have some outliers strongly influencing your model or should you transform your data so it follows the right distribution? When these things happen, it is surely not a problem, but you do need to deal with it.

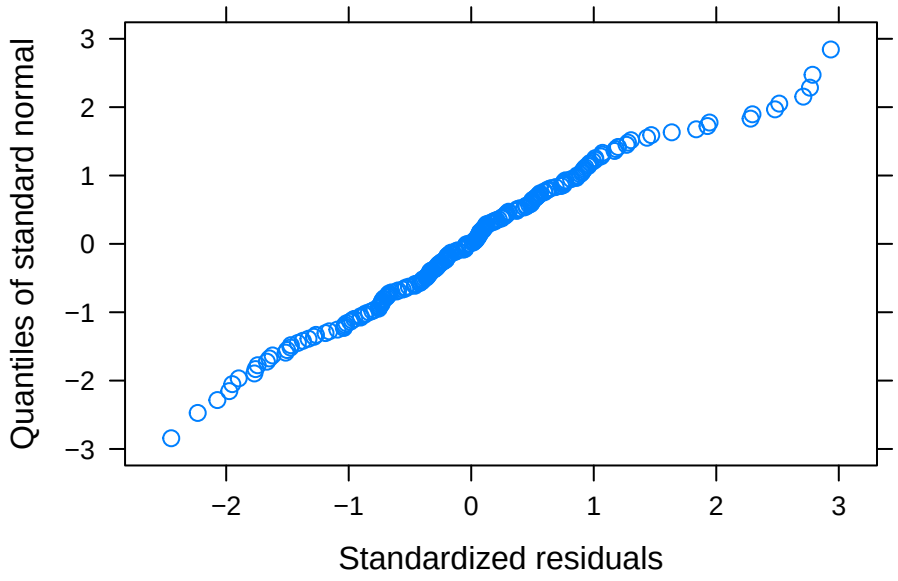
Checking assumptions of mixed models

Above function only works on basic linear models. Other models have some altered assumptions, but some checks should still be carried out. For a linear mixed model, checks for equal of variance by using a residuals plot and checks for normallity by using a qqplot can be carried out:

```
plot(Model_lme)
```




```
qqnorm(Model_lme)
```



Discrete data

Discrete data is data which has a limited number of possible values. This characteristics cuases discrete data to not follow a normal distribution. Discrete data are modelled with what we call "Generalized linear models (GLM)" in which you define the distribution your data follows. The most used distributions are the binomial distribution (for categorical traits, e.g. dead or alive) and Poisson distribution (for count data). How to analyse them in R is described below.

Binomial distribution

This distribition is suited for data which can take 2 values (0-1), for example dead or survival, open or closed etc. This data is often displayed as percentual data in graphs, but should still be analysed as binomial data, as it will not follow a normal distribution (you can check by using a QQplot, which will very obviously show a different pattern, see below). With more then two categories (e.g., flower=not visible, flower=closed, flower=open), you can also use the binomial distribution, but you always define the model as your event of interest (e.g. open flowers) vs other events (not visible or closed) and repeat this depending on the question you want to answer.

In this case we take the example where we want to compare survival of col-0 with 4 mutant lines (A to D) in salt stressed conditions in the soil (as control treatment all survived, I took it out for the sake of simplicity). Survival is noted as 0 or 1. Let's load the dataset

```
Data_survival <- read.csv("Data_survival.csv")
head(Data_survival)
```

```
##   ID Tray Position Treatment Genotype Survival
## 1  4    1       16   Control    Col-0        1
## 2  3    1       17   Control    Col-0        1
## 3  2    1       23   Control    Col-0        1
## 4  1    1       35   Control    Col-0        1
## 5  5    1       37   Control    Col-0        1
## 6 39    2        3   Control    Col-0        1
```

To use this data, we need to summarise it by the amount fo survived and amount of death per genotype/treatment. R has a useful function called ddply (in combination with summarize, from the package plyr), in which you can apply a function based on a certain grouping. You can also use it to calculate descriptive statistics such as the mean or the SE, especially useful if making graphs in R.

```
Data_summary<- ddply(Data_survival, c("Genotype", "Treatment"), summarise,
                     N= sum(!is.na(Survival)),
                     SV = mean(Survival, na.rm = TRUE),
                     se_SV = sd(Survival, na.rm = TRUE) / sqrt(N),
                     SV_YES = sum(Survival == 1),
                     SV_NO = sum(Survival == 0))
head(Data_summary)
```

```
##   Genotype Treatment  N   SV      se_SV SV_YES SV_NO
## 1        A   Control 20 1.00 0.00000000     20     0
## 2        A      Salt 20 0.65 0.10942433     13     7
## 3        B   Control 20 1.00 0.00000000     20     0
## 4        B      Salt 20 0.25 0.09933993      5    15
## 5        C   Control 20 1.00 0.00000000     20     0
## 6        C      Salt 20 0.95 0.05000000     19     1
```

We can now run the model. On the left handside of the model, we put in the death and alive per treatment & genotype. For the anova, it is important to define the test we want to use to determine the P-value (which distribution), because it is not anymore a standard anova. We define the Chi-square test, as this is the standard test for a binomial distribution

```
Model_glm <- glm(cbind(SV_YES, SV_NO)~Treatment*Genotype, data=Data_summary, family=binomial)
anova(Model_glm, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: cbind(SV_YES, SV_NO)
##
## Terms added sequentially (first to last)
##
##
##                    Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                    9     88.865
## Treatment           1   63.607         8     25.257 1.519e-15 ***
## Genotype            4   25.257         4      0.000 4.466e-05 ***
## Treatment:Genotype  4    0.000         0      0.000         1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As the interaction is not significant (all genotypes respond similarly to the treatment, which is logical as no plants die under control conditions), we will drop this term from the model:

```
Model_glm2 <- glm(cbind(SV_YES, SV_NO)~Treatment+Genotype, data=Data_summary, family=binomial)
anova(Model_glm2,test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: cbind(SV_YES, SV_NO)
##
## Terms added sequentially (first to last)
##
##
##           Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                          9     88.865
## Treatment  1   63.607         8     25.257 1.519e-15 ***
## Genotype   4   25.257         4      0.000 4.466e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As the binomial distribution has only some assumptions to how your data is collected, such as the fact that the data is independent, there are now checks we can do to see whether the assumptions are met. Most important is to only use a binomial distribution when you really have a 0-1 situation and your data is clearly discrete. See post-hoc tests for further analysis.

Poisson distribution

This distribution is suited for count data, for example the number of lateral roots. Note that often binomial data can also be noted as count data (number of surviving plant per plate, number of open stomata per leaf) and it depends on the way you designed you experiment what works best. In the end, the output will be the same, but the distribution and power of your experiment will be different. Read more about it and/or ask for help if you don't know what is best. Also check the distribution.

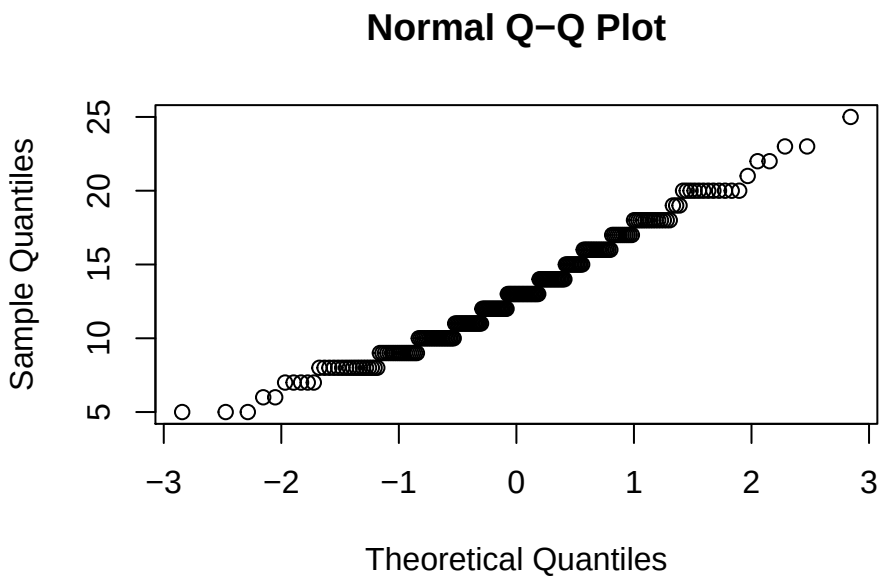
The dataset we use here is a dataset of the number of lateral roots (determined with smartroot) of 4 diffferent genotypes (Col-0, A, B, C) in salt and control conditions.

```
Data_laterals <- read.csv("Data_LR.csv")
head(Data_laterals)
```

```
##    ID Plate Treatment genotypes Root nLRd
## 1 109    32      Salt     Col-0    1    9
## 2  25    11      Salt     Col-0    0   10
## 3 279    17      Salt     Col-0    3   10
## 4 306    24  Sorbitol     Col-0    3   10
## 5 277    29      Salt     Col-0    3   10
## 6 102    30      Salt     Col-0    1   10
```

To show the problem with count data, I have plotted the lateral root data on a QQplot. As you see you get very categorized data, which makes it non-normal.

```
qqnorm(Data_laterals$nLRd)
```



Therefore, we run a poisson model:

```
Model_poisson <- glm(nLRd~Treatment*genotypes, data=Data_laterals, family=poisson)
anova(Model_poisson,test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: poisson, link: log
```

```
##
## Response: nLRd
##
## Terms added sequentially (first to last)
##
##
##                      Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                   223     267.88
## Treatment             2  124.366       221     143.51 < 2.2e-16 ***
## genotypes             3   18.381       218     125.13  0.000367 ***
## Treatment:genotypes   6   14.468       212     110.67  0.024821 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The interaction and the main effects are significant, which means we can now continue for further analysis.

Generalized linear mixed models: random effects with discrete traits
As for the continuous data, also discrete data can account for random effects. In that case, we can use a Generalized linear mixed model (GLMM). The glmer function in R covers this test. It works similar to

```
Model_poisson_mixed <- glmer(nLRd~Treatment*genotypes + (1|Plate), data=Data_laterals, family=poisson)
anova(Model_poisson_mixed,test="F-test")
```

```
## Analysis of Variance Table
##                      Df  Sum Sq Mean Sq F value
## Treatment             2 121.685  60.842 60.8424
## genotypes             3  17.317   5.772  5.7722
## Treatment:genotypes   6  14.552   2.425  2.4253
```

as this model does not directly give p-values, you can do an F-test on the F value presented in the table to calculate the p-value. In the formula you need to fill in the q, this is the F-value from the previous test, the degrees of freedom 1 (df1) and 2 (df2), which are represented by the df in the table (1, this is the amount of levels -1) and the number of samples/individuals - df1 (2).

```
#Treatment effect
pf(q=60.8424, df1=2, df2=222, lower.tail=FALSE)
```

```
## [1] 8.538549e-22
```

```
#Genotype effect
pf(q=5.7722, df1=3, df2=221, lower.tail=FALSE)
```

```
## [1] 0.0008110915
```

```
#interaction effect
pf(q=2.4253, df1=6, df2=218, lower.tail=FALSE)
```

```
## [1] 0.02732603
```

As you can see the values are roughly comparable to the ones in the linear model, as plate does not influence the outcomes much (as it should be).

Further analysis: post-hoc comparisons

After defining the model and checking the model assumptions, you should allready know whether one of your fixed factors has a significant effect on your trait. If your fixed factor is continuous, a significant interaction probably means that you have a significant correlation. If your fixed factor is discrete and has only two levels, the results are also clear: your two levels are significantly different. However, if you have more levels or if there is an interaction, you ofcourse want to know which levels are significantly differing from eachother. For this purpose we can use post-hoc comparisons. We will use the Data_Angle as an example for these post-hoc comparisons, but you can do this with every model which has been discussed above. Before choosing how to analyse the data, always think about which comparisons you want to make. Because with every extra comparison you want to make, you loose some power of your test (as there is a correction for the number of comparisons).

NOTE: when you have significant interactions, it does NOT make sense to test the main effects individually, as there is no "general" conclusion to be drawn about these levels, because they depend on the level of the other factor. SO only test interaction in that case! the post-hoc test will also give a warning in that case

preperation for significant interactions
To do a post-hoc test on an interaction, you should redefine your model, in which you put the interactionfactor as one term (the post-hoc test do not accept an interaction term). To redefine it, use the code below, it generates a new column with the interaction

```
int <- interaction(Data_Angle$genotypes,Data_Angle$Treatment)
Data_Angle <- cbind(Data_Angle, int)
Data_Angle$int <- droplevels(Data_Angle$int)
head(Data_Angle)
```

```
##   ID Plate Treatment genotypes Root DAG     Angle       int
## 1  1     1   Control         C    0  10 -1.268203 C.Control
## 2  2     1   Control         C    1  10 -8.403155 C.Control
## 3  3     1   Control         A    2  10  9.248349 A.Control
## 4  4     1   Control         A    3  10 -5.858582 A.Control
## 5  9     3      Salt         B    0  10 14.106252    B.Salt
## 6 10     3      Salt         B    1  10 -8.238575    B.Salt
```

And redefine the model (nothing should change in the output, because the terms are still the same):

```
Model <- lm(Angle~int, Data_Angle)
anova(Model)
```

```
## Analysis of Variance Table
##
## Response: Angle
##            Df Sum Sq Mean Sq F value    Pr(>F)
## int        11 9358.3  850.75  24.402 < 2.2e-16 ***
## Residuals 212 7391.1   34.86
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Tukey post-hoc: Comparing all levels with eachother
For the post-hoc comparisons we use the glht function from the multcomp package. The most done comparison is a tukey post-hoc test. However, if there is an interaction effect, this could lead to a lot of comparisons. This is not only very annoying because it prints an enormous list, but also because with every comparison,

the probability matrix is adjusted, because the more comparisons you make the bigger the chance for false positives. This therefore lowers the chance of a significant result. Often, you are actually not interested in all comparisons, solutions for that are found in a dunnett post-hoc.

```
summary(glht(Model, linfct = mcp(int = "Tukey")))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: lm(formula = Angle ~ int, data = Data_Angle)
##
## Linear Hypotheses:
##                                      Estimate Std. Error t value Pr(>|t|)
## B.Control - A.Control == 0            7.39543    1.91835   3.855    <0.01
## C.Control - A.Control == 0            4.44876    1.91835   2.319   0.4660
## Col-0.Control - A.Control == 0        3.84970    1.86718   2.062   0.6494
## A.Salt - A.Control == 0              26.89225    1.89159  14.217    <0.01
## B.Salt - A.Control == 0              10.29116    1.86718   5.512    <0.01
## C.Salt - A.Control == 0              13.51284    1.86718   7.237    <0.01
## Col-0.Salt - A.Control == 0           8.83059    1.89159   4.668    <0.01
## A.Sorbitol - A.Control == 0           8.61232    2.05754   4.186    <0.01
## B.Sorbitol - A.Control == 0          11.95872    1.94782   6.140    <0.01
## C.Sorbitol - A.Control == 0          11.85616    1.86718   6.350    <0.01
## Col-0.Sorbitol - A.Control == 0      11.89287    1.89159   6.287    <0.01
## C.Control - B.Control == 0           -2.94667    1.96818  -1.497   0.9397
## Col-0.Control - B.Control == 0       -3.54573    1.91835  -1.848   0.7885
## A.Salt - B.Control == 0              19.49682    1.94211  10.039    <0.01
## B.Salt - B.Control == 0               2.89573    1.91835   1.509   0.9364
## C.Salt - B.Control == 0               6.11741    1.91835   3.189   0.0699
## Col-0.Salt - B.Control == 0           1.43516    1.94211   0.739   0.9999
## A.Sorbitol - B.Control == 0           1.21689    2.10408   0.578   1.0000
## B.Sorbitol - B.Control == 0           4.56329    1.99692   2.285   0.4897
## C.Sorbitol - B.Control == 0           4.46073    1.91835   2.325   0.4611
## Col-0.Sorbitol - B.Control == 0       4.49744    1.94211   2.316   0.4676
## Col-0.Control - C.Control == 0       -0.59907    1.91835  -0.312   1.0000
## A.Salt - C.Control == 0              22.44349    1.94211  11.556    <0.01
## B.Salt - C.Control == 0               5.84240    1.91835   3.046   0.1027
## C.Salt - C.Control == 0               9.06408    1.91835   4.725    <0.01
## Col-0.Salt - C.Control == 0           4.38183    1.94211   2.256   0.5100
## A.Sorbitol - C.Control == 0           4.16356    2.10408   1.979   0.7070
## B.Sorbitol - C.Control == 0           7.50995    1.99692   3.761   0.0121
## C.Sorbitol - C.Control == 0           7.40739    1.91835   3.861    <0.01
## Col-0.Sorbitol - C.Control == 0       7.44411    1.94211   3.833    <0.01
## A.Salt - Col-0.Control == 0          23.04255    1.89159  12.182    <0.01
## B.Salt - Col-0.Control == 0           6.44146    1.86718   3.450   0.0321
## C.Salt - Col-0.Control == 0           9.66315    1.86718   5.175    <0.01
## Col-0.Salt - Col-0.Control == 0       4.98089    1.89159   2.633   0.2674
## A.Sorbitol - Col-0.Control == 0       4.76262    2.05754   2.315   0.4687
## B.Sorbitol - Col-0.Control == 0       8.10902    1.94782   4.163    <0.01
## C.Sorbitol - Col-0.Control == 0       8.00646    1.86718   4.288    <0.01
## Col-0.Sorbitol - Col-0.Control == 0   8.04317    1.89159   4.252    <0.01
```

```
## B.Salt - A.Salt == 0               -16.60109    1.89159  -8.776    <0.01
## C.Salt - A.Salt == 0               -13.37941    1.89159  -7.073    <0.01
## Col-0.Salt - A.Salt == 0           -18.06166    1.91569  -9.428    <0.01
## A.Sorbitol - A.Salt == 0           -18.27993    2.07971  -8.790    <0.01
## B.Sorbitol - A.Salt == 0           -14.93354    1.97123  -7.576    <0.01
## C.Sorbitol - A.Salt == 0           -15.03610    1.89159  -7.949    <0.01
## Col-0.Sorbitol - A.Salt == 0       -14.99938    1.91569  -7.830    <0.01
## C.Salt - B.Salt == 0                 3.22168    1.86718   1.725   0.8543
## Col-0.Salt - B.Salt == 0            -1.46057    1.89159  -0.772   0.9998
## A.Sorbitol - B.Salt == 0            -1.67884    2.05754  -0.816   0.9996
## B.Sorbitol - B.Salt == 0             1.66755    1.94782   0.856   0.9994
## C.Sorbitol - B.Salt == 0             1.56499    1.86718   0.838   0.9995
## Col-0.Sorbitol - B.Salt == 0         1.60171    1.89159   0.847   0.9995
## Col-0.Salt - C.Salt == 0            -4.68225    1.89159  -2.475   0.3600
## A.Sorbitol - C.Salt == 0            -4.90052    2.05754  -2.382   0.4222
## B.Sorbitol - C.Salt == 0            -1.55413    1.94782  -0.798   0.9997
## C.Sorbitol - C.Salt == 0            -1.65669    1.86718  -0.887   0.9992
## Col-0.Sorbitol - C.Salt == 0        -1.61997    1.89159  -0.856   0.9994
## A.Sorbitol - Col-0.Salt == 0        -0.21827    2.07971  -0.105   1.0000
## B.Sorbitol - Col-0.Salt == 0         3.12812    1.97123   1.587   0.9119
## C.Sorbitol - Col-0.Salt == 0         3.02556    1.89159   1.599   0.9072
## Col-0.Sorbitol - Col-0.Salt == 0     3.06228    1.91569   1.599   0.9077
## B.Sorbitol - A.Sorbitol == 0         3.34639    2.13098   1.570   0.9177
## C.Sorbitol - A.Sorbitol == 0         3.24383    2.05754   1.577   0.9154
## Col-0.Sorbitol - A.Sorbitol == 0     3.28055    2.07971   1.577   0.9149
## C.Sorbitol - B.Sorbitol == 0        -0.10256    1.94782  -0.053   1.0000
## Col-0.Sorbitol - B.Sorbitol == 0    -0.06584    1.97123  -0.033   1.0000
## Col-0.Sorbitol - C.Sorbitol == 0     0.03672    1.89159   0.019   1.0000
##
## B.Control - A.Control == 0          **
## C.Control - A.Control == 0
## Col-0.Control - A.Control == 0
## A.Salt - A.Control == 0             ***
## B.Salt - A.Control == 0             ***
## C.Salt - A.Control == 0             ***
## Col-0.Salt - A.Control == 0         ***
## A.Sorbitol - A.Control == 0         **
## B.Sorbitol - A.Control == 0         ***
## C.Sorbitol - A.Control == 0         ***
## Col-0.Sorbitol - A.Control == 0     ***
## C.Control - B.Control == 0
## Col-0.Control - B.Control == 0
## A.Salt - B.Control == 0             ***
## B.Salt - B.Control == 0
## C.Salt - B.Control == 0             .
## Col-0.Salt - B.Control == 0
## A.Sorbitol - B.Control == 0
## B.Sorbitol - B.Control == 0
## C.Sorbitol - B.Control == 0
## Col-0.Sorbitol - B.Control == 0
## Col-0.Control - C.Control == 0
## A.Salt - C.Control == 0             ***
## B.Salt - C.Control == 0
## C.Salt - C.Control == 0             ***
```

```
## Col-0.Salt - C.Control == 0
## A.Sorbitol - C.Control == 0
## B.Sorbitol - C.Control == 0         *
## C.Sorbitol - C.Control == 0         **
## Col-0.Sorbitol - C.Control == 0     **
## A.Salt - Col-0.Control == 0         ***
## B.Salt - Col-0.Control == 0         *
## C.Salt - Col-0.Control == 0         ***
## Col-0.Salt - Col-0.Control == 0
## A.Sorbitol - Col-0.Control == 0
## B.Sorbitol - Col-0.Control == 0     **
## C.Sorbitol - Col-0.Control == 0     **
## Col-0.Sorbitol - Col-0.Control == 0 **
## B.Salt - A.Salt == 0                ***
## C.Salt - A.Salt == 0                ***
## Col-0.Salt - A.Salt == 0            ***
## A.Sorbitol - A.Salt == 0            ***
## B.Sorbitol - A.Salt == 0            ***
## C.Sorbitol - A.Salt == 0            ***
## Col-0.Sorbitol - A.Salt == 0        ***
## C.Salt - B.Salt == 0
## Col-0.Salt - B.Salt == 0
## A.Sorbitol - B.Salt == 0
## B.Sorbitol - B.Salt == 0
## C.Sorbitol - B.Salt == 0
## Col-0.Sorbitol - B.Salt == 0
## Col-0.Salt - C.Salt == 0
## A.Sorbitol - C.Salt == 0
## B.Sorbitol - C.Salt == 0
## C.Sorbitol - C.Salt == 0
## Col-0.Sorbitol - C.Salt == 0
## A.Sorbitol - Col-0.Salt == 0
## B.Sorbitol - Col-0.Salt == 0
## C.Sorbitol - Col-0.Salt == 0
## Col-0.Sorbitol - Col-0.Salt == 0
## B.Sorbitol - A.Sorbitol == 0
## C.Sorbitol - A.Sorbitol == 0
## Col-0.Sorbitol - A.Sorbitol == 0
## C.Sorbitol - B.Sorbitol == 0
## Col-0.Sorbitol - B.Sorbitol == 0
## Col-0.Sorbitol - C.Sorbitol == 0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

Dunnet post-hoc: Comparing all levels with one of the levels

For the post-hoc comparisons we use the glht function from the multcomp package. The dunnett post-hoc test is specifically suited for comparing one "base level" with the rest. It normally takes the base level based on the alphabet, but you can redefine it yourself with the code below (make sure to rerun the model in that case). I often use this kind of test when I want to compare col-0 with the other mutants. Most of the times, I am not interested in comparing the mutants with eachother.

```r
Data_Angle$int <- relevel(Data_Angle$int, "Col-0.Control")
Model <- lm(Angle~int, Data_Angle)
summary(glht(Model, linfct = mcp(int = "Dunnett")))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Dunnett Contrasts
##
##
## Fit: lm(formula = Angle ~ int, data = Data_Angle)
##
## Linear Hypotheses:
##                                      Estimate Std. Error t value Pr(>|t|)
## A.Control - Col-0.Control == 0       -3.8497     1.8672  -2.062  0.26094
## B.Control - Col-0.Control == 0        3.5457     1.9183   1.848  0.38362
## C.Control - Col-0.Control == 0        0.5991     1.9183   0.312  1.00000
## A.Salt - Col-0.Control == 0          23.0426     1.8916  12.182  < 0.001
## B.Salt - Col-0.Control == 0           6.4415     1.8672   3.450  0.00633
## C.Salt - Col-0.Control == 0           9.6631     1.8672   5.175  < 0.001
## Col-0.Salt - Col-0.Control == 0       4.9809     1.8916   2.633  0.07267
## A.Sorbitol - Col-0.Control == 0       4.7626     2.0575   2.315  0.15459
## B.Sorbitol - Col-0.Control == 0       8.1090     1.9478   4.163  < 0.001
## C.Sorbitol - Col-0.Control == 0       8.0065     1.8672   4.288  < 0.001
## Col-0.Sorbitol - Col-0.Control == 0   8.0432     1.8916   4.252  < 0.001
##
## A.Control - Col-0.Control == 0
## B.Control - Col-0.Control == 0
## C.Control - Col-0.Control == 0
## A.Salt - Col-0.Control == 0         ***
## B.Salt - Col-0.Control == 0         **
## C.Salt - Col-0.Control == 0         ***
## Col-0.Salt - Col-0.Control == 0     .
## A.Sorbitol - Col-0.Control == 0
## B.Sorbitol - Col-0.Control == 0     ***
## C.Sorbitol - Col-0.Control == 0     ***
## Col-0.Sorbitol - Col-0.Control == 0 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

post-hoc on demand: defining contrasts

For the post-hoc comparisons we use the glht function from the multcomp package. Although the Dunnett test works nicely, it does have a limit to one base level. Often you might only want to compare within treatments (as you know that salt always affects Angle for example), you can then specifically define which comparisons you are interested in as shown below. This is a bit more work, but does give you the most freedom. In this case we compare A and B in different treatments, just as an example.

```r
contrasts <- c("A.Control - B.Control = 0",
               "A.Salt - B.Salt = 0",
               "A.Sorbitol - B.Sorbitol = 0"
)
summary(glht(Model, linfct = mcp(int = contrasts)))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: User-defined Contrasts
##
##
## Fit: lm(formula = Angle ~ int, data = Data_Angle)
##
## Linear Hypotheses:
##                               Estimate Std. Error t value Pr(>|t|)
## A.Control - B.Control == 0     -7.395      1.918  -3.855 0.000463 ***
## A.Salt - B.Salt == 0           16.601      1.892   8.776  < 1e-05 ***
## A.Sorbitol - B.Sorbitol == 0   -3.346      2.131  -1.570 0.312607
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

Final notes

I have written this file to help others with their statistics in R. It is not meant to be a comprehensive statistics textbook or a perfect file. See it as a tutorial you can use to learn more and start reading. Make sure you always check whether your outcomes are logical and check with others if you are doubting about a certain test. Good luck and enjoy analysing!