Advanced Research Methods - E7004

Day 3 - Linear Modelling

Dr. Fabio Veronesi

21 March 2018

Summary

- Introduction
- Assumptions of parametric tests
- ANOVA
- Statistical vs. Biological Effect
- Linear Regression
- Checking Assumptions
- Non-parametric Tests
- Generalized Linear Models (GLM)

Introduction

In the previous lecture we started talking about inferential statistics and t-test, which we can use to compare two samples. However, in the majority of cases we work with more complex designs where we need to compare different treatments, so we need more advanced tests, which will be describe here.

We will start though by taking a step back and talking about the assumptions of inferential statistics.

Assumptions of parametric tests

Generally speaking, inferential statistics works by comparing mean values and confidence intervals. As we discussed in the previous lecture, this implies computing the standard error of the mean. However, if you remember from the first day, the standard error of the mean can only be used when we have a normal distribution. In cases where the distribution is not normal we need to use quantiles. This implies that with distributions that are different from the normal, we cannot use standard statistical tests.

This is probably the most important assumption of the test we will discuss today. However, it is not the only one For example, if we want to apply ANOVA we need to check the following assumptions:

- Independence
- Normality
- Equality of variances between groups
- Balanced design

Some of these assumptions are quite strict, while other can be relaxed, particularly if we have at least 10 samples per group.

ANOVA, or analysis of variance, is a test that compares mean values from several groups based on the following equation:

$$y_j = \eta + \tau_i + \epsilon$$

where y_j is the effect of treatment τ_i on group j, η is the grand mean, i.e. the global mean of all groups, and $\epsiol$ is the error term.

Most of the assumptions are related to the error term $\episol$, which is assumed to be independent, normally distributed and have constant variance. The assumption of independence is very strict, if our data are correlated this assumption will be violated and the results of ANOVA could be biased. However, if the experiment is properly designed and fully randomized this will not be an issue. Normality and constant variance are assumptions that can be relaxed if sample size is sufficiently large. We will look at how to check both of them below.

ANOVA

This lecture will show the R code necessary to perform an ANOVA, but it will not dig too much into the theory behind the analysis. If you want to know more about it please look at the following document I wrote: ANOVA

In this lecture we will load agricultural datasets from a package named `agridat`. Please install and load it:

```r
install.packages("agridat")

library(agridat)
```

Other packages we need, and which needs to be installed, are:

```r
library(car)
library(pwr)
library(moments)
library(Rfit)
library(AER)
library(MASS)
```

From agridat we can now load the dataset lasrosas.corn, which has more that 3400 observations of corn yield (measured in quintals/ha) in a field in Argentina, plus several explanatory variables both factorial (or categorical) and continuous.

```
data(lasrosas.corn)

str(lasrosas.corn)

## 'data.frame':    3443 obs. of  9 variables:
##  $ year : int  1999 1999 1999 1999 1999 1999 1999 1999 1999 1999 ...
##  $ lat  : num  -33.1 -33.1 -33.1 -33.1 -33.1 ...
##  $ long : num  -63.8 -63.8 -63.8 -63.8 -63.8 ...
##  $ yield: num  72.1 73.8 77.2 76.3 75.5 ...
##  $ nitro: num  132 132 132 132 132 ...
##  $ topo : Factor w/ 4 levels "E","HT","LO",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ bv   : num  163 170 168 177 171 ...
##  $ rep  : Factor w/ 3 levels "R1","R2","R3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ nf   : Factor w/ 6 levels "N0","N1","N2",..: 6 6 6 6 6 6 6 6 6 6 ...
```

We will start by performing a one-way ANOVA, where the treatment structure has only one level. For this experiment we will try to explain yield (which is our dependent variable, y), with nitrogen levels (independent variable, or predictor). In R analysis of variance can be performed with the function aov:

```
One.Way = aov(yield ~ nf, data=lasrosas.corn)
```

To obtain the ANOVA table we can use the function summary:

```
summary(One.Way)

##               Df  Sum Sq Mean Sq F value   Pr(>F)    
## nf             5   23987    4797    12.4 6.08e-12 ***
## Residuals   3437 1330110     387                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As for the t-test, in the summary above we are mostly interested in the *p-value*, which tells us the level of significance of our treatment. In this case the *p-value* is very low, which means some of the groups (i.e. plots treated with particular levels of nitrogen) are statistically different from other.

We can be a bit more precise by performing multiple comparison, where we test each combination of treatments (i.e. each individual contrast):

```
TukeyHSD(One.Way, conf.level=0.95)

##   Tukey multiple comparisons of means
##     95% family-wise confidence level
## 
## Fit: aov(formula = yield ~ nf, data = lasrosas.corn)
## 
```

```
## $nf
##             diff        lwr       upr     p adj
## N1-N0 3.6434635  0.3353282  6.951599 0.0210713
## N2-N0 4.6774357  1.3606516  7.994220 0.0008383
## N3-N0 5.3629638  2.0519632  8.673964 0.0000588
## N4-N0 7.5901274  4.2747959 10.905459 0.0000000
## N5-N0 7.8588595  4.5478589 11.169860 0.0000000
## N2-N1 1.0339723 -2.2770686  4.345013 0.9489077
## N3-N1 1.7195004 -1.5857469  5.024748 0.6750283
## N4-N1 3.9466640  0.6370782  7.256250 0.0089057
## N5-N1 4.2153960  0.9101487  7.520643 0.0038074
## N3-N2 0.6855281 -2.6283756  3.999432 0.9917341
## N4-N2 2.9126917 -0.4055391  6.230923 0.1234409
## N5-N2 3.1814238 -0.1324799  6.495327 0.0683500
## N4-N3 2.2271636 -1.0852863  5.539614 0.3916824
## N5-N3 2.4958957 -0.8122196  5.804011 0.2613027
## N5-N4 0.2687320 -3.0437179  3.581182 0.9999099
```

These results provide a *p-value* for each pair of treatments. This allows us to determine which of these are different from each other. For example, `N0` is statistically different from all other nitrogen levels. However, `N1` is different from `N4` and `N5`, but not from `N2` and `N3`.

We can extract the mean values of each treatment using the following line:

```
model.tables(One.Way, type="means")
```

```
## Tables of means
## Grand mean
##          
## 69.82831 
## 
##  nf 
##         N0     N1     N2     N3     N4     N5
##      64.97  68.62  69.65  70.34  72.56  72.83
## rep 573.00 577.00 571.00 575.00 572.00 575.00
```

This function provides us with the grand mean, mean values of each treatment level and the number of replicates for each level. As you can see not all the levels were replicated the same number of times; this is therefore an unbalanced design. In this case this is not much of a problem because the number of samples is very high. However, for smaller experiments this can create issues and in some cases we cannot avoid unbalanced designs.

If this happens we cannot rely on the standard ANOVA table, but we have to compute what is called a type III test, as follows:

```
Anova(One.Way, type="III")
```

```
## Anova Table (Type III tests)
## 
## Response: yield
##              Sum Sq   Df  F value    Pr(>F)
## (Intercept) 2418907    1 6250.447 < 2.2e-16 ***
## nf            23987    5   12.396 6.075e-12 ***
## Residuals   1330110 3437
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To know more about this please look at: Types of Sums of Squares

We can investigate the effects of each treatment using the following line:

```
model.tables(One.Way, type="effects")

## Tables of effects
##
##  nf
##          N0      N1      N2       N3      N4      N5
##      -4.855  -1.212  -0.178   0.5075   2.735   3.003
## rep 573.000 577.000 571.000 575.0000 572.000 575.000
```

These values are all referred to the grand mean, meaning for example that N0 has a value that is -4.855 below the grand mean. We can verify that looking at the table of means above.

Statistical vs. Biological Effect

Sometimes there is some confusion on the meaning of *p-values*. Generally speaking, *p-values* are reported based on the probability they represent. As we mentioned in the previous lecture, significance is generally accepted at 5%, meaning that our results are considered significant only if the *p-value* is equal or below 0.05. However, *p-values* can also be smaller and in these cases we talk about highly significant differences if *p-value* is equal or below 0.01; very highly significant differences if the *p-value* is equal or below 0.001.

These values express the probabilities of incurring in a Type I error (false positive), and indirectly also the probabilities of incurring in a type II error (it is very difficult to have low power in cases where the *p-value* is very highly significant). However, a very low *p-value* does not necessarily mean that the groups we are testing have large differences. In other words, the magnitude of differences between treatments is not measured by the *p-value*, which only tells that the probabilities of obtaining the same results by chance are very low.

We can better understand this point once again with a little simulation similar to what we did in the previous lecture:

```
S1 = rnorm(n=3, mean=5, sd=2)
S2 = rnorm(n=3, mean=6, sd=2)
S3 = rnorm(n=3, mean=7, sd=2)
```

Here we are creating three samples that have differences equal to half a standard deviation. In terms of effect size their differences are equal to ES = 0.5. Therefore we are talking about relatively large differences between groups. However, the question is: can we detect these differences with a statistical tests?

Before we can do that we need to create a `data.frame` to hold these data in a format that we can then use for ANOVA. We can do that with the following code:

```
SIM.LargeEffect = expand.grid(Rep=1:3, Treatment=c("T1", "T2", "T3"))
SIM.LargeEffect
```

```
##   Rep Treatment
## 1   1        T1
## 2   2        T1
## 3   3        T1
## 4   1        T2
## 5   2        T2
## 6   3        T2
## 7   1        T3
## 8   2        T3
## 9   3        T3
```

The function `expand.grid` is extremely useful for simulations (and for many other things). It basically allows us to supply several variables (in the form of `vector` of equal or unequal length), and it then creates a `data.frame` with all the combinations of elements in each vector. In this example, we simulated an experiment with three treatments (`T1`,`T2`, and `T3`) and three replicates. The function creates a `data.frame` of $3 \times 3 = 9$ rows with each combination. At this point we can simulate dependent variables (let's call it `yield`) using the three samples we created above:

```
SIM.LargeEffect$Yield = 1:9
```

First of all, we create an additional column in the `data.frame`, called `yield`, which we fill with numbers from 1 to 9. These will act as place holders until we replace them with values from our three samples. To do so we can do some subsetting of the object `SIM.LargeEffect`:

```
SIM.LargeEffect[SIM.LargeEffect$Treatment=="T1",]$Yield = S1
SIM.LargeEffect[SIM.LargeEffect$Treatment=="T2",]$Yield = S2
SIM.LargeEffect[SIM.LargeEffect$Treatment=="T3",]$Yield = S3
```

Here we are first subsetting the object by treatment, and at the same time replacing the elements in the column `yield` (for only the treatment we have subset) with one of the samples we created above.

Let's see how the object `SIM.LargeEffect` looks now:

```
SIM.LargeEffect
```

```
##   Rep Treatment    Yield
## 1   1        T1 6.230924
## 2   2        T1 7.439317
## 3   3        T1 1.818266
## 4   1        T2  7.598228
## 5   2        T2  9.022201
## 6   3        T2  5.695456
## 7   1        T3  8.117238
## 8   2        T3 10.992257
## 9   3        T3  7.834603
```

Now that we have a dataset where we know the effect size to be exactly 0.5, we can test it using ANOVA:

```
ANOVA.LargeEffect = aov(Yield ~ Treatment, data=SIM.LargeEffect)
summary(ANOVA.LargeEffect)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## Treatment    2  22.14  11.070   2.276  0.184
## Residuals    6  29.19   4.864
```

Clearly, since this is a simulation it may be that we obtain a significant *p-value* some times. However, chances are the large majority of times the *p-value* will not be significant, despite the relatively large differences between samples. This is because the samples size is small $(n = 3)$.

On the contrary, if the sample size is large enough we can obtain very small *p-values* even in cases where differences between groups are extremely small. Let's look at the following simulation:

```
S4 = rnorm(n=7000, mean=5, sd=2)
S5 = rnorm(n=7000, mean=5.1, sd=2)
S6 = rnorm(n=7000, mean=5.2, sd=2)

SIM.SmallEffect = expand.grid(Rep=1:21000, Treatment=c("T1", "T2", "T3"))
SIM.SmallEffect$Yield = 1:21000

SIM.SmallEffect[SIM.SmallEffect$Treatment=="T1",]$Yield = S4
SIM.SmallEffect[SIM.SmallEffect$Treatment=="T2",]$Yield = S5
SIM.SmallEffect[SIM.SmallEffect$Treatment=="T3",]$Yield = S6

ANOVA.SmallEffect = aov(Yield ~ Treatment, data=SIM.SmallEffect)
summary(ANOVA.SmallEffect)
```

```
##                Df Sum Sq Mean Sq F value Pr(>F)
## Treatment       2    319  159.72   40.44 <2e-16 ***
## Residuals   62997 248808    3.95
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here we have three samples of size $n = 7000$ each, and with this amount of data finding very high significance for extremely small differences between groups is possible. This is clearly an unrealistic example; however, it clearly shows that *p-value* and differences between groups are not correlated. The conclusion is that it is always advisable to not just report the *p-value* but also the effect size (or at least a plot that clearly shows differences between groups).

Sample size for One-Way ANOVA

As we mentioned in the previous lecture, power analysis can help us determine the minimum sample size required to achieve good power, given a particular effect size. Therefore, we can try to see what would be the optimal sample size for our simulations, starting from the first ($ES = 0.5$). The function `pwr.anova.test` has a slightly different syntax compared to `pwr.t.test`. First of all we have to include the option `k` for the number of groups (in this case we have three treatments, so three groups). Then we have an option `f` for the effect size. This is another way of computing the effect size, which is simply $f = \frac{ES}{2}$; the other options are the same:

```
pwr.anova.test(k=3, f=0.25, sig.level=0.05, power=0.8)

## 
##      Balanced one-way analysis of variance power calculation 
## 
##               k = 3
##               n = 52.3966
##               f = 0.25
##       sig.level = 0.05
##           power = 0.8
## 
## NOTE: n is number in each group
```

As you can see, the required sample size (n) is much larger than our three replicates.

Now we can check the sample size for the second simulation. First of all we need to compute the effect size, which we can do with the procedure we followed in the previous lecture:

```
numerator = (mean(S5)-mean(S4))
denominator = sqrt((((length(S5)-1)*sd(S5)^2)+((length(S4)-
1)*sd(S4)^2))/(length(S5)+length(S4)-2))

ES = numerator/denominator
ES

## [1] 0.03975839
```

Now that we have the effect size we can input it in `pwr.anova.test` to compute the number of samples (remember that $f = \frac{ES}{2}$):

```
pwr.anova.test(k=3, f=ES/2, sig.level=0.05, power=0.8)
```

```
##
##      Balanced one-way analysis of variance power calculation
##
##               k = 3
##               n = 8127.787
##               f = 0.01987919
##       sig.level = 0.05
##           power = 0.8
##
## NOTE: n is number in each group
```

A Posteriori Power Analysis

So far we talked about power analysis only in cases where we need to compute the optimal sample size for our experiments. This is the most common way to use power analysis, and it referred to as *a priori* power analysis since it is performed before the experiment. However, this is not the only way to use power analysis. We can also perform it *a posteriori*, meaning after we have run the experiment. This is very valuable to better understand and interpret our experiment. For example, it may be that we run an analysis and our result suggest significant differences. Can we really be sure that our results are reliable?

In the book Statistics Done Wrong, by Alex Reinhart there is a very good explanation of the effect of considering a significance of 5% and a power of 80%. In his example Reinhard argues that even though we are assuming we are risking false positives only in 5% of cases, these may actually be much higher (around 30%). So computing the power of our experiment can allow us to achieve more robust conclusions.

Doing an *a posteriori* power analysis in R is very easy. We can simply use the same function we used above:

```
pwr.anova.test(k=3, n=3, f=0.25, sig.level=0.05)

##
##      Balanced one-way analysis of variance power calculation
##
##               k = 3
##               n = 3
##               f = 0.25
##       sig.level = 0.05
##           power = 0.07756408
##
## NOTE: n is number in each group
```

As you can see, in the line above we are using the function `pwr.anova.test` in a different way. We included the option `n`, with the number of samples per group we used in our first simulation, and excluded the option `power`, since this is what we need to compute.

As expected, results suggest our power is very low.

k-way ANOVA

If we need to perform ANOVA analysis for more complex factorial designs we can just add elements to the formula:

```
Two.Way = aov(yield ~ nf + topo, data=lasrosas.corn)
summary(Two.Way)
```

```
##                Df Sum Sq Mean Sq F value Pr(>F)    
## nf              5  23987    4797   23.21 <2e-16 ***
## topo            3 620389  206796 1000.59 <2e-16 ***
## Residuals    3434 709721     207                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
model.tables(Two.Way, type="means")
```

```
## Tables of means
## Grand mean
##          
## 69.82831 
## 
##  nf 
##         N0     N1     N2     N3     N4     N5
##      64.97  68.62  69.65  70.34  72.56  72.83
## rep 573.00 577.00 571.00 575.00 572.00 575.00
## 
##  topo 
##         E     HT     LO       W
##      78.7  48.67  84.91   66.74
## rep 730.0 785.00 885.00 1043.00
```

For multiple comparisons we need to specify which contrasts to look for:

```
TukeyHSD(Two.Way, conf.level=0.95, which=c("topo"))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
## 
## Fit: aov(formula = yield ~ nf + topo, data = lasrosas.corn)
## 
## $topo
##              diff        lwr        upr p adj
## HT-E   -30.034335 -31.934257 -28.134414     0
## LO-E     6.206619   4.359143   8.054095     0
## W-E    -11.961925 -13.745028 -10.178822     0
## LO-HT   36.240955  34.429291  38.052618     0
## W-HT    18.072411  16.326440  19.818381     0
## W-LO   -18.168544 -19.857294 -16.479794     0
```

If we do not specify the option `which`, the function will return all contrasts.

Interaction

To add an interaction term we simply need to change again the formula in the model:

```
Two.Way.Interaction = aov(yield ~ nf * topo, data=lasrosas.corn)
summary(Two.Way.Interaction)
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)    
## nf             5  23987    4797  23.176 <2e-16 ***
## topo           3 620389  206796 999.025 <2e-16 ***
## nf:topo       15   1993     133   0.642  0.842    
## Residuals   3419 707727     207                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
model.tables(Two.Way.Interaction, type="means")
```

```
## Tables of means
## Grand mean
##          
## 69.82831 
## 
##  nf 
##         N0     N1     N2     N3     N4     N5
##      64.97  68.62  69.65  70.34  72.56  72.83
## rep 573.00 577.00 571.00 575.00 572.00 575.00
## 
##  topo 
##         E     HT     LO       W
##      78.7  48.67  84.91   66.74
## rep 730.0 785.00 885.00 1043.00
## 
##  nf:topo 
##      topo
## nf    E      HT     LO     W     
##   N0   75.14  41.51  81.03  62.08
##   rep 123.00 132.00 146.00 172.00
##   N1   78.13  48.34  83.06  65.75
##   rep 125.00 138.00 145.00 169.00
##   N2   78.93  48.80  85.07  66.71
##   rep 125.00 135.00 140.00 171.00
##   N3   78.99  50.18  85.23  66.17
##   rep 119.00 128.00 153.00 175.00
##   N4   80.39  52.12  87.14  70.11
##   rep 122.00 129.00 145.00 176.00
##   N5   80.55  51.03  87.94  69.66
##   rep 116.00 123.00 156.00 180.00
```

Please notice the asterisk (`*`) separating `nf` and `topo`, which indicates that we are also interested in testing the interaction. As you can see the interaction is not significant.

Notes on Formula

A lot of statistical tests in R are based on formula, like the one we used above. So it is important to take a moment and make sure we know how to code the formula exactly for the model we want to test.

As you probably know by now, the syntax for the classic linear model is the following:

$$y = \beta_0 + \beta_1 x + \epsilon$$

The R syntax is simply: `y ~ x`.

To add elements we would simple include a `+`: `y ~ x1 + x2`. This will test the main effects for `x1` and `x2`. In some cases we are interested in testing the interaction, and the model can thus be written as: `y ~ x1 * x2`. This will test both the main effects and their interaction. If we are only interested in testing the interaction the formula will become: `y ~ x1 : x2`

With more complex models we may be interested in including a lot more terms in the equations, but only testing two-way interactions. This can be coded like so:

```
y ~ (x1 + x2 + x3)^2
```

This formula will guarantee that we do **not** test for the interaction `x1*x2*x3`, but only for interactions including two predictors.

More details about formula in R: Statistical Formula Notation in R, by chicagobooth.edu Statistical Formulas, by nature.nps.gov

ANOVA for Block Designs

For block designs the syntax to perform the ANOVA needs to account for the blocking factor. To experiment with this design we are going to load another dataset from the package `agridat`:

```
data(besag.bayesian)
str(besag.bayesian)

## 'data.frame':    225 obs. of  4 variables:
##  $ col  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ row  : int  75 74 73 72 71 70 69 68 67 66 ...
##  $ yield: num  9.29 8.16 8.97 8.33 8.66 ...
##  $ gen  : Factor w/ 75 levels "G01","G02","G03",..: 57 39 3 48 75 21 66 12
30 32 ...
```

This is randomized complete block design (the blocking factor is under `col`), and the syntax to analyse it is below:

```
besag.bayesian$col = as.factor(besag.bayesian$col)

CBD.ANOVA = aov(yield ~ col + gen, data=besag.bayesian)
summary(CBD.ANOVA)
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)    
## col           2  20.79  10.393  16.104 4.74e-07 ***
## gen          74 107.19   1.448   2.244 1.64e-05 ***
## Residuals   147  94.87   0.645                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 1 observation deleted due to missingness
```

The first line of code simply converts the column `col` from numerical to factorial. This is useful to know because in many datasets blocks are included as numbers, and therefore R reads them as numerical values.

Then we can perform the ANOVA simply by including the blocking factor first.

ANOVA for Split-Plot Designs

Split-plot is another design that is sometimes used and that needs to be treated carefully during analysis. For this example we are loading a dataset (modified) presented in a recent paper in the European Journal of Soil Science by Webster and Lark, entitled Analysis of variance in soil research: let the analysis fit the design.

The design is represented in the image below (taken from the paper):

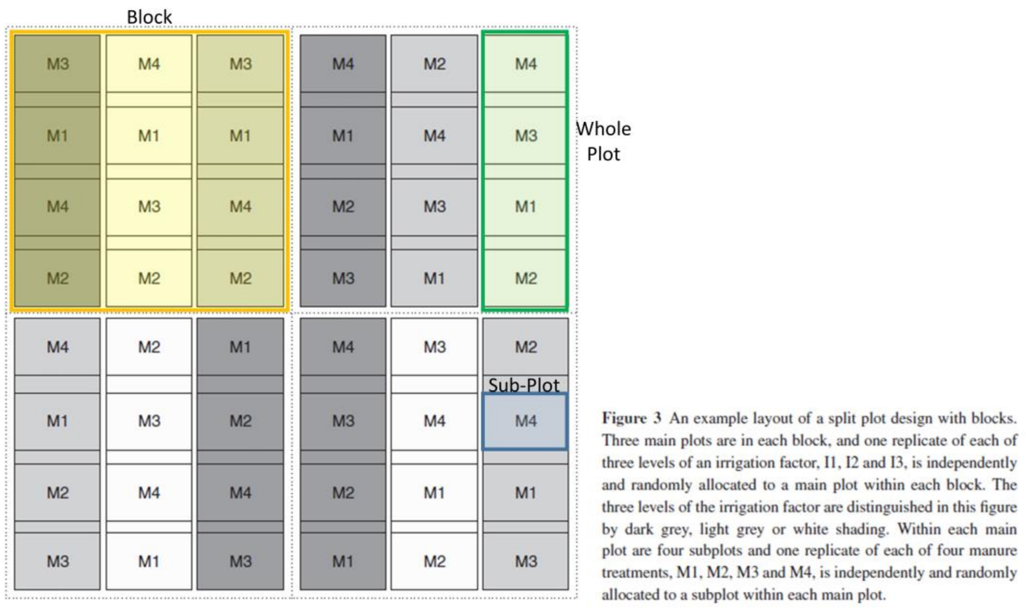


Figure 3 An example layout of a split plot design with blocks. Three main plots are in each block, and one replicate of each of three levels of an irrigation factor, I1, I2 and I3, is independently and randomly allocated to a main plot within each block. The three levels of the irrigation factor are distinguished in this figure by dark grey, light grey or white shading. Within each main plot are four subplots and one replicate of each of four manure treatments, M1, M2, M3 and M4, is independently and randomly allocated to a subplot within each main plot.

Source: Analysis of variance in soil research: let the analysis fit the design

Please download the file `exp3.csv` from the learning hub, place it in a folder of your choice and load it using the `read.csv` function:

```
soil.df = read.csv("exp3.csv",header=T)
```

An alternative way of importing the data is from my GitHub with the package `RCurl`:

```
install.packages("RCurl")

library(RCurl)

Data.URL = 
getURL("https://raw.githubusercontent.com/fveronesi/AdvancedResearchMethods/m
aster/exp3.csv")

soil.df = read.csv(text=Data.URL)
```

After the data are loaded, we can convert some columns into factors and perform the analysis with the following line:

```
soil.df$Manures<-factor(soil.df$Manures)
soil.df$Irrigation<-factor(soil.df$Irrigation)
soil.df$Blocks<-factor(soil.df$Blocks)
soil.df$Whole_Plot<-factor(soil.df$Whole_Plot)
soil.df$Split_Plot<-factor(soil.df$Split_Plot)

exp3 <-aov(Respiration_rate ~ Blocks + Irrigation*Manures + 
Error(Whole_Plot/Split_Plot), data = soil.df)

summary(exp3)

## 
## Error: Whole_Plot
##            Df Sum Sq Mean Sq
## Irrigation  2  31669   15834
## 
## Error: Whole_Plot:Split_Plot
##                    Df Sum Sq Mean Sq F value  Pr(>F)   
## Manures             3  29645    9882 110.357 0.00899 **
## Irrigation:Manures  4   1196     299   3.338 0.24357   
## Residuals           2    179      90                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Error: Within
##                    Df Sum Sq Mean Sq F value   Pr(>F)  
## Blocks              3   2349     783   2.590   0.0763 .
## Manures            3  89969   29990  99.174 1.16e-13 ***
## Irrigation:Manures  6    624     104   0.344   0.9064
## Residuals          24   7258     302
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Linear Regression

Up to now we dealt with factorial designs, for which ANOVA is the most appropriate test. However, in cases when we are dealing with continuous variables (or a mix between continuous and categorical) we need to change to linear regression.

For example, in the dataset `lasrosas.corn` there are two variables related to nitrogen amendments. The first the is the variable `nf`, which is categorical; the second is `nitro`, which is continuous. However, they are representing the same amendments, the difference is that `nitro` recorded the actual amount of nitrogen applied to each plot, while `nf` assigned a category to each level.

In the one-way ANOVA model we used the variable `nf`, and the research question underlying the test was to detect any differences between these levels. However, in some occasions we could be interested in knowing what is the impact on yield of unit increases in nitrogen. To answer this question we need to fit a different model:

```
LinReg = lm(yield ~ nitro, data = lasrosas.corn)
summary(LinReg)

## 
## Call:
## lm(formula = yield ~ nitro, data = lasrosas.corn)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -53.183 -15.341  -3.079  13.725  45.897 
## 
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 65.843213   0.608573 108.193  < 2e-16 ***
## nitro        0.061717   0.007868   7.845 5.75e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 19.66 on 3441 degrees of freedom
## Multiple R-squared:  0.01757,    Adjusted R-squared:  0.01728 
## F-statistic: 61.54 on 1 and 3441 DF,  p-value: 5.754e-15
```

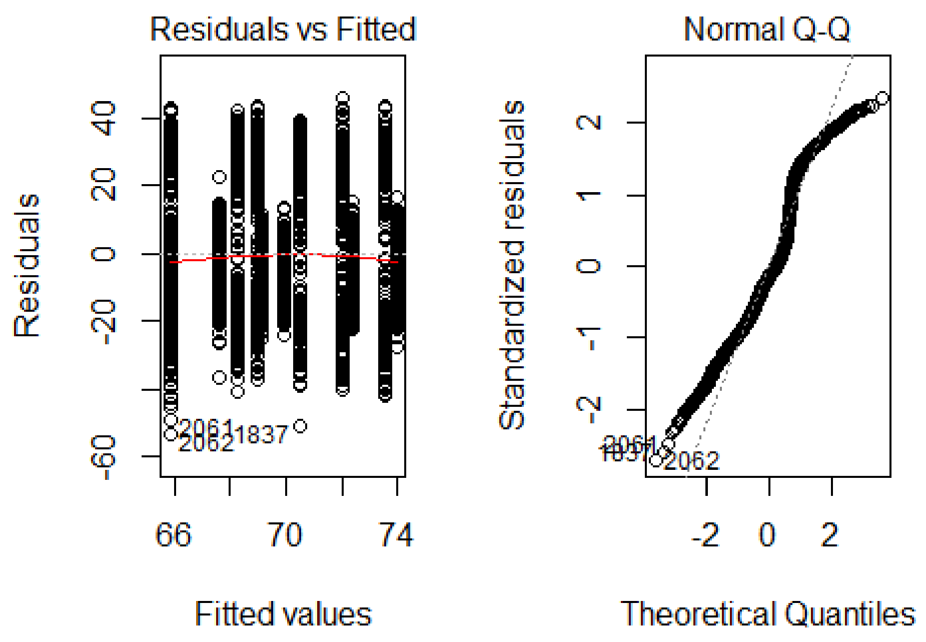
As you can see we are now using the function `lm`, which stands for linear model. The predictor now is `nitro`, which is a continuous variable. The summary provides the slope of line (0.061717), which tells us the average increase in yield for each unit increase in nitrogen. Basically, for each additional unit of nitrogen we add to the soil we would increase yield on average of 0.06 quintals/ha, or 6 kg/ha.

Checking Assumptions

As mentioned, there are some assumptions required to fit any linear model, being an ANOVA or a linear regression. In particular, normality and equality of variance are the most important.

We can check that our model complies with these assumptions using the function `plot`:

```
par(mfrow=c(1,2))
plot(LinReg, which=c(1,2))
```



The first plot on the left represents the residuals against the fitted values (or the estimates from the model). One of our assumptions is that the error term had mean of zero and constant variance. This means that we should see the residuals equally spread around zero, and a more or less horizontal line (red line) with intercept on the zero. In this case the line is very close to being straight across zero, and the spread is more or less constant throughout the range of fitted values. Therefore, we can conclude our model does **not** violate this assumption.

The second plot is the QQplot of the residuals. This plots the quantiles of the distribution of residuals against quantiles of a standard normal distribution (with mean = 0 and sd = 1).

The more our samples fit a normal distribution, the more these points should lie on a straight line with an inclination of 45 degrees. In this case it seems the quantile line is not exactly straight, so we need an additional test to determine whether we can accept normality. We can compute the skewness of the residuals, with the function from the package `moments`:

```
skewness(residuals(LinReg))

## [1] 0.4073682
```

Since the skewness is below ± 0.5 (Webster and Oliver, 2007), we can conclude that our results do **not** violate the assumption of normality either.

Non-parametric Tests

For certain datasets the assumption of normality cannot be met. In such cases we may consider different options: GLM is one of them and it should be a good solution for datasets like counts and presence/absence data (we will look at GLM below). Another option could be to transform the data and "normalize"" them to meet the assumption of normality. However, with transformations we need to be extremely careful because the estimates of the slopes will also be transformed, and so we always need to know how to back-transform our data. The final option would be to use non-parametric tests, which do not assume a normal distribution.

For the one-way ANOVA the non-parametric alternative is the Kruskal-Wallis test:

```
kruskal.test(yield ~ nf, data=lasrosas.corn)

##
##  Kruskal-Wallis rank sum test
##
## data:  yield by nf
## Kruskal-Wallis chi-squared = 81.217, df = 5, p-value = 4.669e-16
```

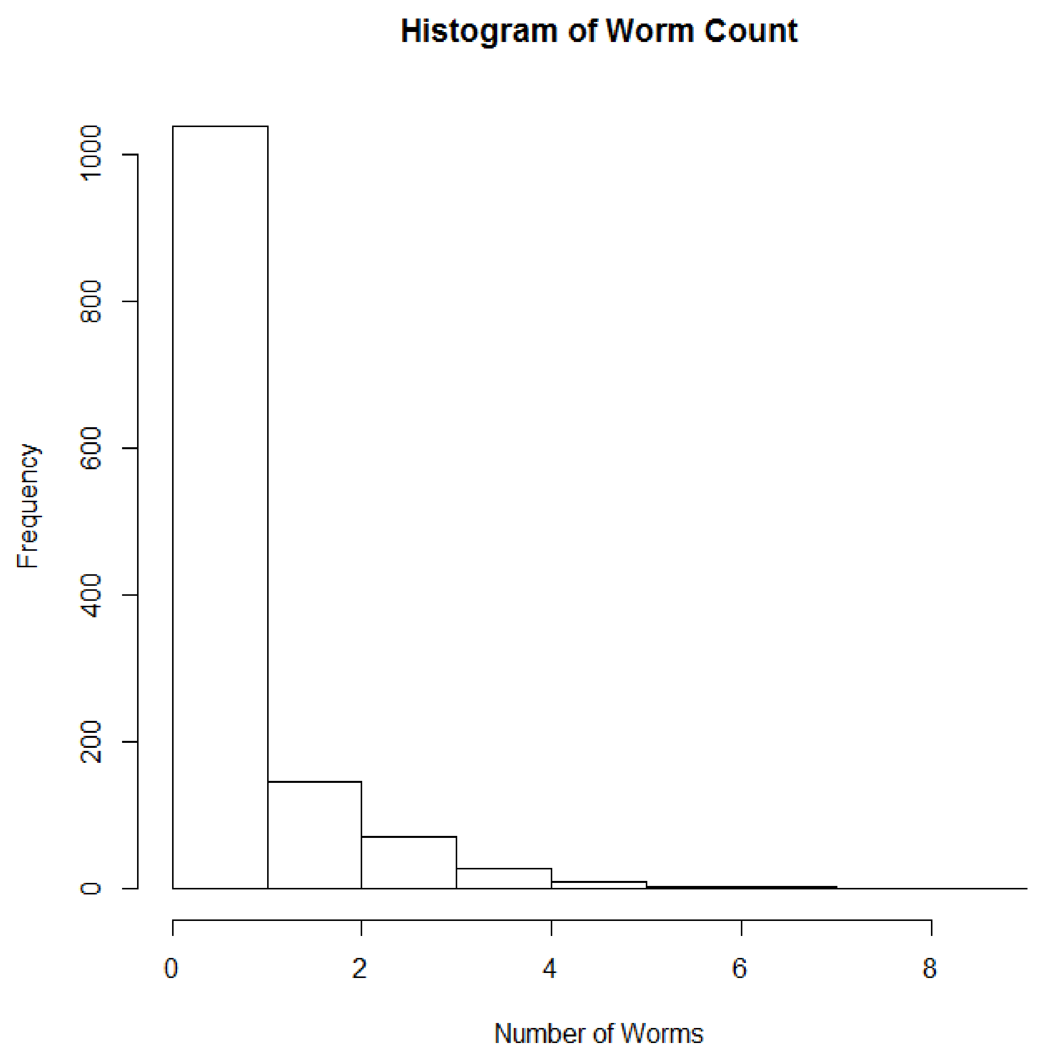
For more complex designs we can use the function `raov` from the package `Rfit`:

```
raov(yield ~ nf * topo, data=lasrosas.corn)

##
## Robust ANOVA Table
##         DF          RD    Mean RD         F p-value
## nf       5   764.56053  152.91211  21.96030 0.00000
## topo     3 17418.76333 5806.25444 833.85875 0.00000
## nf:topo 15    59.15213    3.94348   0.56634 0.90215
```

GLM - Count Data

As mentioned above, generalized linear models or GLM, can be used in cases of violation of the assumption of normality. These models can work with error distributions that do not fit a normal distribution, so theoretically they could be employed every time we are working with non-normal distributions. However, in order to apply GLM we need to know what distribution to fit to our data. In other words, knowing that the distribution is not normal is not enough, we need to know what other distribution fits our data. This is not the case for non-parametric tests, which only assume non-normality. However, non-parametric tests only allow relatively simple designs, so we need to be careful.

For some datasets though, knowing the distribution is fairly simple and therefore GLM are the preferred choice. One of these datasets is count data, e.g. number of insects, number of events per hours. These data generally fit a Poisson distribution, which looks similar to the histogram below:



Histogram of Poisson Distribution

The characteristic of the poisson distribution is that it only includes non-negative integers (i.e. whole numbers), and usually the large majority of data are close to zero.

GLM still solve linear equations, but because of the data distribution they employ a link function to "linearise" the model. In fact, GLM for count data solve the equation below:

$$\ln(y) = \beta_0 + \beta_1 x$$

For testing GLM for counts in R we are going to import another dataset from the package `agridat`:

```
data(beall.webworms)
str(beall.webworms)
```

```
## 'data.frame':    1300 obs. of  7 variables:
##  $ row  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ col  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ y    : int  1 0 1 3 6 0 2 2 1 3 ...
##  $ block: Factor w/ 13 levels "B1","B10","B11",..: 1 1 1 1 1 6 6 6 6 6 ...
##  $ trt  : Factor w/ 4 levels "T1","T2","T3",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ spray: Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ lead : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
```

This dataset represents counts of worms in a beet field, with insecticide treatments.

The syntax to fit GLM in R is very simple and follows the same formula approach we used for previous models:

```
PoisReg = glm(y ~ trt, data=beall.webworms, family=poisson(link=log))
```

As you can see the main difference between the syntax for the function `glm` as compared to `lm` is that here we need to add the options `family`, which is the family of distributions for our data, and `link`, which is the link function for this particular model.

As always we can call the functions `Anova`…

```
Anova(PoisReg)

## Analysis of Deviance Table (Type II tests)
##
## Response: y
##     LR Chisq Df Pr(>Chisq)
## trt   235.45  3  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

and `summary` to check the details of the model:

```
summary(PoisReg)

##
## Call:
## glm(formula = y ~ trt, family = poisson(link = log), data =
beall.webworms)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6733  -1.0046  -0.9081   0.6141   4.2771
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.33647    0.04688   7.177 7.12e-13 ***
## trtT2       -1.02043    0.09108 -11.204  < 2e-16 ***
## trtT3       -0.49628    0.07621  -6.512 7.41e-11 ***
## trtT4       -1.22246    0.09829 -12.438  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 1955.9  on 1299  degrees of freedom
## Residual deviance: 1720.4  on 1296  degrees of freedom
## AIC: 3125.5
##
## Number of Fisher Scoring iterations: 6
```

As for the other models we tested above, `Anova` provides a *p-value* for the whole treatment, while `summary` provides *p-values* for individual contrasts, computed with a Wald test. In this dataset, treatment has 4 levels (`T1`, `T2`, `T3` and `T4`). as you can see in the summary table, level `T1` is not shown. This is called the reference level and all the *p-values* are computed based on the comparison between them and the reference level. For example, the *p-value* for `T2` is referred to the contrast between `T1` and `T2`.

In case we need to compute *p-values* for other contrasts we can simply change the order within the variable `trt`:

```
beall.webworms$trt = relevel(beall.webworms$trt, "T4")
```

Now we have changed the reference level to `T4`, so that if we run the model again we can check *p-values* for direct comparisons with `T4`:

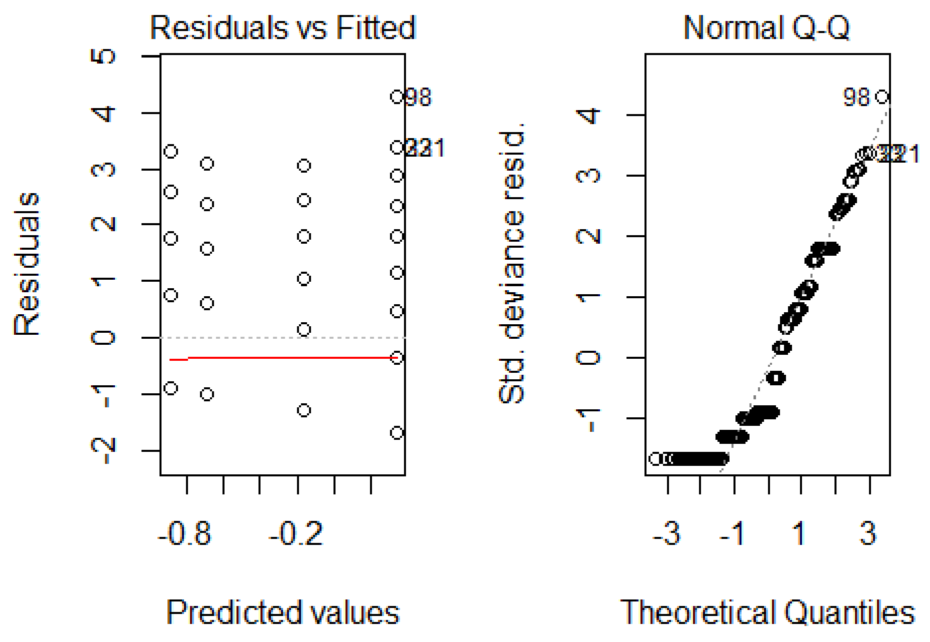
```
PoisReg = glm(y ~ trt, data=beall.webworms, family=poisson(link=log))

summary(PoisReg)

##
## Call:
## glm(formula = y ~ trt, family = poisson(link = log), data =
beall.webworms)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6733  -1.0046  -0.9081   0.6141   4.2771
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.88599    0.08639 -10.256  < 2e-16 ***
## trtT1        1.22246    0.09829  12.438  < 2e-16 ***
## trtT2        0.20203    0.11645   1.735   0.0828 .
## trtT3        0.72618    0.10523   6.901 5.16e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 1955.9  on 1299  degrees of freedom
## Residual deviance: 1720.4  on 1296  degrees of freedom
## AIC: 3125.5
##
## Number of Fisher Scoring iterations: 6
```

As you can see, the *p-values* are different. For example, the *p-value* for `T2` is now not significant. This means that the difference between `T2` and `T4` is not significant.

Checking Assumption

We can again use the function plot to produce diagnostic plots:

```
par(mfrow=c(1,2))
plot(PoisReg, which=c(1:2))
```



The interpretation is the same as for `lm`. The left plot should indicate a straight line crossing zero, and a constant spread of points. The right QQplot should present quantiles on a straight line. In this case it seems our model could be better particularly in terms of normality of the residuals.

Interpretation

To interpret the coefficients of the model we need to remember that this GLM uses a log link function. Therefore for example the -1.02 is log transformed, so the coefficient for `T2` would be `exp(-1.02) = 0.36`. In terms of interpretation, we can say that the number of worms for `T2` is 0.36 times the number of worms for `T1` (this is because the coefficient is always referred to the reference level). So there is a decrease, and that is why the coefficient is negative.

More info here: stats.stackexchange.com

Overdispersion

In some cases count data can be overdispersed, meaning that the variance of the distribution is higher that what we would expect in case of a poisson distribution. In such cases we need to change the error distribution in the model.

To assess the overdispersion we can compute both variance and mean:

```
mean(beall.webworms$y); var(beall.webworms$y)

## [1] 0.7923077

## [1] 1.290164
```

As you can see they are slightly different. If these data followed a perfect poisson distribution, these two values would be almost identical. The fact that the variance is larger than the mean implies a certain degree of overdispersion, which we can account using the "quasi-poisson" distribution:

```
QuasPois.Reg = glm(y ~ trt, data=beall.webworms,
family=quasipoisson(link=log))

summary(QuasPois.Reg)

##
## Call:
## glm(formula = y ~ trt, family = quasipoisson(link = log), data =
beall.webworms)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6733  -1.0046  -0.9081   0.6141   4.2771
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.8860     0.1005  -8.812  < 2e-16 ***
## trtT1         1.2225     0.1144  10.686  < 2e-16 ***
## trtT2         0.2020     0.1355   1.491    0.136
## trtT3         0.7262     0.1225   5.929  3.9e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 1.35472)
##
##     Null deviance: 1955.9  on 1299  degrees of freedom
## Residual deviance: 1720.4  on 1296  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 6
```

There are cases though were the variance is much larger the the mean, and a quasi-poisson would not be appropriate. In such cases we need to resort to using a “negative binomial” distribution of the error term (using a function from the package `MASS`):

```
NegBin.Reg = glm.nb(y ~ trt, data=beall.webworms)

summary(NegBin.Reg)

##
## Call:
## glm.nb(formula = y ~ trt, data = beall.webworms, init.theta = 2.004130573,
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4572  -0.9488  -0.8660   0.5340   2.7698
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.88599    0.09486  -9.340  < 2e-16 ***
## trtT1        1.22246    0.11283  10.834  < 2e-16 ***
## trtT2        0.20203    0.12896   1.567    0.117
## trtT3        0.72618    0.11893   6.106 1.02e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(2.0041) family taken to be 1)
##
##     Null deviance: 1442.7  on 1299  degrees of freedom
## Residual deviance: 1275.3  on 1296  degrees of freedom
## AIC: 3053
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  2.004
##           Std. Err.:  0.325
##
##  2 x log-likelihood:  -3042.969
```

GLM - Presence/Absence

Another popular form of regression that can be tackled with GLM is the logistic regression, where the variable of interest is binary (0 or 1, presence or absence, and any other binary outcome). In this case the regression model takes the following equation:

$$\ln\left(\frac{p(y)}{1-p(y)}\right) = \beta_0 + \beta_1 x$$

The equation is identical to the standard linear model, but what we are computing here is the log of the probability that one of the two outcomes will occur, also referred as logit function.

For this example we are loading the dataset `johnson.blight`, again available in `agridat`. Here the binary variable of interest is the presence or absence of blight (either 0 or 1) in potatoes:

```
data(johnson.blight)

str(johnson.blight)

## 'data.frame':    25 obs. of  6 variables:
##  $ year    : int  1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 ...
##  $ area    : int  0 0 0 0 50 810 120 40 0 0 ...
##  $ blight  : int  0 0 0 0 1 1 1 1 0 0 ...
##  $ rain.am : int  8 9 9 6 16 10 12 10 11 8 ...
##  $ rain.ja : int  1 4 6 1 6 7 12 4 10 9 ...
##  $ precip.m: num  5.84 6.86 47.29 8.89 7.37 ...
```

The syntax to fit a logistic regression model is very similar to what we used above:

```
LogReg = glm(blight ~ rain.am, data=johnson.blight,
family=binomial(link=logit))
```

Once again we can call `Anova`...

```
Anova(LogReg)

## Analysis of Deviance Table (Type II tests)
##
## Response: blight
##         LR Chisq Df Pr(>Chisq)
## rain.am   9.8353  1   0.001712 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

...and `summary` to get the details:

```
summary(LogReg)
```

```
##
## Call:
## glm(formula = blight ~ rain.am, family = binomial(link = logit),
##     data = johnson.blight)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9395  -0.6605  -0.3517   1.0228   1.6048
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -4.9854     2.0720  -2.406   0.0161 *
## rain.am       0.4467     0.1860   2.402   0.0163 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 34.617  on 24  degrees of freedom
## Residual deviance: 24.782  on 23  degrees of freedom
## AIC: 28.782
##
## Number of Fisher Scoring iterations: 5
```

The *p-values* can be interpreted as we described above.

Interpretation

The interpretation of estimates for a logistic regression model are a bit more complex than what we saw for count data. In fact, here we are dealing with a logit transformation and to compute the probabilities we need to solve the following:

$$p(y = 1) = \frac{\exp(\beta_0 + \beta_1 x)}{1 + \exp(\beta_0 + \beta_1 x)}$$

here β_0 is the value of the intercept (-4.9854), and β_1 is the value of the slope for `rain.am` (0.4467).

Let's say for example that we need to compute probabilities of blight if rainfall is 10 mm, we need to solve the equation above using the estimates from the model.

```
exp(-4.9854 + 0.4467 * 10)/(1 + exp(-4.9854 + 0.4467 * 10))
```

```
## [1] 0.3732264
```

Therefore the probability of blight for rain of 10 mm is 37%. If we need to compute the rate of change, i.e. changes in probabilities for each unit change in rain, we need to use a linear approximation, as suggested by Agresti (2007):

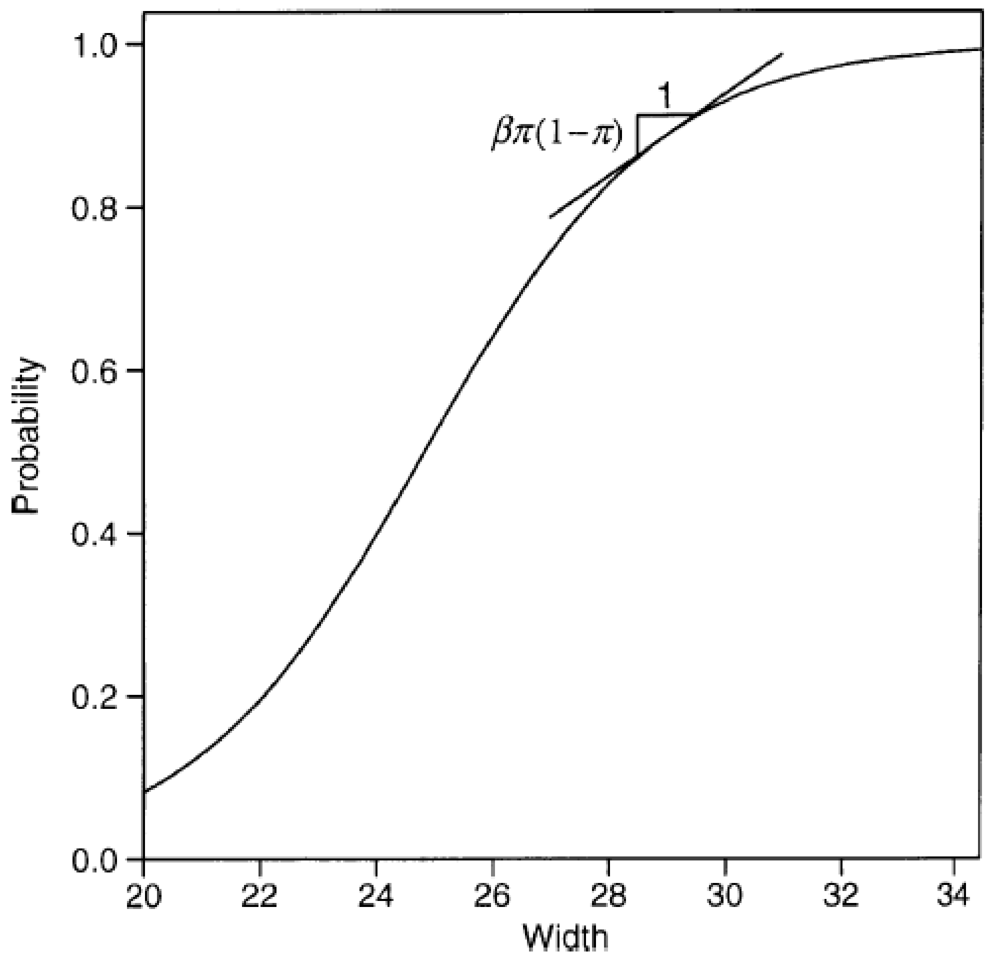


Figure 4.1. Linear approximation to logistic regression curve.

Linear Approximation

where β is the coefficient for rain (0.4467) and π is the probability we just calculated.

The code to solve that is:

```
0.4467 * 0.37 * (1 - 0.37)

## [1] 0.1041258
```

So for each 1 mm of rain the increase in probability is around 10% (allowing for differences due to the linear approximation).

Now that we know how to compute probabilities by hand, you will be happy to know that we can do all this using the function `predict` and avoid manual computations. For example, to compute the probability for `rain.am` equal 10 we can simply run:

```
predict(LogReg, newdata=data.frame(rain.am=10), type="response")
```

```
##         1
## 0.3732829
```

here `newdata` is used to tell the model which new dataset to use for prediction, and `type="response"` is the option to use to get probabilities.

If we want to know the rate of change we can simply predict two values separated by one:

```
predict(LogReg, newdata=data.frame(rain.am=c(10,11)), type="response")

##         1         2
## 0.3732829 0.4821493
```

As you can see the rate of change is around 10% (for this part of the curve).

GLM - Proportion

Proportions can also be analysed with GLM. For this example we can use the dataset `crowder.seeds`, where the variable `germ` is the number of seeds that germinated, while `n` is the total number of seeds:

```
data(crowder.seeds)

str(crowder.seeds)

## 'data.frame':    21 obs. of  5 variables:
##  $ plate  : Factor w/ 21 levels "P1","P10","P11",..: 1 12 15 16 17 18 19
20 21 2 ...
##  $ gen    : Factor w/ 2 levels "O73","O75": 2 2 2 2 2 1 1 1 1 1 ...
##  $ extract: Factor w/ 2 levels "bean","cucumber": 1 1 1 1 1 1 1 1 1 1 ...
##  $ germ   : int  10 23 23 26 17 8 10 8 23 0 ...
##  $ n      : int  39 62 81 51 39 16 30 28 45 4 ...
```

The model is the following:

```
PropMod = glm(cbind(germ, n) ~ gen + extract, data=crowder.seeds,
family="binomial")
```

here we are using the function `cbind` to compute proportions for the number of seeds that germinated in relation to the total number of seeds. The interpretation of the model is the same as above, with `Anova`:

```
Anova(PropMod)

## Analysis of Deviance Table (Type II tests)
##
## Response: cbind(germ, n)
##         LR Chisq Df Pr(>Chisq)
## gen       0.7439  1     0.3884
## extract  18.3500  1  1.838e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

and summary:

```
summary(PropMod)

##
## Call:
## glm(formula = cbind(germ, n) ~ gen + extract, family = "binomial",
##     data = crowder.seeds)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5431  -0.5006  -0.1852   0.3968   1.4796
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -1.0594     0.1326  -7.989 1.37e-15 ***
## genO75            0.1128     0.1311   0.860     0.39
## extractcucumber   0.5232     0.1233   4.242 2.22e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 33.870  on 20  degrees of freedom
## Residual deviance: 14.678  on 18  degrees of freedom
## AIC: 104.65
##
## Number of Fisher Scoring iterations: 4
```

again we can use the function `predict` to compute proportions for particular variables of the predictors.

Conclusions

In this second lecture on inferential statistics we looked at all the most important statistical test we can perform on our data. We started describing ANOVA, which is a particular form of linear modelling specific for factorial designs, and learning how to interpret its results and check if we met all the assumptions. Then we covered linear regression, which is what we normally use when we have either only continuous predictors or a mix between categorical and continuous. Finally, we explored GLM, which are particularly useful in specific dataset where the assumption of normality is violated.

References

Please look at my Blog for additional functions that were not covered in the lecture:

- Linear Modelling - Fabio Veronesi
- GLM - Fabio Veronesi
- Other books can be found in my OneDrive folder under “Inferential Statistics”

Homework

1. From this page, load the Crime dataset and create a regression model that can explain part of the variance in the dependent variable (CrimeRate).
2. From the same page load the Birthweigth dataset and perform a logistic regression as suggested in the data description.