Notes for Ecological Modelling

Tiago A. Marques

2020-11-14

Contents

Chapter 1

Introduction

These notes were written in 2020, during the ecological modelling classes (Modelação Ecológica, ME in short). While it all started as just a way to teach the course, it soon became obvious that with a bit of extra effort put into it these notes might become material that would be useful to others beyond the ME students. This is being written as a bookdown project. Maybe one day it will become a book, for now, these are essentially notes I am using for my course on "Modelação Ecológica" at FCUL.

In particular, I have started thinking about whether this could be a book in two stages, a first part that would appeal to students in Ecologia Numérica, and a second part aimed at students from Modelação Ecológica. The former might be more focused on a traditional approach to statistics, focussed on procedures and statistical tests. TYhe second would be more about the way to think about data and how one might go about to do the best job possible regarding inferences by thinking carefully on how to model the data.

For the current version I am keeping chapters as individual lectures. Once the Ecological Modelling classes are over I might organize them into sensible chapters for a possible book on statistical models for ecological data.

Chapter 2

Preamble

This is a book about Ecological Modelling. Since we are talking about a field that links Models and Ecology, we need first to think about what ecology is, and what models are. And so that is where we begin.

Ecology is the science that studies the relations between all living beings, namely how they influence each other and how they are influenced by the abiotic factors of the surrounding environment. To be able to describe those relationships there is a natural requirement to quantify the nature and strength of these relationships, and hence the need for quantitative methods becomes obvious.

Models are representations of reality. This is a fundamental aspect that one should never forget, they approximate reality, but they are not real. All models are wrong, but some are useful, is perhaps one of the most famous quotes by G.P. Box, and it is a fundamental aspect that I constantly return to, and that I tend to impose upon my students. Because unfortunately, just like artists, statisticians have the bad habit of falling in love with their models. Unlike in the celebrity world, in science there are usual worst consequences than a broken heart or an empty wallet.

Actually, as I think about this book I realize that the "all models are wrong" is in itself wrong. There are in fact many true models, and we will use them extensively in this book. Those are the models that we simulate inside a computer. And those are perhaps some of the best models, since they are both real and extremely useful. By simulating a reality we can control all the aspects of a small world, and hence we can evaluate methods performance, we can illustrate theoretical results and we can show that things do what they say on the tin, or not!

I refer to Ecological Modeling as the art of being able to conceptualize the ecological world as a set of relations, and to be able to translate those relations into equations. Then, to be able to make these equations functions of parameters

that hopefully have a meaning, and then, if lucky, to be able to obtain data that contain information about the relevant parameters of the model. To think about where in the data is information about a given aspect of a system is a crucial step. Perhaps even harder, is to perceive what are the filters that information went through, via a sampling process. And then to be able to introduce observation models to undo those filters and allow reliable inferences to be based on noisy data. Not all models do that, and most models in this book do not separate observation and process model. Those models are arguably the more useful, but you need to understand simpler models before going into those more sophisticated and hence much harder to implement models. By estimating the parameters of the models one is then able to make some informed statements about useful descriptions of the natural world, like relating the weight of a fish with the temperature of the water it lives in or the number of eggs in a nest to the nearest source of food or body of water. Beyond the scope of this book, if one is really really lucky, those models and the results they provide might then be used by others to make management decisions that will ultimately be useful to the ecosystem under study.

Chapter 3

Aknowledgments

A number of colleagues have contributed with comments, upfront anf foremost Susana França and Ana Sofia Reboleira, my partners in Crime at FCUL, co-teachers for Modelação Ecológica and Ecologia Numérica at FCUL, the courses that inspired tis manuscript.

To all my students in Ecologica Numérica and Modelação Ecológica 2020/2021 which questions every time make me realize that what an ecological statistician might think is straightfoward is far from it for a biologist. Without aknowledging that gap, that difference in the way brains are wired, the efforts to convey statistics to biologists are flawed from the beggining.Some students have contributed with direct input on requesting clarifications and typos: those include Diogo Raposo.

To Dinis Pestana, the person that first made me believe that it was possible to make the biology to statistics transition.

To Russel-Alpizar Jara for incintivating me to go to St Andrews, which I see as one of the epicenters of ecologicasl statistics in the 20th century. I was so fortunate to end there and be able to be part of the fantastic family of CREEMinals.

I have been fortunate to have exposed to the minds of a set of amazing scientists and statisticians.

To Steve Buckland, my PhD Supervisor at CREEM that opened to me the world of distance sampling. Steve was the best supervisor I could have asked for. Not only for the support provided, but also for all the opportunites he exposed me to during that period. Teaching in CREEM's Distance Sampling workshops during my PhD years was possibly the source of many of the great things that happenned after. Point in case all the polar bear work that has been the most amazing work-life experience.

To David Borchers, the most amazing statistical brain I have ever had the pleasure to meet and work with. A generous brain that I have often picked up on, always generous with his brilliant contributions. If I could pick a brain for work, I'd pick David's!

To Eric Rexstad, with whom I was fortunate to share an office during his sabbatical at CREEM many years ago. As I was starting a PhD his constant challenges, tips and thoughts made me realize what kind of scientist I would like to become one day. Not there yet, but still trying!

To Len Thomas, an office mate at first, a colleague after, as he claims wasting my time with bad ideas during my Phd (not true!), my boss since I had a PhD and a good friend. If Carlsberg made bosses, Len would be their poster child ;)

Chapter 4

Using R

To work through this book it is fundamental that you know your way around R. I recommend that if you do not know your way around R before hand you take some time to do so before reading the rest of the book, and this first chapter provides some resources to help.

R might seem frightening at first, but even monsters can make something look more pleasant if you look from the right angle. It is all a matter of perspective :) So I will use the help of some monsters here to convince you that this is the right thing to do!

The amazing images in this chapter are all by Allison Horst, Artwork by '@'allison_horst, and I recomend you visit Allison's github repository filled up with amazing stats and maths illustrations (https://github.com/allisonhorst/stats-illustrations), including so many amazing resources to make R look less frightening. To be honest, this chapter is actually also an homage to Allison's work.

And it is not just about stats. If you do not understand how to find the derivative of a function after looking at Artwork by **?** and her amazing visualization series, just give it up, as I suspect you will never will!

Nowadays learning R by example is easy to do, with so many free online resources available to do so.

I recomend that you do it via the RStudio environment, since it provides an integrated environment to integrate wth all R things. And there are many! And if you do so, I can guarantee that in no time you will be having funR.

The advantages of mastering R are priceless, but the learning curve can be daunting at first.

If you want a gentle tutorial into R I have set up such a resource here:

https://github.com/TiagoAMarques/AnIntro2RTutorial

Figure 4.1: Illustrating R Monsters: Artwork by '@'allison_horst

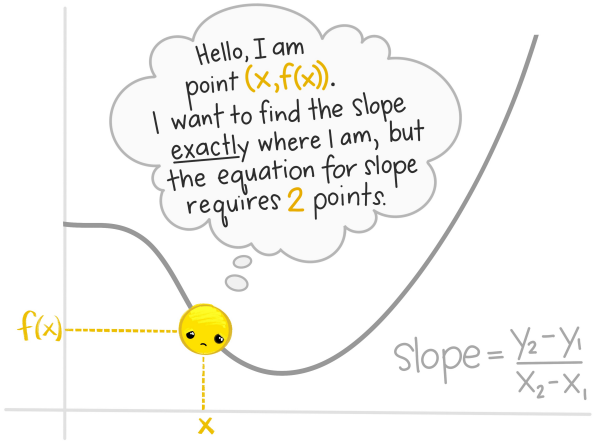


Figure 4.2: Illustrating a Derivative: Artwork by '@'allison_horst

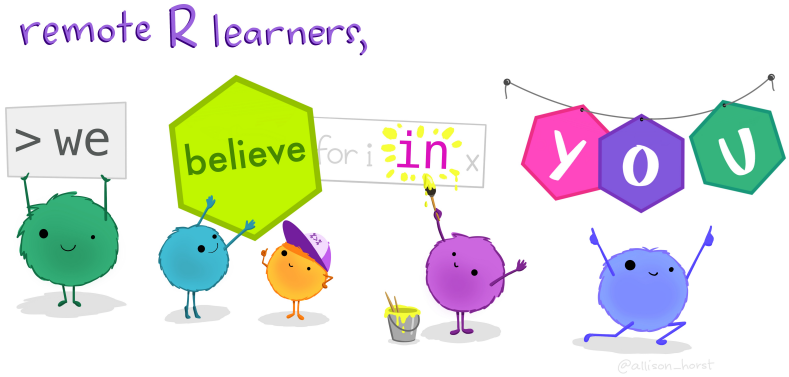


Figure 4.3: Illustrating learning R online: Artwork by '@'allison_horst

Additionally, this book is written in bookdown, and bookdown is the 4.0 version of RMarkdown, a tool that allows you to build dynamic reports based on R code, providing integrated documets that contain all that is required for a given project, from reading the data in to final results and discussion, passing through all the analysis and results. If you want a gentle introduction to RMarkdown using a hands on tutorial based on a versatile template that will do many of the things you'll need to get started, look for no more, there is also one here:

https://github.com/TiagoAMarques/RMarkdownTemplate

Actually, since this book is a bookdown book, you can easily look at it too. It is also on github:

https://github.com/TiagoAMarques/ECOMODbook

Go out and explore, litle grasshoper. You will conquer many great things if you do. You will become a code giant one day. But never forget, you need to be thankful to an entire community, and you are standing on the shoulders of giants!

Figure 4.4: Illustrating having funR: Artwork by '@'allison_horst

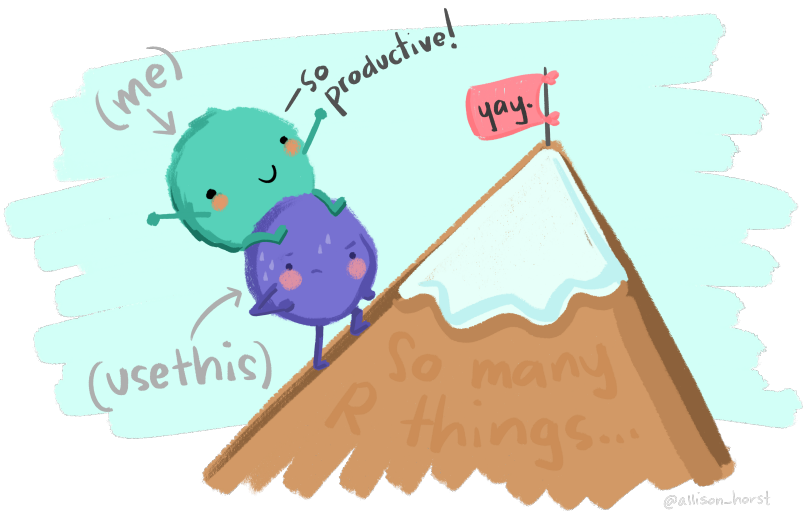


Figure 4.5: Illustrating R's learning curve: Artwork by '@'allison_horst

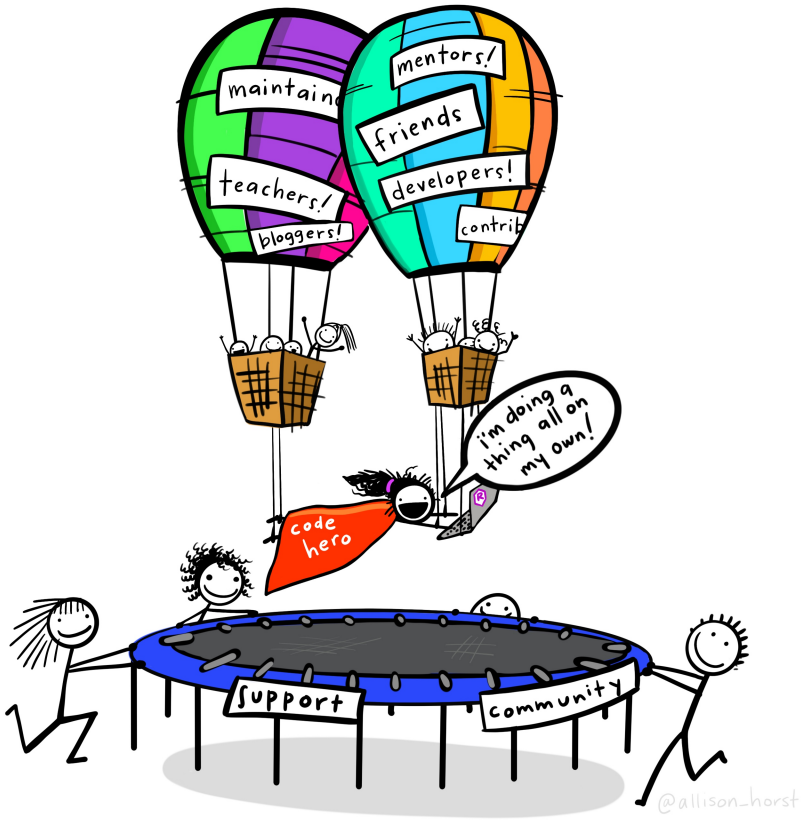


Figure 4.6: Illustrating standing on the shoulders of giants: Artwork by '@'allison_horst

Chapter 5

About regression

Some references woth looking into.

An intro to R: (Zuur et al., 2009a)

Models for ecological data: (Zuur et al., 2007)

More on regression and extending the linear model (just an example): (Faraway, 2006)(Zuur et al., 2009b)

5.1 What is a regression?

Where does the word come from? Gauss and regression towards the mean.

A regression is a model that allows us to predict a response variable y (a.k.a the dependent variable, because it depends on the other variables) as a function of the values ofindependent variables (a.k.a. covariates, predictors or explanatory).

5.2 The general linear model

A general expression for a regression model (i.e. the expression for a generalized linear model is)

$$f[E(Y|X)] = \mu = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k$$

where f is a function - also known as the **link function** - that links the mean value of the response, conditional on the value of the predictors, to the **linear predictor** $\beta_0 + \beta_1 x_1 + ... + \beta_k x_k$ (μ, a linear function of k covariates). In general books tend to represent this as

$$E(Y|X) = f^{-1}(\beta_0 + \beta_1 x_1 + ... + \beta_k x_k)$$

i.e., where what is shown is the inverse of the link function, and sometimes the notation ignores the formal conditioning on the values of the covariates

$$E(Y) = f^{-1}(\beta_0 + \beta_1 x_1 + ... + \beta_k x_k)$$

Because this is a model, for any given observation we have

$$f(y_i|x_i) = \beta_0 + \beta_1 x_{1i} + ... + \beta_k x_{ki} + e_i$$

where the e_i represents the residual (a.k.a. the error).

Most people are used to see the representation when the link function is the identity and hence

$$y_i = \beta_0 + \beta_1 x_{1i} + ... + \beta_k x_{ki} + e_i$$

The simplest form of a generalized linear model is that where there is only one predictor, the link function is the identity and the error is Gaussian (or normal). Note that is the usual simple linear regression model

$$y_i = a + bx_i + e_i$$

with residuals

$$e_i = y_i - (a + bx_i) = y_i - \widehat{y}_i$$

being Gaussian, i.e. e_i~Gau(0,σ), and where the link function is the identity (i.e. $f(E(y)) = 1 \times E(y) = E(y)$).

Chapter 6

Hands On Regression

Here we come up wit a story… an example about regression.

A simple regression is just the situation where we want to model a response variable as a function of a single explanatory variable. As an examples, say, the time a fish takes to react to a predator introduced in an aquarium by getting into shelter, as a function of the water temperature. Let's simulate some data that would represent this scenario, but I am not showing you the way the data was simulated just yet.

Nonetheless, let me tell you that the reaction times were created in object `react`, the temperatures in object `temp`, and these were then packed as a `data.frame` called `reaction`.

The first few lines of the simulated data are shown in Table 6.1.

```
knitr::kable(
  head(reaction, 5), caption = 'The simulated dataset',
  booktabs = TRUE
)
```

Table 6.1: The simulated dataset

react	temp
3.864305	11.84882
6.002570	17.02374
5.400667	15.73326
4.462596	11.68052
6.689445	19.43839

The data is shown in figure 6.1.

```
par(mar = c(4, 4, .1, .1))
plot(react~temp,xlab="Temperature (degrees, Celcius)",ylab="Reaction time (seconds)")
```

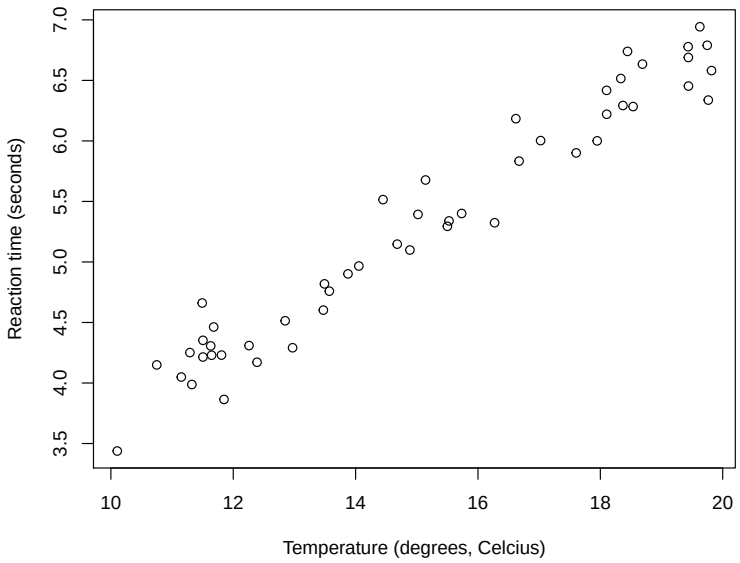


Figure 6.1: An example regression data set that could be eplained by a linear regression

It seems like there is a linear relationship between the predictor (temperature) and the response (the reaction time). We could therepore model it with a simple linear regression. We can do that using R´s function `lm`. We do so here and then look at the summary of the object produced.

The required argument for `lm` is the `formula` that defines the regression model. The symbol `~`is used to represent "as a function of". So here we will want something like "reaction time ~ water temperature".

While this might seem like a detail, it is a good policy to always fit models using explicitly the `data` argument, instead of fitting the model to objects hanging around the workspace. Learn how to be tidy!

Therefore, while the imideate result would be the same, we suggest that you do not do this

```
mod0<-lm(react~temp)
```

nor this

```
mod0<-lm(reaction$react~reaction$temp)
```

but always consider this

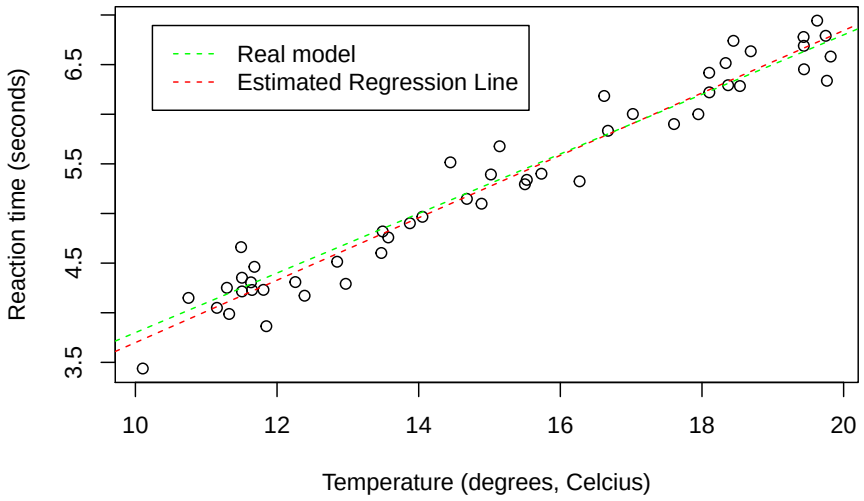
```
mod0<-lm(react~temp,data=reaction)
summary(mod0)
```

```
##
## Call:
## lm(formula = react ~ temp, data = reaction)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.43172 -0.13223 -0.01381  0.13843  0.49265
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.55487    0.15642   3.547 0.000881 ***
## temp         0.31442    0.01015  30.986  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2182 on 48 degrees of freedom
## Multiple R-squared:  0.9524,Adjusted R-squared:  0.9514
## F-statistic: 960.2 on 1 and 48 DF,  p-value: < 2.2e-16
```

This will be easier to read for others, makes you tidy, and will save you headaches when using functions like `predict` over the resulting fitted model.

We can the estimated regression line to the above plot. I color it red to remind us of the fact that this is an estimated line, not the true line that generated the data. While in general we do not know this with real data, here I know the model that was used to simulate the data. Just for comparison I can add it to the plot to compare with the estimated regression line.

```
plot(react~temp,xlab="Temperature (degrees, Celcius)",ylab="Reaction time (seconds)")
abline(mod0,lty=2,col="red")
abline(beta0,beta1,lty=2,col="green")
legend("topleft",legend=c("Real model","Estimated Regression Line"),col=c("green","red"),lty=2, i
```



The estimated line and the true line are very similar, as expected since we have a reasonable sample size, a small error, and a model that is the reality. With real data, this will be the exception, not the rule. All models are wrong, but some are useful. The linear regression model is perhaps one of the simplest, but also one of the most widely used, and hence, one of those that has been extremely useful. But of course, its simplicity is also its major disadvantage, as we shall see.

6.1 The assumptions are on the residuals, not the data

Imagine that you have a single variable that you are interested in modelling. This is the concentration of an enzime in the blood of small rodents, from 4 diffferent species. This is represented in the image below 6.2.

```
hist(ys,breaks=0:40,xlab="Concentration of enzime (mg/L)")
```

A poor (conventional and traditional) biologist would die if shown this dataset, but the truth is there would be no reason for it. If one accounts for the different species, this is what we see. Clear differences between two groups of species.

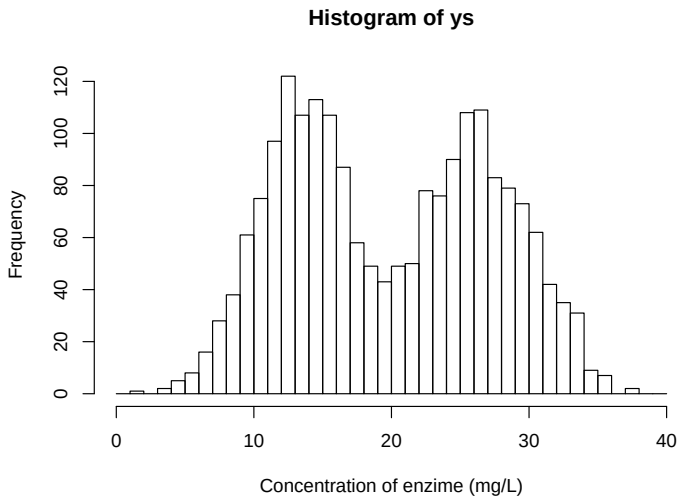
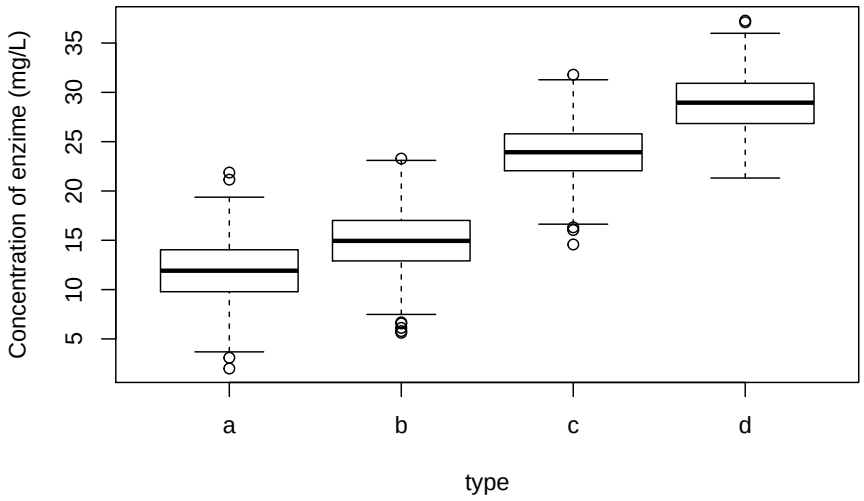


Figure 6.2: Concentration of an enzime (mg/L) in the blood of small rodents, from 4 diffferent species

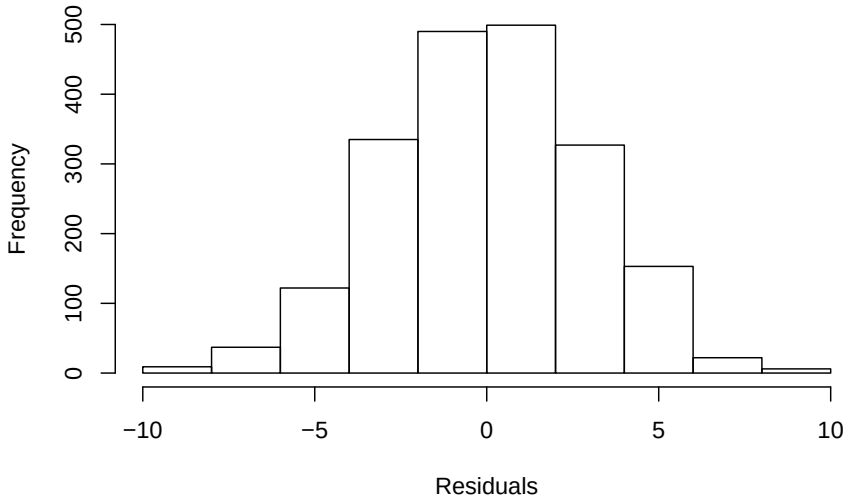
```
boxplot(ys~type,ylab="Concentration of enzime (mg/L)")
```



And further, we can see that the remaining residuals are a beautiful Gaussian.

Not a surprise, since this was simulated data, from a Gaussian model :) !

```
hist(residuals(lm(ys~type)),main="",xlab="Residuals")
```



The take home message from the story: what the data looks like might be irrelevant. The patterns that remain in the residuals, if any, those are the ones we might need to worry about. So do not transform data just becasue the data looks odd. It might just be Gaussian data in disguise!

Chapter 7

Class 6 13 10 2020

7.1 Implementing a regression

We begin by reading the data in "lagartos.txt" and fitting a regression model to it.

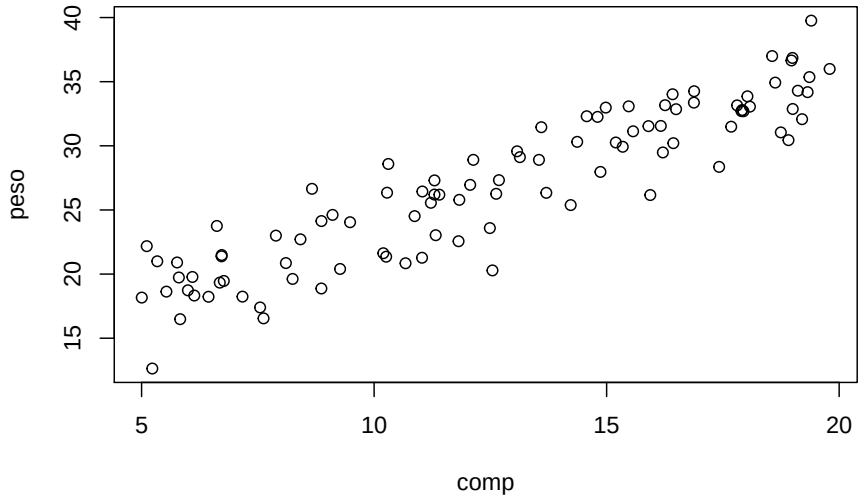
Read the data in

```
#folder<-"../Aula6 13 10 2020/"
folder<-"extfiles/"
lagartos <- read.csv(file=paste0(folder,"lagartos.txt"), sep="")
n <- nrow(lagartos)
```

We see that we have 97 observations.

Plot the data

```
with(lagartos,plot(peso~comp))
```



A linear model seems adequate. Lets fit a regression line to the data

```
lmlag <- lm(peso~comp,data=lagartos)
summary(lmlag)
```

```
##
## Call:
## lm(formula = peso ~ comp, data = lagartos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5199 -1.6961  0.3495  1.7490  4.7127
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.72234    0.72299   16.21   <2e-16 ***
## comp         1.20233    0.05402   22.26   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.415 on 95 degrees of freedom
## Multiple R-squared:  0.8391,Adjusted R-squared:  0.8374
## F-statistic: 495.3 on 1 and 95 DF,  p-value: < 2.2e-16
```

Remember that a linear model is just a special GLM:

```r
glmlag <- glm(peso~comp,data=lagartos,family=gaussian(link="identity"))
summary(glmlag)
```

```
##
## Call:
## glm(formula = peso ~ comp, family = gaussian(link = "identity"),
##     data = lagartos)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.5199  -1.6961   0.3495   1.7490   4.7127
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.72234    0.72299   16.21   <2e-16 ***
## comp         1.20233    0.05402   22.26   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 5.830492)
##
##     Null deviance: 3441.8  on 96  degrees of freedom
## Residual deviance:  553.9  on 95  degrees of freedom
## AIC: 450.27
##
## Number of Fisher Scoring iterations: 2
```

as we can see the output looks a bit different (after all, `lm` and `glm` are different functions!), but the results are exactly the same. This does not prove it, but it illustrates by example that a lm is just a GLM with a Gaussian response and an `identity` link function.

Lets use the results from `lm`, while noting that everything else would be the same.

The estimated regression line is

peso $=11.72 + 1.2 \times$ comp

and the estimated R-squared is 0.84. The standard error associated with the model is estimated to be 2.4146. Below we explain what each of these values correspond to.

The estimated standard error corresponds to the standard deviation of the residuals of the model, that is, the diference between the observations and the predicted values given the model.

The observation we have already, those are the `peso`. We can obtain the predicted `peso` for each observation with the function `predict`, but here we do

it manually so that we see that the errors are just the observations minus the predictions.

```
#get estimated values
estimated<-with(lagartos,summary(lmlag)$coefficients[1]+summary(lmlag)$coefficients[2]*
# note this would be the same as
# estimated<-predict(lmlag)
```

Now we can compute the residuals and their corresponding standard error

```
#get residuals
#erros = observações - valores previstos
# e= y- (a+bx)
# y= (a+bx) + e
resid<-lagartos$peso-estimated
sd(resid)
```

```
## [1] 2.402032
```

Note as predict, we could use just the function `residuals` with the model object as argument to get us the residuals in a single line of code.

The reason the above standard error is not exactly the same as in the model output above has to do with the degrees of freedom, a concepth that is hard to explain in this applied context, but relates to the number of available independent bits of information available. So turst me when I say that we loose a degree of freedom for each parameter estimated in a model. The exact value of the standard deviation as estimated in the model must account for that loss of one extra degree of freedom (associated with estimating the slope of the line), and so the standard formula of the `sd` needs to be adjusted for the lost degree of freedom, like this:

```
#Residual Standard error (Like Standard Deviation)
#the right way
#Subtract one to ignore intercept
k=length(lmlag$coefficients)-1
#get the error sum of squares
SSE=sum(lmlag$residuals**2)
#Residual Standard Error
sqrt(SSE/(n-(1+k)))
```

```
## [1] 2.414641
```

```
#equivalently
sqrt(var(resid)*(n-1)/(n-2))
```

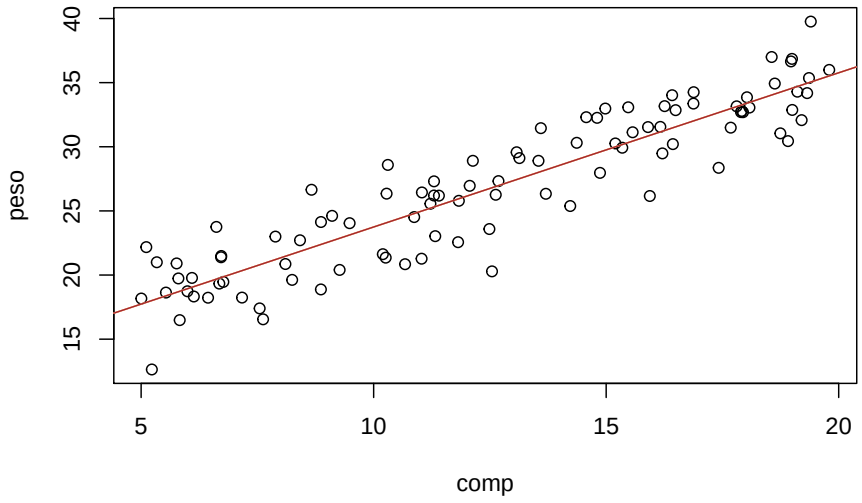
```
## [1] 2.414641
```

Now we get the exact same value as above: 2.4146412.

The `summary` of the model above is very useful, but nothing like adding the

estimated model to the plot with the data. We can easily add the line to the plot with function `abline` (tip: note the ab in `abline` correspond to the a and b in $y = a + bx$, but the function abline is "smart" enough to take an object of class `lm` and extract the corresponding a and b for plotting)

```
#with(lagartos,plot(peso~comp))
plot(peso~comp,data=lagartos)
#these next 3 lines are equivalent
abline(lmlag,col="orange")
abline(a=11.72234,b=1.20233,col="pink")
# y = a + bx
abline(a=summary(lmlag)$coefficients[1,1],b=summary(lmlag)$coefficients[2,1],col="brown")
```



Note the last line works because the parameter estimates are hold in a component of the `summary` of the fitted model called `coefficients`

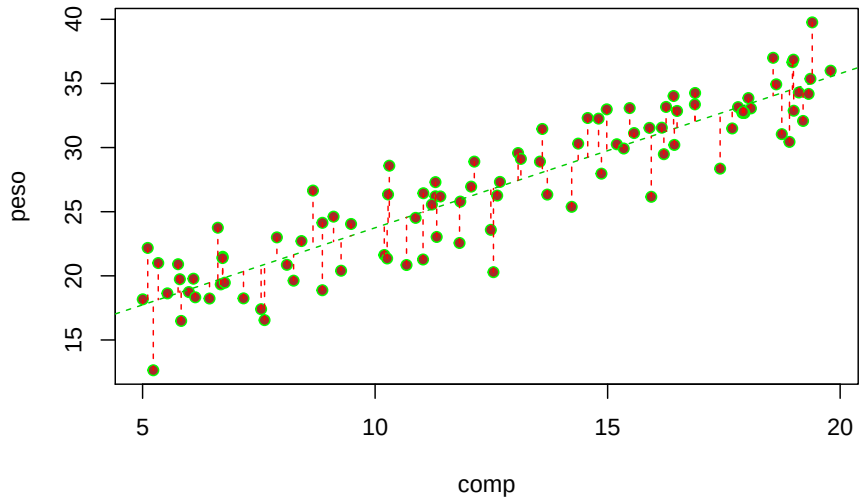
```
summary(lmlag)$coefficients
```

```
##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 11.722343 0.72299153 16.21367 4.135172e-29
## comp         1.202333 0.05402397 22.25555 1.839784e-39
```

Additionally, we can also add the residuals in the plot (we use the very handy function `segments`, that adds segments to plots, to do so)

```
# get estimated/predicted values with function residuals
estimated2<-predict(lmlag)
```

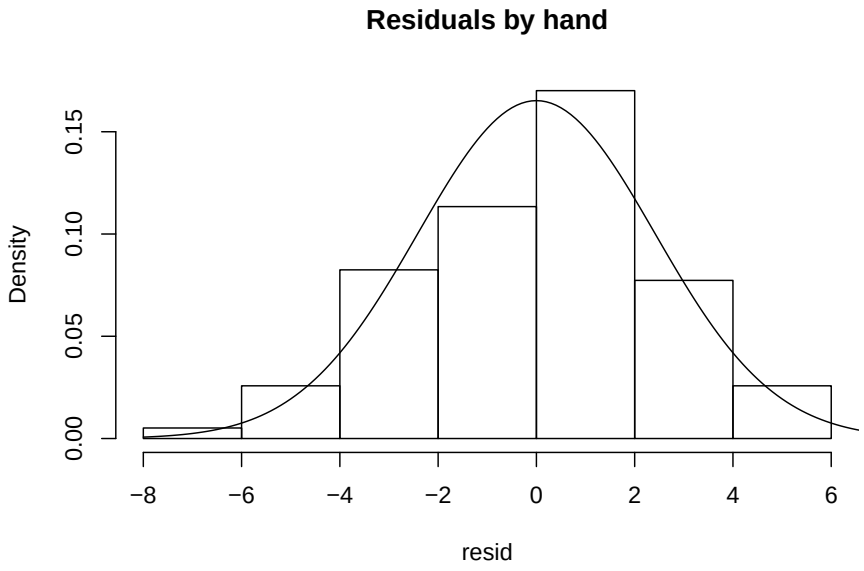
```
#plot the data
with(lagartos,plot(peso~comp,pch=21,bg="brown",col="green"))
abline(lmlag,col=3,lty=2)
#add residuals
with(lagartos,segments(x0 = comp,y0 = peso, x1= comp, y1=estimated,lty=2,col="red"))
```



The regression line is the line that minimizes the sum of the red distances in the plot above. That is also why it is called a minimum squares estimate in the special case of a Gaussian model (in PT, é a reta dos mínimos quadrados).

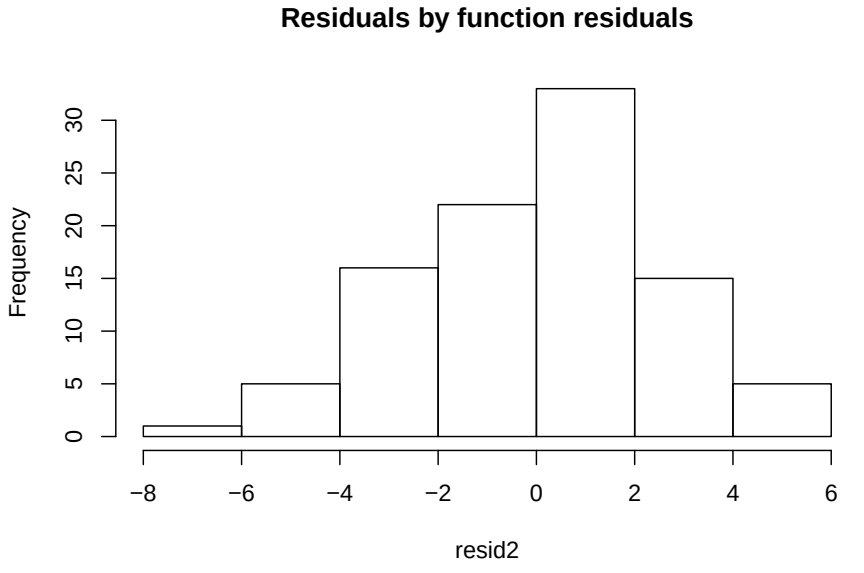
The residuals should, if the model is reasonable - and here that should be the case - be well approximated by a Gaussian distribution. Note we can get the values of the residuals by the difference between the observations and the estimated values, as we did above. We can look at their histogram below

```
hist(resid,main="Residuals by hand",freq=FALSE)
#adding the theorethical density of a Gaussian with mean 0 and the
#correct standard error
lines(seq(-8,8,by=0.1),dnorm(seq(-8,8,by=0.1),mean=0,sd=summary(lmlag)$sigma))
```



or alternatively we can use the bespoke function `residuals`

```
resid2<-residuals(lmlag)
hist(resid2,main="Residuals by function residuals")
```



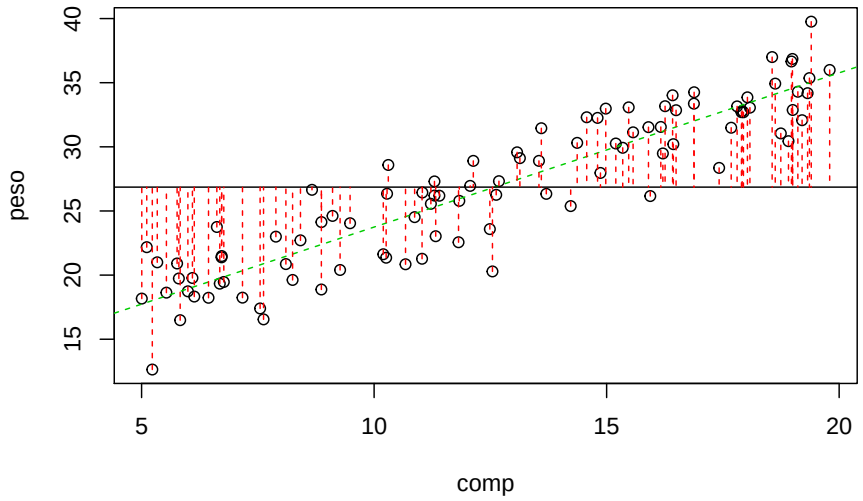
It is often said that the R^2 represents the amount of variation in the response that the regression explains. Why is that?

Because if you assume that all the variability in the response data, the y_i the diference between the data points and a common mean

$$\sum_{i=1}^{n}(y_i - \bar{y})^2$$

in an image, the sum of the square of these quantities

```
#plot
with(lagartos,plot(peso~comp))
abline(lmlag,col=3,lty=2)
abline(h=mean(lagartos$peso))
with(lagartos,segments(x0 = comp,y0 = peso, x1= comp, y1=mean(peso),lty=2,col=2))
```




```
all.var<-sum((lagartos$peso-mean(lagartos$peso))^2)
all.var
```

```
## [1] 3441.795
```

and the variability that is not explained is the one that remains in the errors (the corresponding plot was shown above)

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

```
error.var<-sum((lagartos$peso-estimated)^2)
error.var
```

```
## [1] 553.8968
```

then the ratio of those two quantities is what is NOT explained by the regression model, and therefore, 1 minus that is what explained by the regression model.

```
1-error.var/all.var
```

```
## [1] 0.8390675
```

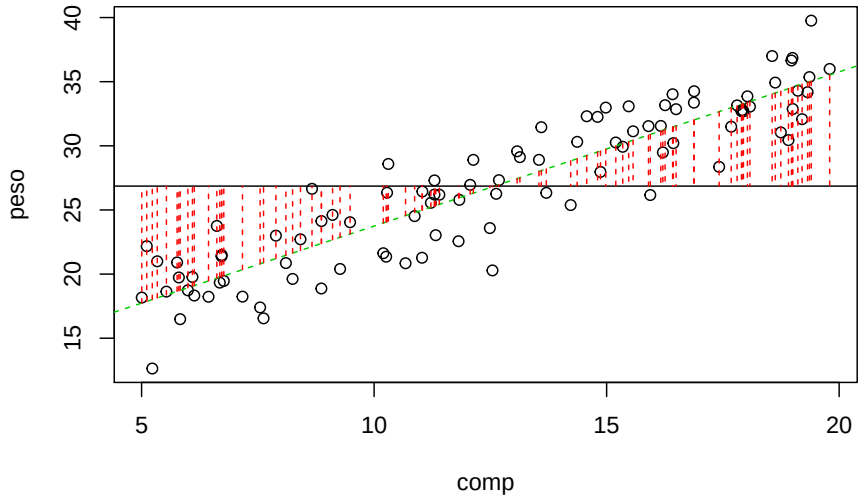
And that... as noted above... is the R^2=0.8391. This comes from the fact that all of the variability in the data (the y, the response, here the `peso`) can be decomposed into the variability explained by the model, and the unexplained variability, that of the errors. In a formula

$$SS_{TOTAL} = SS_{REGRESSO} + SS_{ERRO}$$

Note naturally we could also represent in an image what is explained by the regression model, which is

$$\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2$$

```
#plot
with(lagartos,plot(peso~comp))
abline(lmlag,col=3,lty=2)
abline(h=mean(lagartos$peso))
with(lagartos,segments(x0 = comp,y0 = estimated, x1= comp, y1=mean(peso),lty=2,col=2))
```



and that naturally is obtained as

```
reg.var<-sum((mean(lagartos$peso)-estimated)^2)
reg.var
```

```
## [1] 2887.898
```

and hence the total variability is given by the sum $SS_{REGRESSO} + SS_{ERRO}$

```
reg.var+error.var
```

```
## [1] 3441.795
```

```
all.var
```

```
## [1] 3441.795
```

So, always remember that

$$SS_{TOTAL} = SS_{REGRESSO} + SS_{ERRO}$$

This is also something that comes out often in books without a clear explanation of the reaon why that holds. While here I show it by example, it could be easily demonstrated algebraically if one wanted that

$$\sum_{i=1}^{n}(y_i - \bar{y})^2 = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2 + \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

If you want that, this 28 minute video shows you the proof: https://www.youtube.com/watch?v=aQ32qTjqqJM I think it could take just 5 minutes ;) but many thanks to Dmitry Leiderman for having it out there! He does it in the context of ANOVA, but ANOVA is just a special case of regression, were you have a continous response and a single categorical explanatory variable. Therefore, have fun !

7.2 Simulating regression data

Using the above, simulate data assuming that the TRUE relation between the weight and length of a lizzard was given by

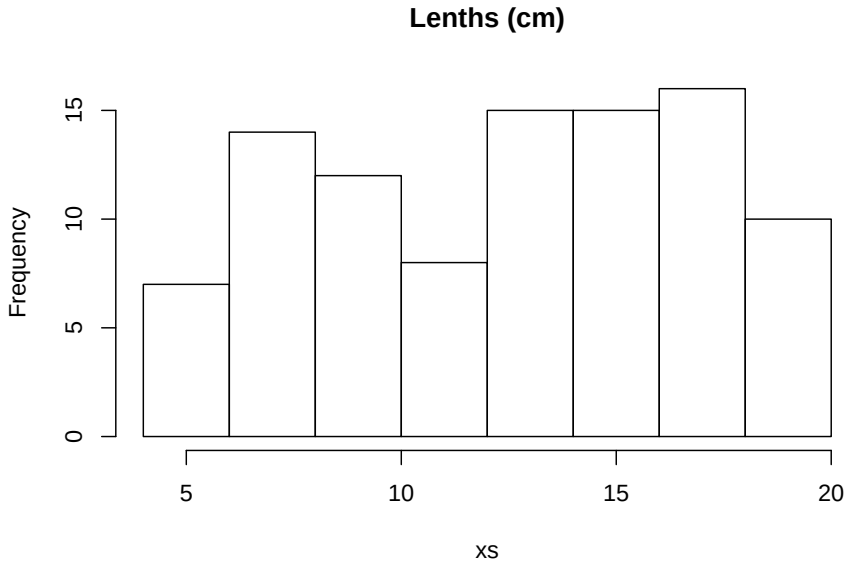
$$peso = 12 + 1.2 * comp$$

We consider that the usual length of a lizard can be between 5 and 20 cm, and the standard error is 4.

As in the data we will have 97 lizards

Then you were told to create the lengths:

```
set.seed(121)
n=97
#lengths
xs=runif(n,5,20)
hist(xs,main="Lenths (cm)")
```

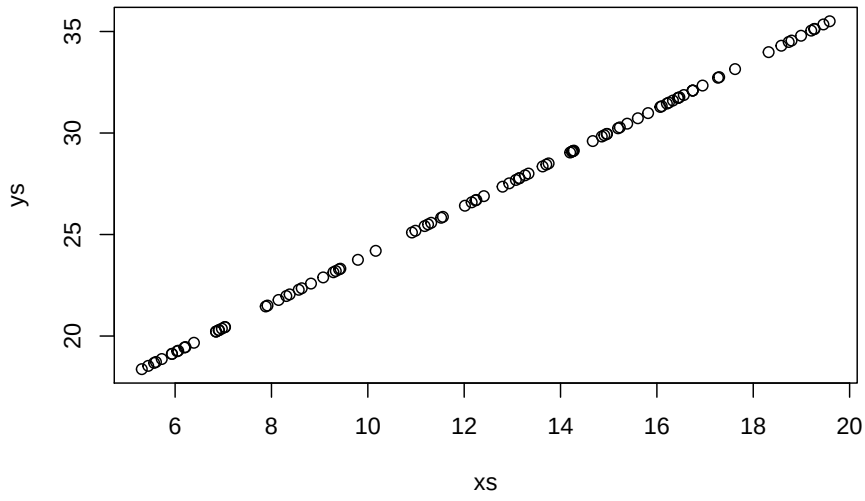


and then to create weights of lizards

```
a=12
b=1.2
ys=a+b*xs
```

If we plot the data, all points are in a single line. Why, because there is no randomness.

```
plot(xs,ys)
```



This means that if you try to run a model, it gives you a warning that the model might be unreliable

```
summary(lm(ys~xs))
```

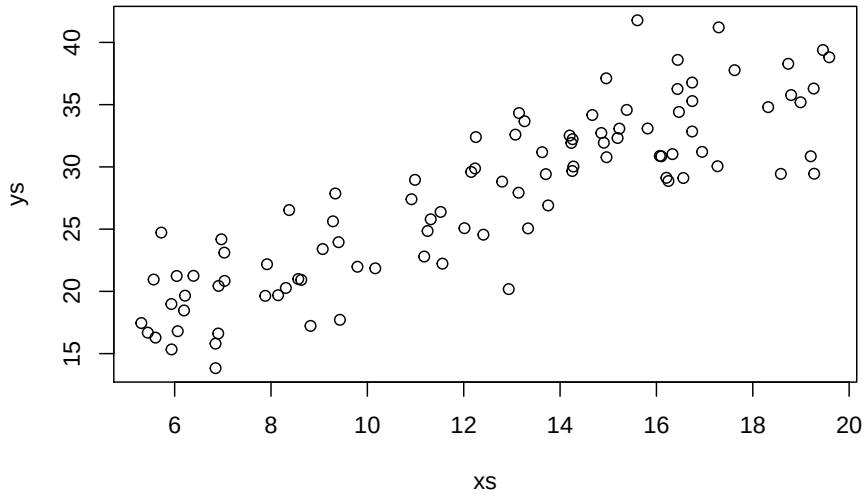
```
## Warning in summary.lm(lm(ys ~ xs)): essentially perfect fit: summary may be
## unreliable
```

```
##
## Call:
## lm(formula = ys ~ xs)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -5.595e-15 -2.460e-15 -1.878e-15 -1.422e-15  1.873e-13
##
## Coefficients:
```

```
##              Estimate Std. Error   t value Pr(>|t|)
## (Intercept) 1.200e+01  6.050e-15 1.983e+15   <2e-16 ***
## xs          1.200e+00  4.611e-16 2.603e+15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.934e-14 on 95 degrees of freedom
## Multiple R-squared:      1,Adjusted R-squared:      1
## F-statistic: 6.773e+30 on 1 and 95 DF,  p-value: < 2.2e-16
```

So… , we add some variance, and plot the data:

```
ys=a+b*xs+rnorm(n,0,4)
plot(xs,ys)
```



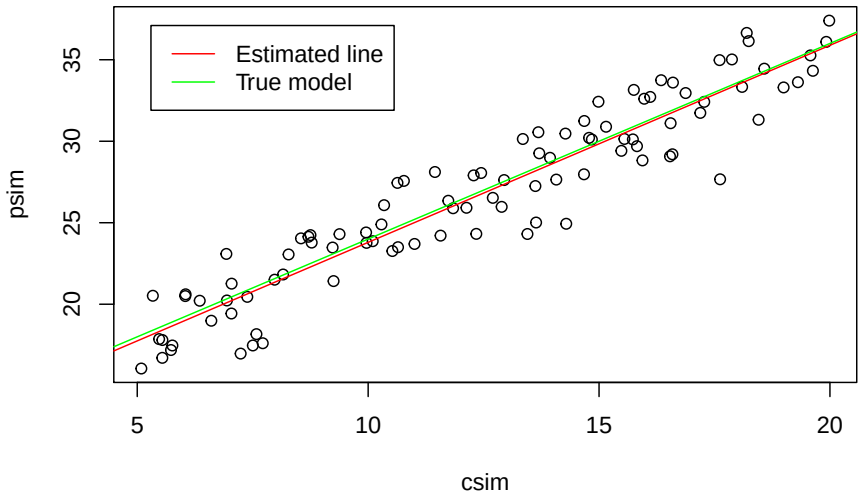
Using the code above, experiment with changing the standard deviation of the error, and see what happens to the estimated R^2, to the parameter estimates, to the estimated error, and to how close the estimated regression mode is to the true model (note this is the amazing advantage of a simulation, which we do not have in real data, we know what reality is, and a true model exists!). This will give you a good feeling for what a regression model is and what is does, and what it can't do. An example of what it can't give you is reliable estimates when the error is large compared to the systematic part of the model.

```
n <- 97
#simular comprimentos
comp.sim <- runif(n,5,20)
```

```
a<-12
b<-1.2
#simular pesos
peso.sim<-a+b*comp.sim+rnorm(n,mean=0,sd=2)
data.sim=data.frame(csim=comp.sim,psim=peso.sim)
plot(psim~csim,data=data.sim)
mod.sim<-lm(psim~csim,data=data.sim)
abline(mod.sim,col="red")
summary(mod.sim)
```

```
##
## Call:
## lm(formula = psim ~ csim, data = data.sim)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.3460 -1.1652  0.1329  1.5072  3.0036
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.70390    0.56783   20.61   <2e-16 ***
## csim         1.20912    0.04325   27.96   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.822 on 95 degrees of freedom
## Multiple R-squared:  0.8916,Adjusted R-squared:  0.8905
## F-statistic: 781.6 on 1 and 95 DF,  p-value: < 2.2e-16
```

```
abline(a,b,col="green")
legend("topleft",legend=c("Estimated line","True model"),col=c("red","green"),lty=1,ins
```

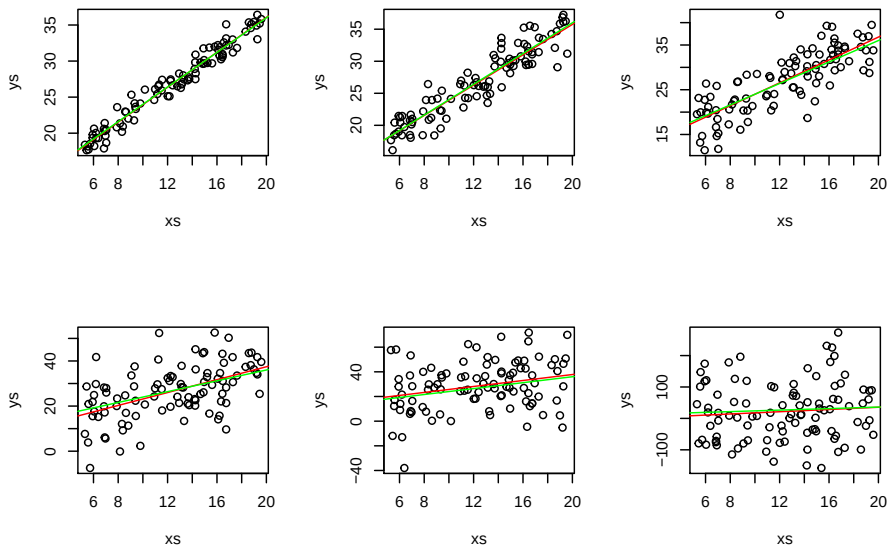


7.2.1 What is the effect of increasing the error: a simulation experiment

Now, let's consider there's more and less variance. We also add to each plot the real line (that with the true parameter values) and the one with the estimated parameter values.

```
par(mfrow=c(2,3))
ys=a+b*xs+rnorm(n,0,1)
plot(xs,ys)
mod1=lm(ys~xs)
abline(mod1,col="red")
abline(a,b,col="green")
ys=a+b*xs+rnorm(n,0,2)
plot(xs,ys)
mod2=lm(ys~xs)
abline(mod2,col="red")
abline(a,b,col="green")
ys=a+b*xs+rnorm(n,0,4)
plot(xs,ys)
mod4=lm(ys~xs)
abline(mod4,col="red")
abline(a,b,col="green")
ys=a+b*xs+rnorm(n,0,10)
plot(xs,ys)
```

```
mod10=lm(ys~xs)
abline(mod10,col="red")
abline(a,b,col="green")
ys=a+b*xs+rnorm(n,0,20)
plot(xs,ys)
mod20=lm(ys~xs)
abline(mod20,col="red")
abline(a,b,col="green")
ys=a+b*xs+rnorm(n,0,100)
plot(xs,ys)
mod100=lm(ys~xs)
abline(mod100,col="red")
abline(a,b,col="green")
```



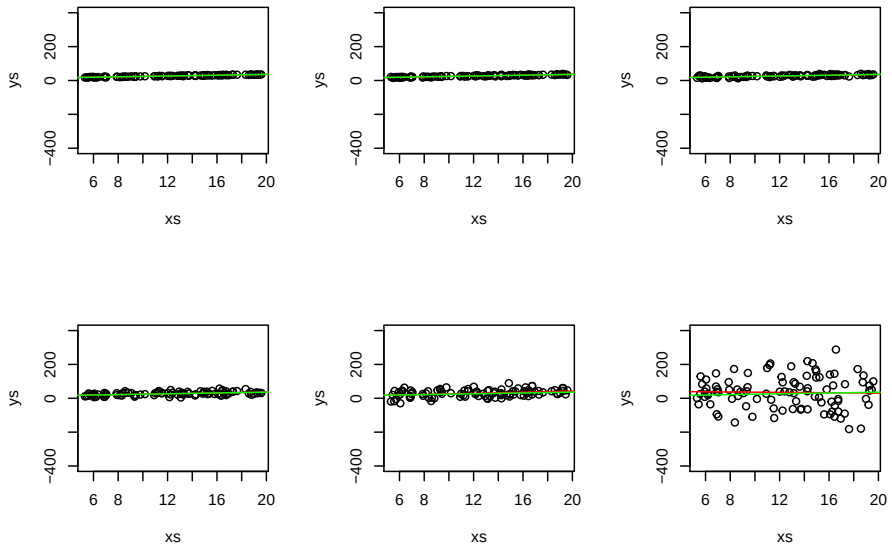
Not surprisingly, as the variance increases, we get data that more and more looks like it is not coming from a real linear process.

You can also look at the model summaries, and there you can see that, in fact, the models become essentially useless as the variance increases! You can see that both from the correlation, but also by the predictions generated from the model (comparing to the truth), and also the significance of the coefficients associated with the regression parameters.

Make no mistake, the reality is always the same, in terms of the fixed part of the model, it's just the variance that increases.

Also, don't get confused, the different green lines might look different, but they are always exactly the same line! You can check that by forcing the y axis to span the same limits.

```
par(mfrow=c(2,3))
ys=a+b*xs+rnorm(n,0,1)
plot(xs,ys,ylim=c(-400,400))
mod1=lm(ys~xs)
abline(mod1,col="red")
abline(a,b,col="green")
ys=a+b*xs+rnorm(n,0,2)
plot(xs,ys,ylim=c(-400,400))
mod2=lm(ys~xs)
abline(mod2,col="red")
abline(a,b,col="green")
ys=a+b*xs+rnorm(n,0,4)
plot(xs,ys,ylim=c(-400,400))
mod4=lm(ys~xs)
abline(mod4,col="red")
abline(a,b,col="green")
ys=a+b*xs+rnorm(n,0,10)
plot(xs,ys,ylim=c(-400,400))
mod10=lm(ys~xs)
abline(mod10,col="red")
abline(a,b,col="green")
ys=a+b*xs+rnorm(n,0,20)
plot(xs,ys,ylim=c(-400,400))
mod20=lm(ys~xs)
abline(mod20,col="red")
abline(a,b,col="green")
ys=a+b*xs+rnorm(n,0,100)
plot(xs,ys,ylim=c(-400,400))
mod100=lm(ys~xs)
abline(mod100,col="red")
abline(a,b,col="green")
```



but since then you loose all the ability to look at the actual data in some of the plots, that is not really that useful!

Below I look at the summary of each model. Look at correlations, at the estimated values for the parameters, their corresponding variances and the R^2.

```
summary(mod1)
```

```
##
## Call:
## lm(formula = ys ~ xs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.87715 -0.62444  0.03329  0.69103  2.47244
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.24930    0.33691   36.36   <2e-16 ***
## xs           1.18596    0.02568   46.19   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.077 on 95 degrees of freedom
## Multiple R-squared:  0.9574,Adjusted R-squared:  0.9569
## F-statistic:  2133 on 1 and 95 DF,  p-value: < 2.2e-16
```

```r
summary(mod2)
```

```
##
## Call:
## lm(formula = ys ~ xs)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7122 -1.5245 -0.0731  1.5830  4.5457
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.22718    0.66606   16.86   <2e-16 ***
## xs           1.26028    0.05076   24.83   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.129 on 95 degrees of freedom
## Multiple R-squared:  0.8665,Adjusted R-squared:  0.8651
## F-statistic: 616.4 on 1 and 95 DF,  p-value: < 2.2e-16
```

```r
summary(mod4)
```

```
##
## Call:
## lm(formula = ys ~ xs)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.8201 -2.6923  0.0809  2.8841 11.1413
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.93903    1.26042   10.27   <2e-16 ***
## xs           1.11757    0.09606   11.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.028 on 95 degrees of freedom
## Multiple R-squared:  0.5876,Adjusted R-squared:  0.5833
## F-statistic: 135.4 on 1 and 95 DF,  p-value: < 2.2e-16
```

```r
summary(mod10)
```

```
##
## Call:
## lm(formula = ys ~ xs)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.236  -6.053  -2.186   7.375  25.314
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  12.1949     3.0136   4.047 0.000106 ***
## xs            1.2102     0.2297   5.269 8.54e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.632 on 95 degrees of freedom
## Multiple R-squared:  0.2262,Adjusted R-squared:  0.218
## F-statistic: 27.76 on 1 and 95 DF,  p-value: 8.541e-07
```

```
summary(mod20)
```

```
##
## Call:
## lm(formula = ys ~ xs)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -48.932 -16.093   1.632  13.610  55.287
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.2412     6.4002   1.444 0.152058
## xs            1.6652     0.4878   3.414 0.000943 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.46 on 95 degrees of freedom
## Multiple R-squared:  0.1093,Adjusted R-squared:  0.09991
## F-statistic: 11.66 on 1 and 95 DF,  p-value: 0.0009426
```

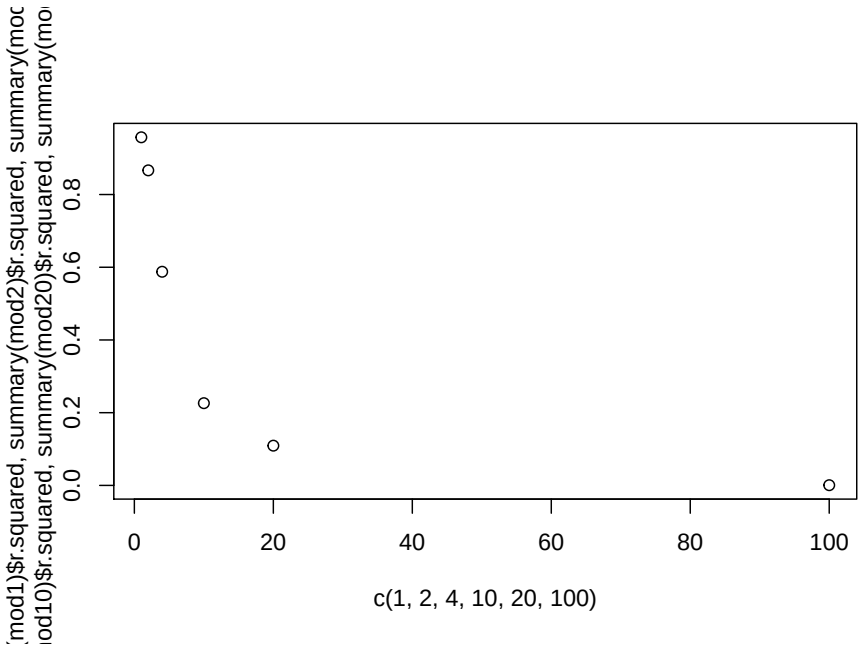
```
summary(mod100)
```

```
##
## Call:
## lm(formula = ys ~ xs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -214.028  -69.887    0.743   60.602  255.109
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  41.6496    30.0996   1.384    0.170
## xs           -0.5455     2.2939  -0.238    0.813
##
## Residual standard error: 96.2 on 95 degrees of freedom
## Multiple R-squared:  0.0005949,Adjusted R-squared:  -0.009925
## F-statistic: 0.05655 on 1 and 95 DF,  p-value: 0.8126
```

As an example, we can plot the R^2 as a function of the variance

```
plot(c(1,2,4,10,20,100),c(summary(mod1)$r.squared,summary(mod2)$r.squared,summary(mod4)$r.squared
```

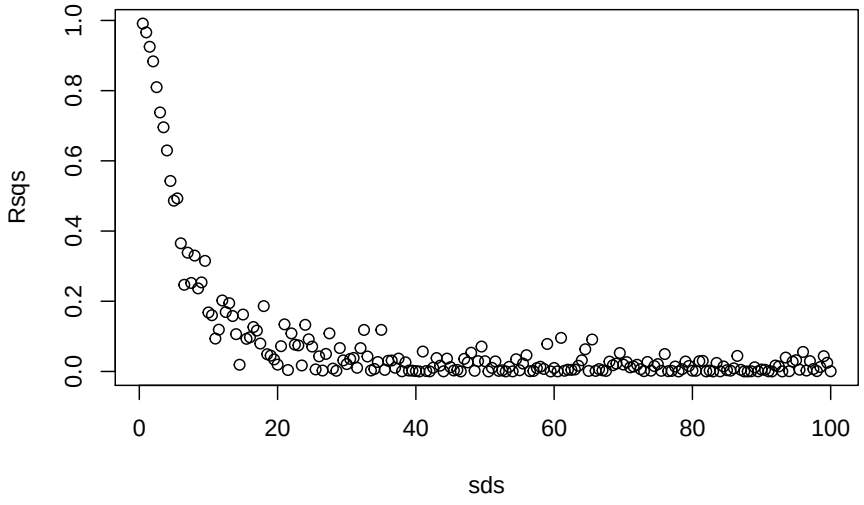


That is quite interesting actually… There seems to be a nonlinear relationship, but we only have a sample size of six (different standard deviatios, i.e., variances, as variance is standard deviation squared), so hard to tell…

Let's show off in R…

```
sds=seq(0.5,100,by=0.5)
nsds=length(sds)
#an object to hold the correlations
Rsqs=numeric(nsds)
for (i in 1:nsds){
  #create data
  ys=a+b*xs+rnorm(n,0,sds[i])
  #estimate model
```

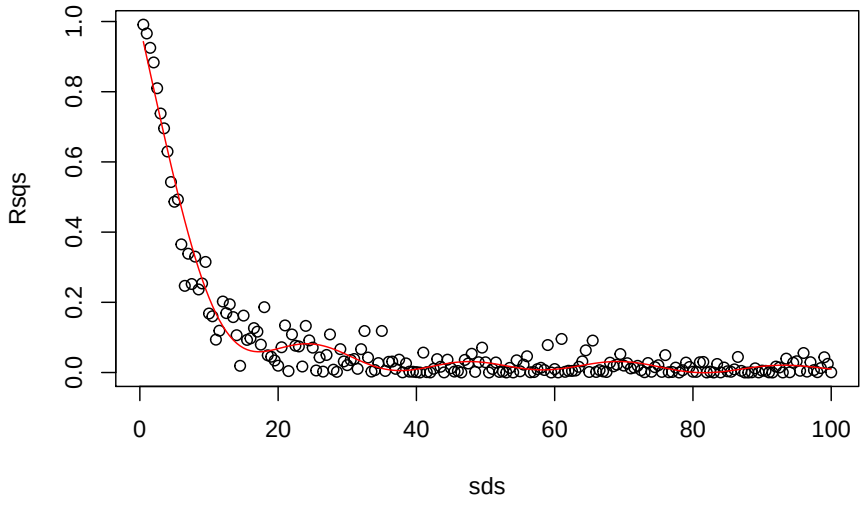
```
  modi=lm(ys~xs)
  #get R-squared
  Rsqs[i]=summary(modi)$r.squared
}
#and at the end... plot results
plot(sds,Rsqs)
```



How cool is that!! Actually, this means we can model the R^2 as a function of the original variance! But we would not be able to model it using a linear model...

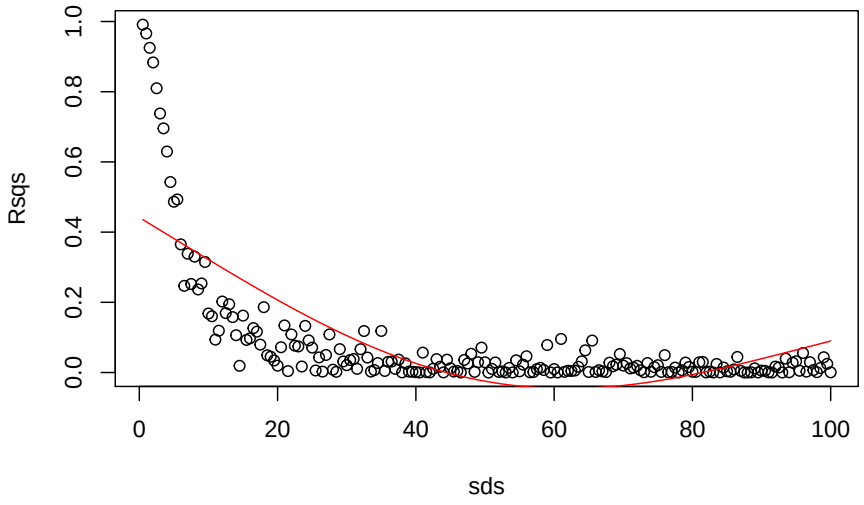
You are not supposed to know about this yet, but I'll continue to show off. Let's use a GAM

```
library(mgcv)
gam1=gam(Rsqs~s(sds),link=log)
#make predictions to plot the estimated GAM model
predRsqs=predict.gam(gam1,newdata = list(sds=sds),type="response")
plot(sds,Rsqs)
lines(sds,predRsqs,col="red")
```



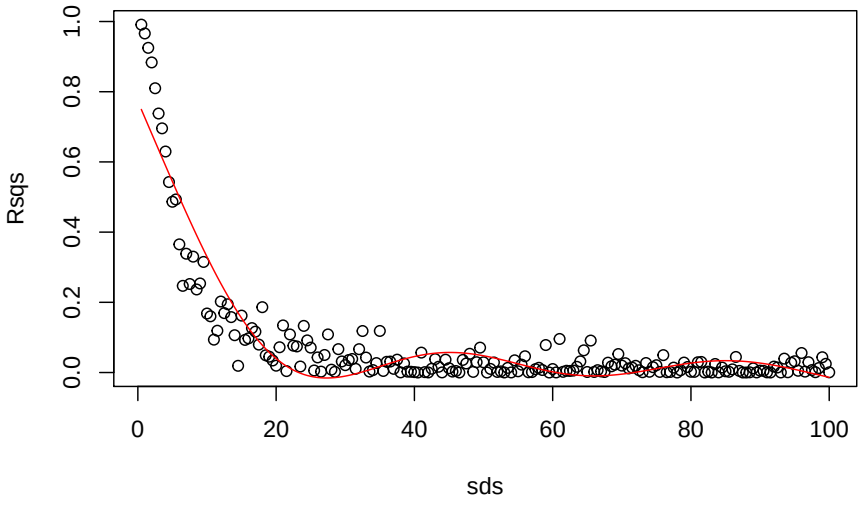
Aha... remember what we talked in class today? It seems like we have overfitted. Then, I constrain the GAM.

```
library(mgcv)
gam1=gam(Rsqs~s(sds,k=3),link=log)
#make predictions to plot the estimated GAM model
predRsqs=predict.gam(gam1,newdata = list(sds=sds),type="response")
plot(sds,Rsqs)
lines(sds,predRsqs,col="red")
```



That was too much...

```
library(mgcv)
gam1=gam(Rsqs~s(sds,k=6),link=log)
#make predictions to plot the estimated GAM model
predRsqs=predict.gam(gam1,newdata = list(sds=sds),type="response")
plot(sds,Rsqs)
lines(sds,predRsqs,col="red")
```



That is already overfitting… conclusion, the GAM is not the right tool here :)

What is…? Well, stay tuned and one day you'll learn!

Chapter 8

Class 7 14 10 2020

In this class we continue exploring regression models, but we are going to increase their complexity. No longer just `a+bx`, but we add more explanatory variables. In particular, we will also add a factor covariate. And then we look under the hood to what that means.

8.1 Task 1

The first task the students were faced was to use some code to explore, by simulation, the impact of having variables in the model that are not relevant to explain the response. In particular, we wanted to identify when there would be no errors, or when there would be type I (a variable not relevant to explain the response is found relevant) and type II (a relevant variable to explain the response is not considered relevant) errors. For the sake of this example we consider a significance level of 5%, but remember there is nothing sacred about it.

The guidelines provided were: "Using the code below, and while changing the `seed` (`******` to begin with, so the code does not run as is!), explore how changing the parameters and the error leads to different amounts of type I and type II errors."

```
# xs1 and xs2 wrong - type II error, xs3 and xs4 ok
seed<-******
set.seed(seed)
#define parameters
n<-50;b0<-5;b1<-3;b2<--2;error <- 4
#simulate potential explanatory variables
xs1=runif(n,10,20)
xs2=runif(n,10,20)
xs3=runif(n,10,20)
```

```
xs4=runif(n,10,20)
#simulate response
ys=b0+b1*xs1-b2*xs2+rnorm(n,sd=error)
#plot data
plot(xs1,ys)
#a model missing a variable, xs2
#summary(lm(ys~xs1))
#the true model
#summary(lm(ys~xs1+xs2))
#a model including irrelevant variables
summary(lm(ys~xs1+xs2+xs3+xs4))
```

The first thing to notice is that the model we simulate from only includes `xs1` and `xs2`. So, `xs3` and `xs4` do not have any impact on the response `y`

So we try different values for the seed and check what happens, Try `seed<-1`

```
seed<-1
set.seed(seed)
#define parameters
n<-50;b0<-5;b1<-3;b2<--2;error <- 4
#simulate potential explanatory variables
xs1=runif(n,10,20)
xs2=runif(n,10,20)
xs3=runif(n,10,20)
xs4=runif(n,10,20)
#simulate response
ys=b0+b1*xs1-b2*xs2+rnorm(n,sd=error)
#plot data
plot(xs1,ys)
#look at model summary
summary(lm(ys~xs1+xs2+xs3+xs4))
```

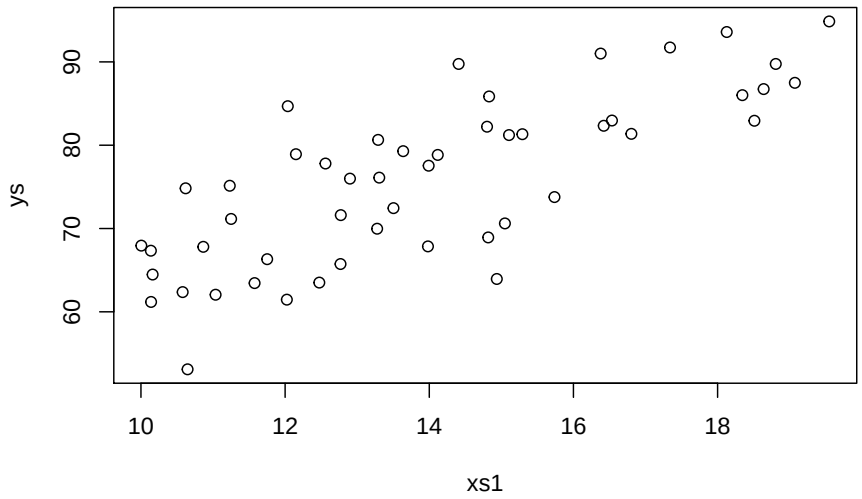
All good, no errors. That is, `xs1` and `xs2` are considered statistically significant at th 5% level and `xs3` and `xs4` are not found relevant to explain the response. Now, we keep changing `seed`

```
seed<-4
set.seed(seed)
#define parameters
n<-50;b0<-5;b1<-3;b2<--2;error <- 4
#simulate potential explanatory variables
xs1=runif(n,10,20)
xs2=runif(n,10,20)
xs3=runif(n,10,20)
xs4=runif(n,10,20)
#simulate response
```

```
ys=b0+b1*xs1-b2*xs2+rnorm(n,sd=error)
#plot data
plot(xs1,ys)
#look at model summary
summary(lm(ys~xs1+xs2+xs3+xs4))
```

We find our first type I error, `xs4` is found statistically significant, but we know it has no effect on the response. The same happens with `seed` being e.g. 9, 10. When we try `seed <- 11` we get another type I error, this time on `xs4`

```
seed<-11
set.seed(seed)
#define parameters
n<-50;b0<-5;b1<-3;b2<--2;error <- 4
#simulate potential explanatory variables
xs1=runif(n,10,20)
xs2=runif(n,10,20)
xs3=runif(n,10,20)
xs4=runif(n,10,20)
#simulate response
ys=b0+b1*xs1-b2*xs2+rnorm(n,sd=error)
#plot data
plot(xs1,ys)
```

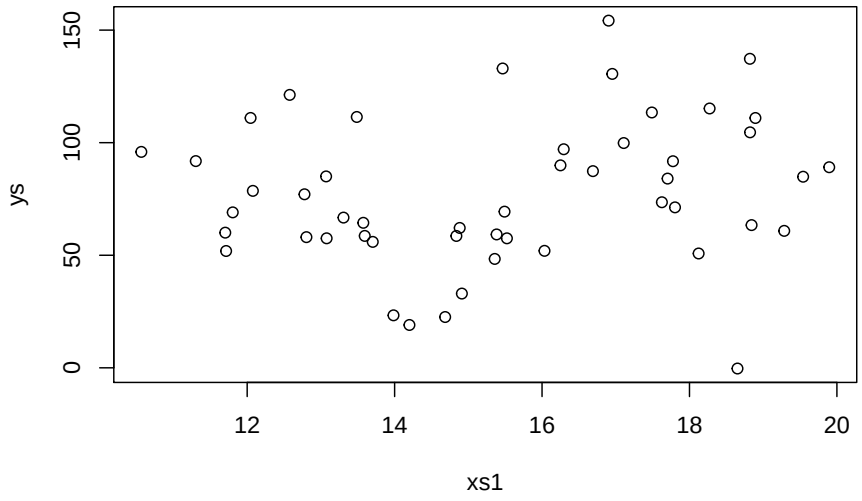



```
#look at model summary
summary(lm(ys~xs1+xs2+xs3+xs4))
```

```
##
## Call:
## lm(formula = ys ~ xs1 + xs2 + xs3 + xs4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.3076  -2.7316   0.3072   2.2102   7.9088
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 16.44303    7.42710   2.214   0.0319 *
## xs1          2.87327    0.21665  13.262  < 2e-16 ***
## xs2          1.93313    0.25274   7.649 1.12e-09 ***
## xs3         -0.47689    0.22325  -2.136   0.0381 *
## xs4         -0.08922    0.20937  -0.426   0.6720
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.074 on 45 degrees of freedom
## Multiple R-squared:  0.8484,Adjusted R-squared:  0.8349
## F-statistic: 62.94 on 4 and 45 DF,  p-value: < 2.2e-16
```

However, even after several runs, we never make a type II error. That must mean this setting has a large power, i.e. the ability to detect a true effect when one exists. Well, there are many ways to decrease power, like having a smaller sample size, increase the error or lower the true effect. Let's try to increase the error, instead of the 4 used above, let's pump it up10 fold to 40

```
seed<-100
set.seed(seed)
#define parameters
n<-50;b0<-5;b1<-3;b2<--2;error <- 40
#simulate potential explanatory variables
xs1=runif(n,10,20)
xs2=runif(n,10,20)
xs3=runif(n,10,20)
xs4=runif(n,10,20)
#simulate response
ys=b0+b1*xs1-b2*xs2+rnorm(n,sd=error)
#plot data
plot(xs1,ys)
```

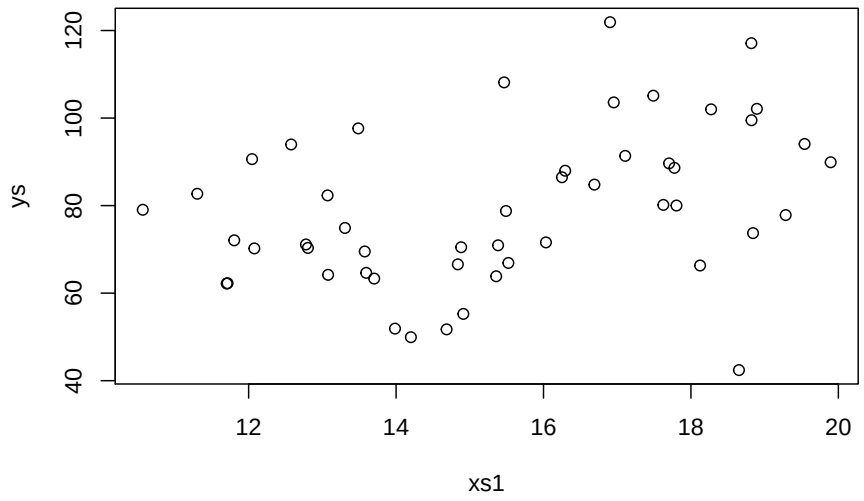



```
#look at model summary
summary(lm(ys~xs1+xs2+xs3+xs4))
```

```
##
## Call:
## lm(formula = ys ~ xs1 + xs2 + xs3 + xs4)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -73.25 -17.86  -6.76  21.25  68.42
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  22.8661    41.3133   0.553    0.583
## xs1           1.6034     1.8564   0.864    0.392
## xs2           0.9163     1.8092   0.506    0.615
## xs3           1.9650     1.6038   1.225    0.227
## xs4          -0.8544     1.7449  -0.490    0.627
##
## Residual standard error: 32.4 on 45 degrees of freedom
## Multiple R-squared:  0.06896,Adjusted R-squared:  -0.0138
## F-statistic: 0.8333 on 4 and 45 DF,  p-value: 0.5112
```

That was an overkill, now there is so much noise must seeds we use do not allow us to find an effect, let's cut that in half to 20

```
seed<-100
set.seed(seed)
#define parameters
n<-50;b0<-5;b1<-3;b2<--2;error <- 20
#simulate potential explanatory variables
xs1=runif(n,10,20)
xs2=runif(n,10,20)
xs3=runif(n,10,20)
xs4=runif(n,10,20)
#simulate response
ys=b0+b1*xs1-b2*xs2+rnorm(n,sd=error)
#plot data
plot(xs1,ys)
```



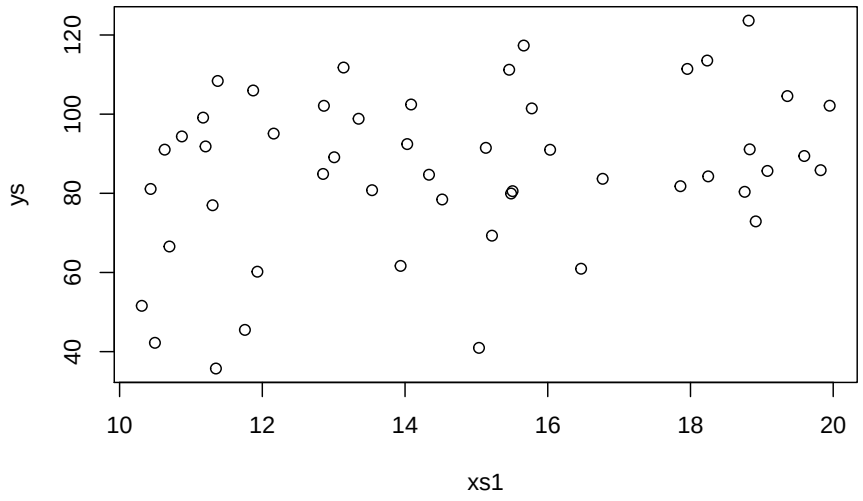

```
#look at model summary
summary(lm(ys~xs1+xs2+xs3+xs4))
```

```
##
## Call:
## lm(formula = ys ~ xs1 + xs2 + xs3 + xs4)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.624  -8.927  -3.380  10.623  34.211
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  13.9330    20.6566   0.675    0.503
## xs1           2.3017     0.9282   2.480    0.017 *
## xs2           1.4582     0.9046   1.612    0.114
## xs3           0.9825     0.8019   1.225    0.227
## xs4          -0.4272     0.8724  -0.490    0.627
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.2 on 45 degrees of freedom
## Multiple R-squared:  0.2262,Adjusted R-squared:  0.1574
## F-statistic: 3.288 on 4 and 45 DF,  p-value: 0.01901
```

back to all correct. Now, let's change seed again

```
seed<-103
set.seed(seed)
#define parameters
n<-50;b0<-5;b1<-3;b2<--2;error <- 20
#simulate potential explanatory variables
xs1=runif(n,10,20)
xs2=runif(n,10,20)
xs3=runif(n,10,20)
xs4=runif(n,10,20)
#simulate response
ys=b0+b1*xs1-b2*xs2+rnorm(n,sd=error)
#plot data
plot(xs1,ys)
```




```
#look at model summary
summary(lm(ys~xs1+xs2+xs3+xs4))
```

```
##
## Call:
## lm(formula = ys ~ xs1 + xs2 + xs3 + xs4)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.079 -10.686  -0.148  16.413  33.845
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.70575   33.83339   0.287   0.7755
## xs1          2.32559    0.93980   2.475   0.0172 *
## xs2          1.97688    1.20464   1.641   0.1078
## xs3          0.69013    1.02110   0.676   0.5026
## xs4          0.08031    0.97427   0.082   0.9347
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.6 on 45 degrees of freedom
## Multiple R-squared:  0.1524,Adjusted R-squared:  0.07706
## F-statistic: 2.023 on 4 and 45 DF,  p-value: 0.1073
```

Bang on, a type II error, as `xs2` is no longer considered statistically significant.

I am sure you can now play with the relevant model parameters, `b1`, `b2`, to incleare and decreese the actual effect, and with sample size `n` or as above with the `error` and explore the consequences of changing the balance in effect size, error and sample size on the ability of incurring in errors when doing regression. But remmeber the key, the reason we are able to see if an error is made or not is because we simulated reality. In this case, as it is never the case in an ecological dataset, we know the true model, which was

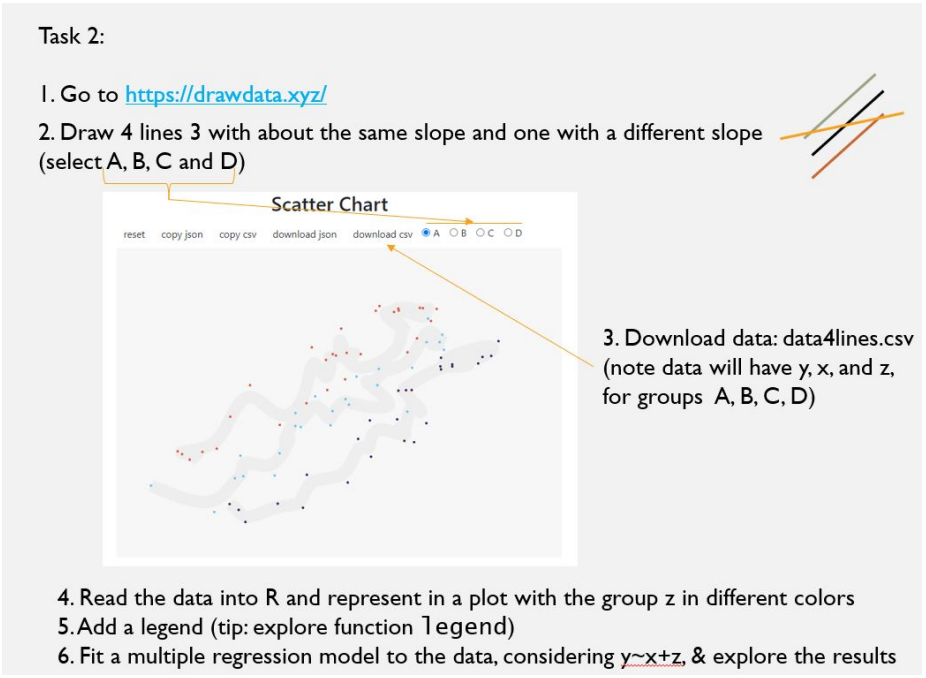
$$y = \beta_0 + \beta_1 xs_1 + \beta_2 xs2$$

That is the luxury of simulation, allowing you to test scenarios where "reality" is known, hence evaluating methods performance.

```
folder<-"extfiles/"
#folder<-"../Aula7 14 10 2020/"
d4l <- read.csv(file=paste0(folder,"data4lines.csv"))
n <- nrow(d4l)
```

8.2 Task 2

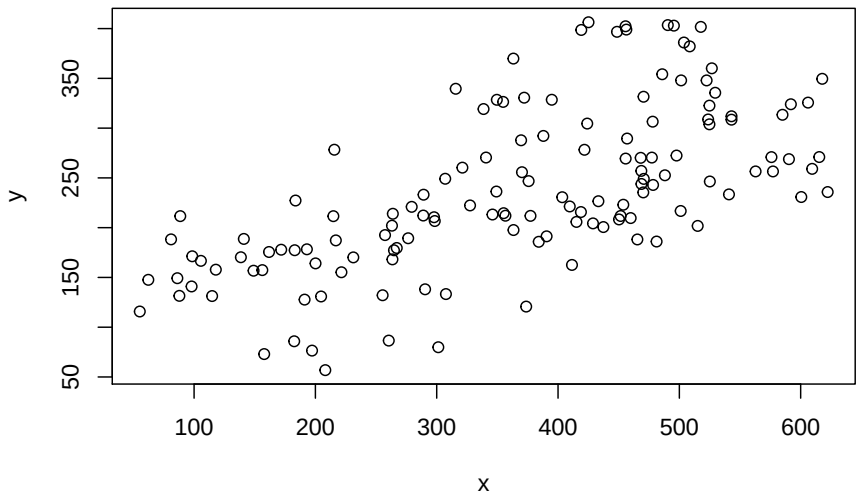
The second task the students were faced was to create some regression data and the explore fitting models to it.



The data was simulated via this website: https://drawdata.xyz/ and was named `data4lines.csv`. Each student had its own dataset, here I work with my example.

We begin by reading the data in and plot it

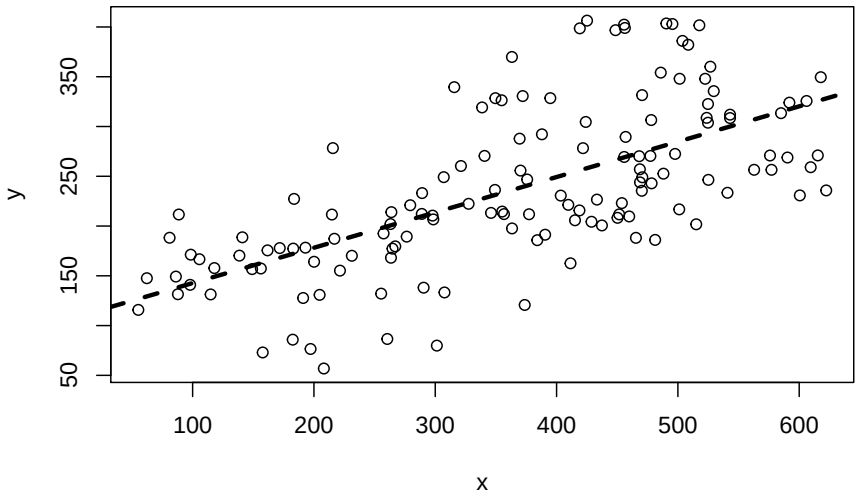
```
#read the data
#folder<-"../Aula7 14 10 2020/"
folder<-"extfiles/"
data4lines <- read.csv(file=paste0(folder,"data4lines.csv"))
#plot all the data
plot(y~x,data=data4lines)
```



Now, to turn this a bit more interesting, we come up with a narrative.

These correspond to observations from weights and lenghts of a sample of animals, fish from the species *Fishus inventadicus*. We could fit a regression line to this data and see if we can predict weight from length

```
#plot all the data
plot(y~x,data=data4lines)
#fit model to pooled data
lmlinesG<-lm(y~x,data=data4lines)
abline(lmlinesG,lwd=3,lty=2)
```




```
summary(lmlinesG)
```

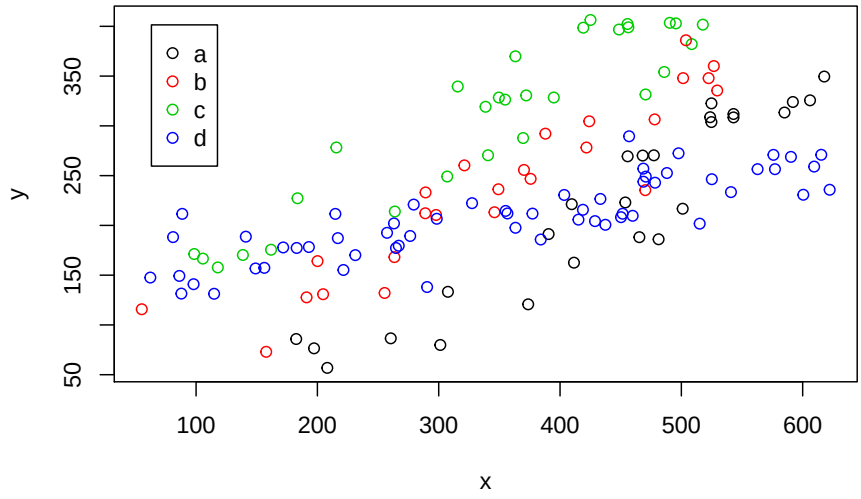
```
##
## Call:
## lm(formula = y ~ x, data = data4lines)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -134.304  -44.188   -1.995   29.432  148.202
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 107.17590   14.03069   7.639 3.51e-12 ***
## x             0.35513    0.03553   9.995  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 61.78 on 136 degrees of freedom
## Multiple R-squared:  0.4235,Adjusted R-squared:  0.4193
## F-statistic: 99.91 on 1 and 136 DF,  p-value: < 2.2e-16
```

and it looks like we can indeed predict the weight of the species from its length. The length is higly statistically significant. Not surprisingly, the longer the fish the heavier it is.

Now, the plot thickens. These animals actually came from 4 different museums,

and are assumed to be the same species. However, a scientis decides to look at wether there are differences in the data from the 4 museums. So he colours the data by museum

```
#plot all the data
plot(y~x,col=as.numeric(as.factor(z)),data=data4lines,pch=1)
legend("topleft",inset=0.05,legend=letters[1:4],col=1:4,pch=1)
```



We see a patter in the data, the data from the different museums tend to cluster. He decides to investigate. Note folks providing names to museum in this country are a bit borring, and the museums are called "a", "b", "c" and "d".

Our smart researcher says: "well, it seems like the relationship might be different in each museum". Then, maybe I should fit a model that includes museum as a covariate `weight~length+museum`.

$$y = \beta_0 + \beta_1 \times length + \beta_2 \times museum$$

And so he does and plots it

```
#fit model per group
lmlines<-lm(y~x+z,data=data4lines)
summary(lmlines)
```

```
##
## Call:
## lm(formula = y ~ x + z, data = data4lines)
```

```
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -90.01 -35.01   2.54  35.51 108.10
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  45.51813   13.83069   3.291  0.00128 **
## x             0.39657    0.02516  15.763  < 2e-16 ***
## zb           54.92376   12.11597   4.533 1.28e-05 ***
## zc          128.20339   11.72572  10.934  < 2e-16 ***
## zd           22.82412   10.26509   2.223  0.02787 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 42.5 on 133 degrees of freedom
## Multiple R-squared:  0.7332,Adjusted R-squared:  0.7252
## F-statistic: 91.38 on 4 and 133 DF,  p-value: < 2.2e-16
```

The output shows that the length is relevant, but the museum is relevant too. The relationship might be different per museum! In the output we see the `x`, the length, but not the `z`, it has been transformed into `zb`, `zc` and `zd`. Why is that? That is a mistery that we shall unfold now!

While the model we are fitting might be represented by `weight~length+museum`, th edesign matrix being fitted replaces the `museum` (a factor with 4 levels) with 3 dummy variables (a factor with k levels required k-1 dummy variables). So the real model being fitted is really

$$y = \beta_0 + \beta_1 \times length + \beta_{2b} \times zb + \beta_{2c} \times zc + \beta_{2d} \times zd$$

Wait, but where is the level `a`? It is in the intercept, and if I had an euro for each time that confused a student, I would not be here but in a beach in the Bahamas having a piña colada :)

But let's unfold the mistery, shall we? By default, R takes 1 level of (each/a) factor and uses it as the intercept. Here it used `a` (the choice is in this case by alphabetical ored, but one can change that, which might be useful if e.g. you want to have as the intercept a control level, say; look e.g. into function `factor` help to see how you can change the baseline `level` of a factor).

Hence, the intercept for museum `a` is 45.5181335. What about the intercep of the othe rmuseums? They are always reported with `a` as the reference. Look at the equation above, what happens when say zc is 1 and zd and zb are 0, it becomes

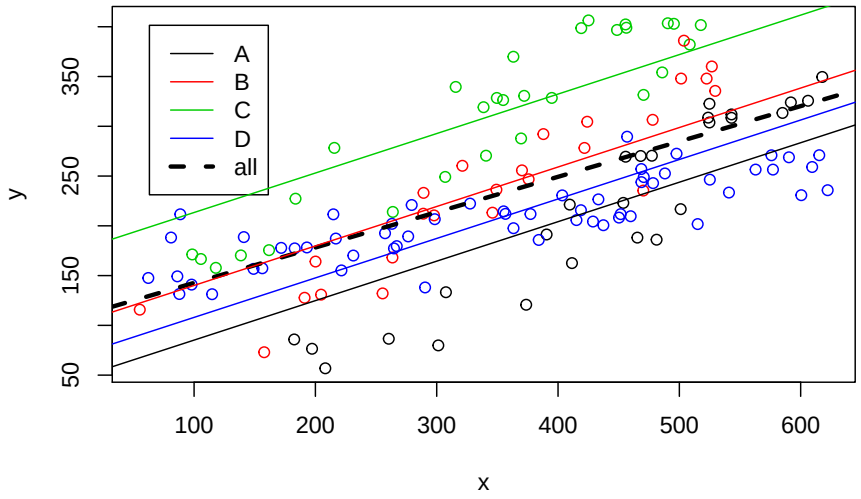
$$y = \beta_0 + \beta_1 \times length + \beta_{2c} \times zc$$

$$y = (\beta_0 + \beta_{2c}) + \beta_1 \times length = intercep + slope \times length$$

and so, from the above output, that equates to `y=lmlines$coefficients[1]+lmlines$coefficients[4]` or 173.7215247+0.3965715 × `length`.

So now we can add all these estimated regression lines to the plot

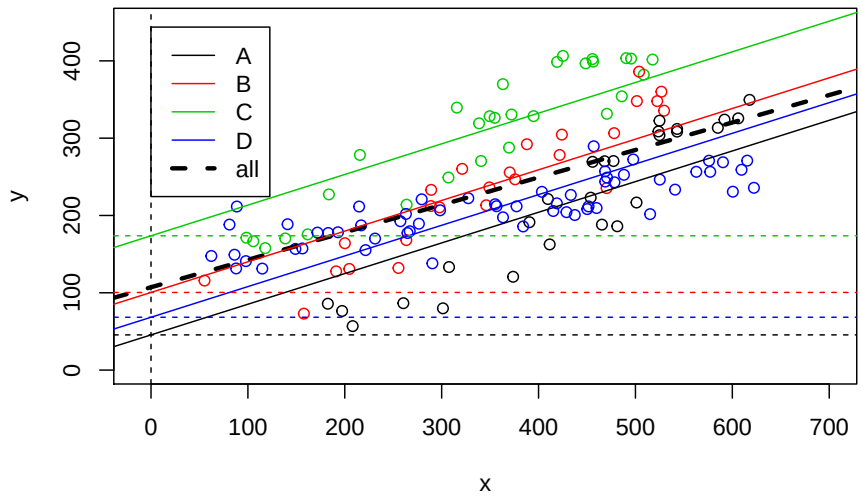
```
#plot all
plot(y~x,col=z,data=data4lines)
legend("topleft",inset=0.05,legend=c(LETTERS[1:4],"all"),col=c(1:4,1),lty=c(rep(1,4),2)
#these are the wrong lines... why?
abline(lmlinesG,lwd=3,lty=2)
abline(lmlines$coefficients[1],lmlines$coefficients[2],col=1)
abline(lmlines$coefficients[1]+lmlines$coefficients[3],lmlines$coefficients[2],col=2)
abline(lmlines$coefficients[1]+lmlines$coefficients[4],lmlines$coefficients[2],col=3)
abline(lmlines$coefficients[1]+lmlines$coefficients[5],lmlines$coefficients[2],col=4)
```



note that, not surprisingly, all these lines have the same slope. Or in other words, the model we considered assumes that the slope of the model is the same across museums (which, remember, we know if not true!). We can easily check that the intercepts (i.e. where the lines cross when length=x=0) of all lines are indeed easy to get from the model's output

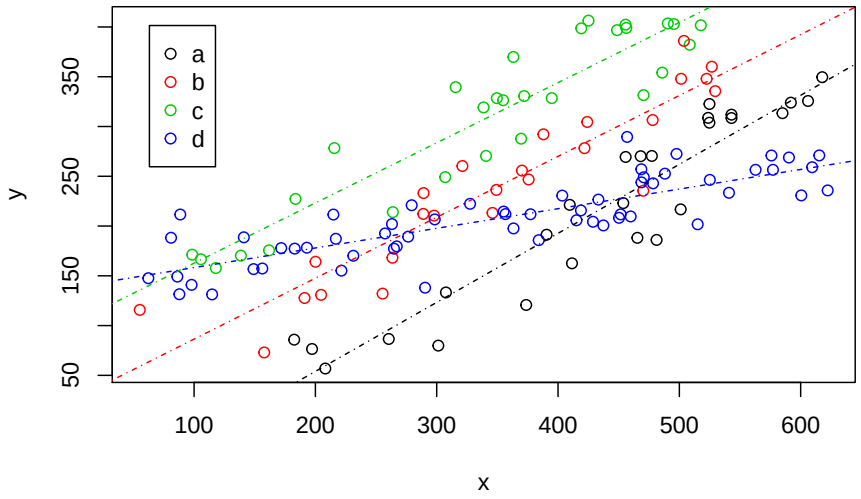
```
#plot all
plot(y~x,xlim=c(-10,700),ylim=c(0,450),col=z,data=data4lines)
legend("topleft",inset=0.05,legend=c(LETTERS[1:4],"all"),col=c(1:4,1),lty=c(rep(1,4),2)
#these are the wrong lines... why?
```

```
abline(lmlinesG,lwd=3,lty=2)
abline(lmlines$coefficients[1],lmlines$coefficients[2],col=1)
abline(lmlines$coefficients[1]+lmlines$coefficients[3],lmlines$coefficients[2],col=2)
abline(lmlines$coefficients[1]+lmlines$coefficients[4],lmlines$coefficients[2],col=3)
abline(lmlines$coefficients[1]+lmlines$coefficients[5],lmlines$coefficients[2],col=4)


abline(v=0,lty=2)
abline(h=45.51813,lty=2,col=1)
abline(h=45.51813+54.92376,lty=2,col=2)
abline(h=45.51813+128.20339,lty=2,col=3)
abline(h=45.51813+22.82412,lty=2,col=4)
```



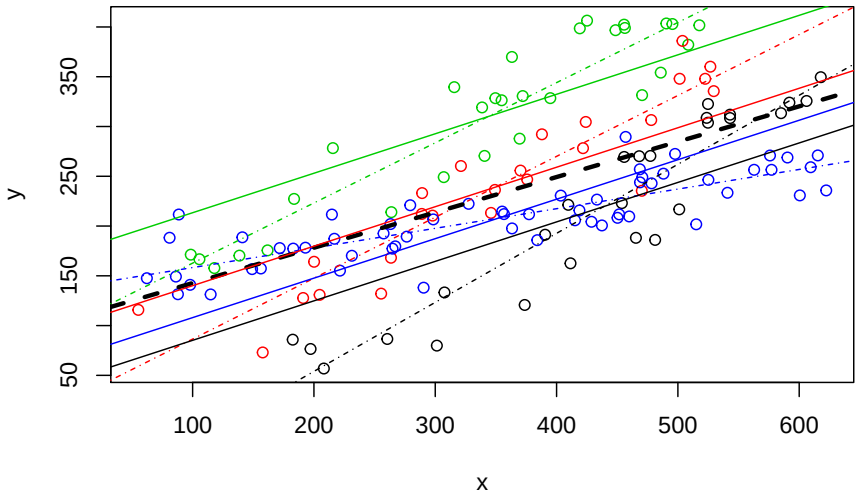
Now, the smart biologist then says that he could also fit a separate line to each museum's data. And so he does, and that looks like this:

```
#plot all the data
plot(y~x,col=as.numeric(as.factor(z)),data=data4lines,pch=1)
#completely independet regression lines
abline(lm(y~x,data=data4lines[data4lines$z=="a",]),col=1,lty=4)
abline(lm(y~x,data=data4lines[data4lines$z=="b",]),col=2,lty=4)
abline(lm(y~x,data=data4lines[data4lines$z=="c",]),col=3,lty=4)
abline(lm(y~x,data=data4lines[data4lines$z=="d",]),col=4,lty=4)
legend("topleft",inset=0.05,legend=letters[1:4],col=1:4,pch=1)
```



Naturally, now the lines do not have the same slope, and we can compare all these in a single plot. This plot is really messy, as it includes the pooled regression (the thick black line), the regressions fitted to independent data sets, one for each museum (the solid lines), and the regressions resulting from the model with museum as a factor covariate (dotted-dashed lines).

```
#plot all the data
plot(y~x,col=as.numeric(as.factor(z)),data=data4lines,pch=1)
#completely independet regression lines
abline(lm(y~x,data=data4lines[data4lines$z=="a",]),col=1,lty=4)
abline(lm(y~x,data=data4lines[data4lines$z=="b",]),col=2,lty=4)
abline(lm(y~x,data=data4lines[data4lines$z=="c",]),col=3,lty=4)
abline(lm(y~x,data=data4lines[data4lines$z=="d",]),col=4,lty=4)
#these are the wrong lines... why?
abline(lmlinesG,lwd=3,lty=2)
abline(lmlines$coefficients[1],lmlines$coefficients[2],col=1)
abline(lmlines$coefficients[1]+lmlines$coefficients[3],lmlines$coefficients[2],col=2)
abline(lmlines$coefficients[1]+lmlines$coefficients[4],lmlines$coefficients[2],col=3)
abline(lmlines$coefficients[1]+lmlines$coefficients[5],lmlines$coefficients[2],col=4)
```



But what is the best model to describe the data? That is a mistery that will remain to unfold. For that we will need and additional complication in a regression model: interactions.

But note 1 thing to begin with. The pooled model uses just 2 parameters, one slope and one intercept. The independent lines use 8 parameters, 4 slopes and 4 intercepts, one line for each museum. And the single model with `lenght` and `museum` uses 5 parameters, the intercept, the slope for `length`, and 3 parameters associated with the $k-1=3$ levels of `museum` (remember, one level of each factor is absorbed by the regression intercept).

So the choice of what is best might be not straightforward. While we created the data by hand, we do not know the true model! Choosing the best model requires choosing between models with different complexity, i.e. different number of parameters. We will need a parsimonious model, one that describes the data well, but with a number of parameters that is not too high for the available data. That will also require selection criteria.

Stay tuned for the next episodes on our regression saga!

Chapter 9

Class 8 20 10 2020 - t-test and ANOVA are just linear models

The objective of this chapter is to explore different regression models and to see how they relate to statistical procedures one might not associate with a regression, when in fact, they are just special cases of a regression.

9.1 The t-test

While we did not do the t-test in class, this is useful because it allows you to see how a simple t-test is just a linear model too, and acts as a building block for the next examples. The t-test allows us to test the null hypothesis that two samples have the same mean.

Create some data

```
#Making up a t-test
#making sure everyone gets the same results
set.seed(980)
```

Then we define the sample size and the number of treatments

```
#define sample size
n=100
#define treatments
tr=c("a","b")
#how many treatments - 2 for a t test
ntr=length(tr)
```

```
#balanced design
n.by.tr=n/ntr
```

Now, we can simulate some data. First, the treatments

```
type=as.factor(rep(tr,each=n.by.tr))
cores=rep(1:ntr,each=n.by.tr)
```

Then we define the means by treatment - note that they are different, so the null hypothesis in the t-test, that the mean of a is equal to the mean of b, is known to be false in this case.

```
#define 4 means
ms=c(3,4)
```

Then, the key part, the response variable, with a different mean by treatment. Note the use of the `ifelse` function, which evaluates its first argument and then assigns the value of its second argument if the first is true or the value of the second if its first argument is false. An example

```
ifelse(3>4,55,77)
```

```
## [1] 77
```

```
ifelse(3<4,55,77)
```

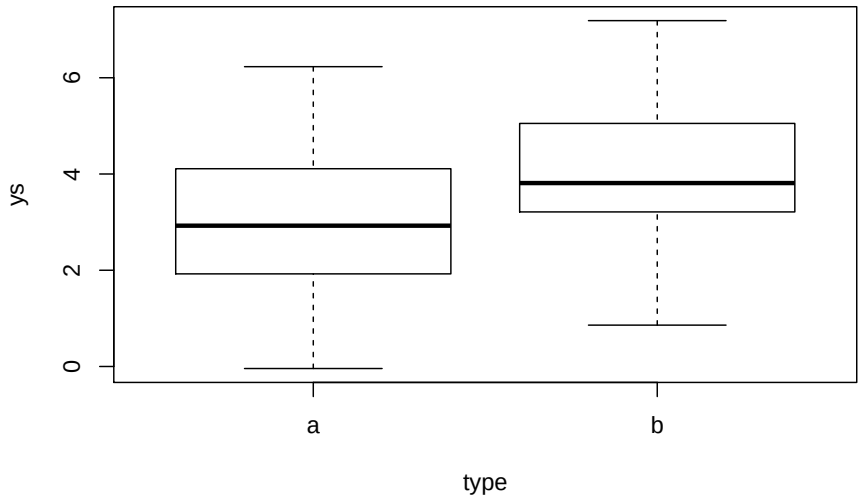
```
## [1] 55
```

So now, generate the response data

```
ys=ifelse(type=="a",ms[1],ms[2])+rnorm(n,0,1.5)
```

Look at the data

```
plot(ys~type)
```



Now, we can run the usual t-test

```
t.test(ys~type)
```

```
##
##  Welch Two Sample t-test
##
## data:  ys by type
## t = -2.8043, df = 97.475, p-value = 0.006087
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.4263293 -0.2441277
## sample estimates:
## mean in group a mean in group b
##        3.106656        3.941884
```

and now we can do it the linear regression way

```
lm0=lm(ys~type)
summary(lm0)
```

```
##
## Call:
## lm(formula = ys ~ type)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -3.1489 -0.9131 -0.1315  1.0295  3.2450
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.1067     0.2106  14.751  < 2e-16 ***
## typeb         0.8352     0.2978   2.804  0.00608 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.489 on 98 degrees of freedom
## Multiple R-squared:  0.07428,Adjusted R-squared:  0.06484
## F-statistic: 7.864 on 1 and 98 DF,  p-value: 0.006081
```

and as you can see, we get the same result for the test statistic. It is the same thing! And we can naturally get the estimated means per group. The mean for a is just the intercept of the model. To get the mean of the group b we add the mean of group b to the intercept, as

```
#mean of ys under treatment a
summary(lm0)$coefficients[1]
```

```
## [1] 3.106656
```

```
#mean of ys under treatment b
summary(lm0)$coefficients[1]+lm0$coefficients[2]
```

```
##    typeb
## 3.941884
```

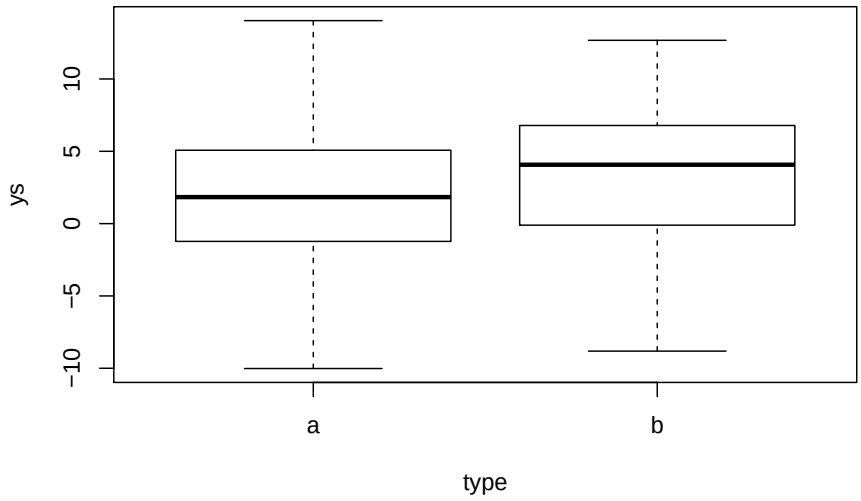
This is required because in a linear model, all the other parameters associated with levels of a factor will be compared to a reference value, that of the intercept, which happens to be the mean under treatment a. Below you will see more examples of this.

Note we were able to detect the null was false, but this was because we had a decent sample size compared to the variance of the measurements and the magnitude of the true effect (the difference of the means). If we keep the sample size constant but we increase the noise or decrease the magnitude of the difference, we might not get the same result, and make a type II error!

```
#define 2 means
ms=c(3,4)
#increase the variance of the process
ys=ifelse(type=="a",ms[1],ms[2])+rnorm(n,0,5)
```

Look at the data, we can see much more variation

```
plot(ys~type)
```



Now, we can run the usual t-test

```
t.test(ys~type)
```

```
##
##  Welch Two Sample t-test
##
## data:  ys by type
## t = -1.3609, df = 97.949, p-value = 0.1767
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.2822693  0.6118174
## sample estimates:
## mean in group a mean in group b
##        2.024963        3.360189
```

and now we can do it the linear regression way

```
lm0=lm(ys~type)
summary(lm0)
```

```
##
## Call:
## lm(formula = ys ~ type)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -12.1746  -3.2719   0.2527   3.0578  12.0085
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.0250     0.6938   2.919  0.00436 **
## typeb         1.3352     0.9811   1.361  0.17667
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.906 on 98 degrees of freedom
## Multiple R-squared:  0.01855,Adjusted R-squared:  0.008533
## F-statistic: 1.852 on 1 and 98 DF,  p-value: 0.1767
```

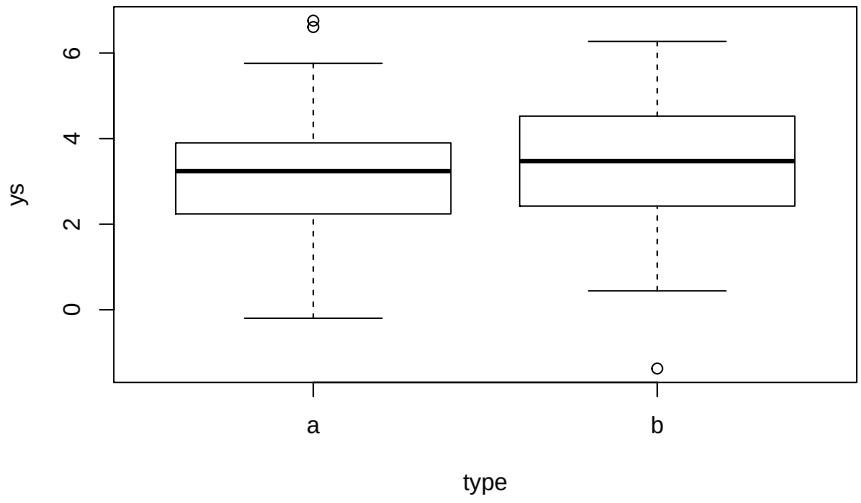
and as you can see, we get the same result for the test statistic, but now with a non significant test.

The same would have happened if we decreased the true difference, while keeping the original magnitude of the error

```
#define 2 means
ms=c(3,3.1)
#increase the variance of the process
ys=ifelse(type=="a",ms[1],ms[2])+rnorm(n,0,1.5)
```

Look at the data, we can see again lower variation, but the difference across treatments is very small (so, hard to detect!)

```
plot(ys~type)
```



Now, we can run the usual t-test

```
t.test(ys~type)
```

```
##
##  Welch Two Sample t-test
##
## data:  ys by type
## t = -0.7994, df = 97.455, p-value = 0.426
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.8149517  0.3469402
## sample estimates:
## mean in group a mean in group b
##        3.158868        3.392874
```

and now we can do it the linear regression way

```
lm0=lm(ys~type)
summary(lm0)
```

```
##
## Call:
## lm(formula = ys ~ type)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -4.7661 -0.9318  0.0812  0.9087  3.5981
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.1589     0.2070  15.261   <2e-16 ***
## typeb         0.2340     0.2927   0.799    0.426
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.464 on 98 degrees of freedom
## Multiple R-squared:  0.006479,Adjusted R-squared:  -0.003659
## F-statistic: 0.639 on 1 and 98 DF,  p-value: 0.426
```

9.2 ANOVA

We move on with perhaps the most famous example of a statistical test/procedure, the ANOVA. An ANOVA is nothing but a linear model, where we have a continuous response variable, which we want to explain as a function of a factor (with several levels, or treatments).

We simulate a data set, beginning by making sure everyone gets the same results by using `set.seed`

```
#Making up an ANOVA
#An ANOVA
#making sure everyone gets the same results
set.seed(12345)
```

Then we define the sample size and the number of treatments

```
#define sample size
n=2000
#define treatments
tr=c("a","b","c","d")
#how many treatments
ntr=length(tr)
#balanced design
n.by.tr=n/ntr
```

now, we can simulate some data. First, the treatments, but we also generate a independent variable that is not really used for now (`xs`).

```
#generate data
xs=runif(n,10,20)
type=as.factor(rep(tr,each=n.by.tr))
#if I wanted to recode the levels such that c was the baseline
#type=factor(type,levels = c("c","a","b","d"))
#get colors for plotting
```

```
cores=rep(1:ntr,each=n.by.tr)
```

Then we define the means by treatment - note that they are different, so the null hypothesis in an ANOVA, that all the means are the same, is false.

```
#define 4 means
ms=c(3,5,6,2)
```

Then, the key part, the response variable, with a different mean by treatment. Note the use of the `ifelse` function, which evaluates its first argument and then assigns the value of its second argument if the first is true or the value of the second if its first argument is false. An example

```
ifelse(3>4,55,77)
```

```
## [1] 77
```

```
ifelse(3<4,55,77)
```

```
## [1] 55
```

Note these can be used nested, leading to possible multiple outcomes, and I use that below to define 4 different means depending on the treatment of the observation

```
ifelse(3<4,55,ifelse(3>2,55,68))
```

```
## [1] 55
```

```
ifelse(3>4,55,ifelse(3>2,666,68))
```

```
## [1] 666
```

```
ifelse(3>4,55,ifelse(3<2,666,68))
```

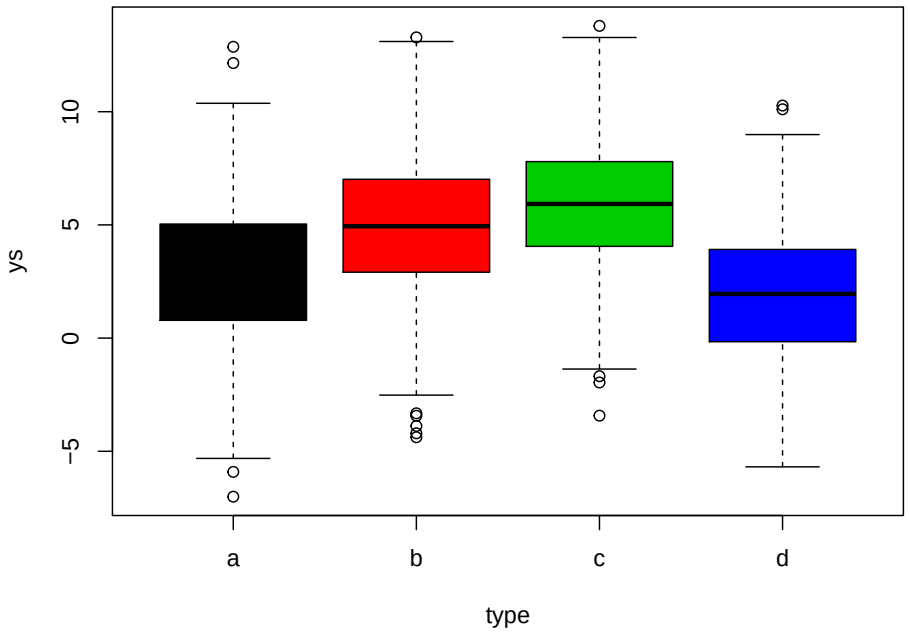
```
## [1] 68
```

So now, generate the data

```
#ys, not a function of the xs!!!
ys=ifelse(type=="a",ms[1],ifelse(type=="b",ms[2],ifelse(type=="c",ms[3],ms[4])))+rnorm(n,0,3)
```

We can actually look at the simulated data

```
par(mfrow=c(1,1),mar=c(4,4,0.5,0.5))
plot(ys~type,col=1:4)
```




```
#abline(h=ms,col=1:4)
```

finally, we can implement the linear model and look at its summary

```
lm.anova=lm(ys~type)
summary(lm.anova)
```

```
##
## Call:
## lm(formula = ys ~ type)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.8735 -2.0115  0.0301  2.0208  9.9976
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.8694     0.1319  21.753  < 2e-16 ***
## typeb         2.0788     0.1865  11.143  < 2e-16 ***
## typec         2.9806     0.1865  15.978  < 2e-16 ***
## typed        -0.8726     0.1865  -4.678 3.09e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.95 on 1996 degrees of freedom
## Multiple R-squared:  0.2163,Adjusted R-squared:  0.2151
```

```
## F-statistic: 183.6 on 3 and 1996 DF,  p-value: < 2.2e-16
```

note that, again, we can manipulate any sub-components of the created objects

```
#see the parameters
lm.anova$coefficients
```

```
## (Intercept)       typeb       typec       typed
##   2.8694412   2.0787628   2.9806367  -0.8726428
```

```
#see the third parameter
lm.anova$coefficients[3]
```

```
##    typec
## 2.980637
```

Not surprisingly, because the means were different and we had a large sample size, everything is highly significant. Note that the ANOVA test is actually presented in the regression output, and that is the corresponding F-test

```
summary(lm.anova)$fstatistic
```

```
##     value     numdf     dendf
##  183.6156    3.0000 1996.0000
```

and we can use the F distribution to calculate the corresponding P-value (note that is already in the output above)

```
ftest=summary(lm.anova)$fstatistic[1]
df1=summary(lm.anova)$fstatistic[2]
df2=summary(lm.anova)$fstatistic[3]
pt(ftest,df1,df2)
```

```
##         value
## 1.402786e-131
```

OK, this is actually the exact value, while above the value was reported as just a small value ($< 2.2 \times 10^{-16}$), but it is the same value, believe me!

Finally, to show (by example) this is just what the ANOVA does, we have the NAOVA itself

```
summary(aov(lm.anova))
```

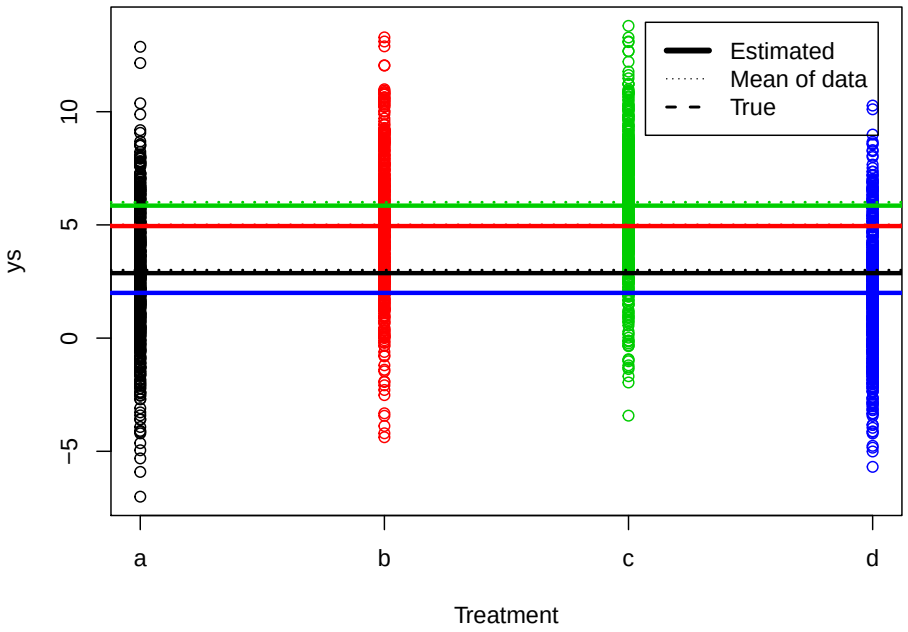
```
##               Df Sum Sq Mean Sq F value Pr(>F)
## type           3   4792  1597.5   183.6 <2e-16 ***
## Residuals   1996  17365     8.7
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

where everything is the same (test statistic, degrees of freedom and p-values).

Conclusion: an ANOVA is just a special case of a linear model, one where we have a continuous response variable and a factor explanatory covariate. In fact, a two way ANOVA is just the extension where we have a continuous response variable and 2 factor explanatory covariates, and, you guessed it, a three way ANOVA means we have a continuous response variable and a 3 factor explanatory covariates.

Just to finish up this example, we could now plot the true means per treatment, the estimated means per treatment

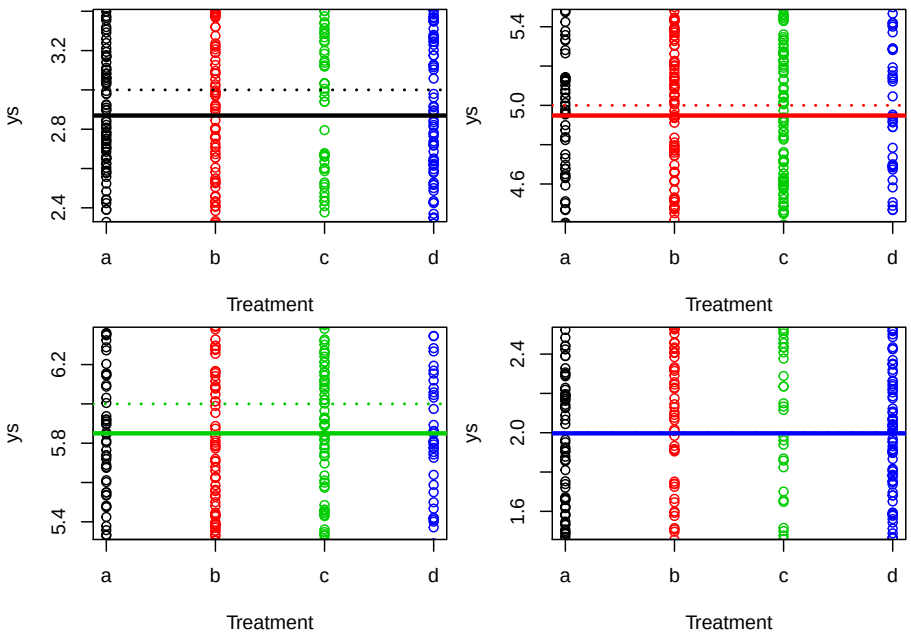
```
par(mfrow=c(1,1),mar=c(4,4,0.5,0.5))
plot(as.numeric(type),ys,col=as.numeric(type),xlab="Treatment",xaxt="n")
axis(1,at=1:4,letters[1:4])
#plot the estimated line for type a
abline(h=lm.anova$coefficients[1],lwd=3,col=1)
#plot the mean line for type a
abline(h=mean(ys[type=="a"]),lwd=1,col=1,lty=2)
#plot the real mean for type a
abline(h=ms[1],lwd=2,col=1,lty=3)
#and now for the other types
abline(h=lm.anova$coefficients[1]+lm.anova$coefficients[2],lwd=3,col=2)
abline(h=mean(ys[type=="b"]),lwd=1,col=2,lty=2)
#plot the real mean for type b
abline(h=ms[2],lwd=2,col=2,lty=3)
abline(h=lm.anova$coefficients[1]+lm.anova$coefficients[3],lwd=3,col=3)
abline(h=mean(ys[type=="c"]),lwd=1,col=3,lty=2)
#plot the real mean for type c
abline(h=ms[3],lwd=2,col=3,lty=3)
abline(h=lm.anova$coefficients[1]+lm.anova$coefficients[4],lwd=3,col=4)
abline(h=mean(ys[type=="d"]),lwd=1,col=4,lty=2)
#plot the real mean for type a
abline(h=ms[4],lwd=2,col=4,lty=3)
legend("topright",c("Estimated","Mean of data","True"),lwd=c(4,1,2),lty=c(1,3,2),inset=
```



It's not easy to see because these overlap (large sample size, high precision) but the estimated means are really close to the real means. It's a bit easier to see if we separate in 4 plots and zoom in on the mean of each treatment, but still the blue lines are all on top of each other, since the mean value was estimated real close to truth (truth=2, estimated = 1.9967984).

```
#see this in 4 plots, less blur
par(mfrow=c(2,2),mar=c(4,4,0.5,0.5))
plot(as.numeric(type),ys,col=as.numeric(type),xlab="Treatment",xaxt="n",ylim=mean(ys[type=="a"])+
axis(1,at=1:4,letters[1:4])
#plot the estimated line for type a
abline(h=lm.anova$coefficients[1],lwd=3,col=1)
#plot the mean line for type a
abline(h=mean(ys[type=="a"]),lwd=1,col=1,lty=2)
#plot the real mean for type a
abline(h=ms[1],lwd=2,col=1,lty=3)
#and now for the other types
plot(as.numeric(type),ys,col=as.numeric(type),xlab="Treatment",xaxt="n",ylim=mean(ys[type=="b"])+
axis(1,at=1:4,letters[1:4])
abline(h=lm.anova$coefficients[1]+lm.anova$coefficients[2],lwd=3,col=2)
abline(h=mean(ys[type=="b"]),lwd=1,col=2,lty=2)
#plot the real mean for type b
abline(h=ms[2],lwd=2,col=2,lty=3)
plot(as.numeric(type),ys,col=as.numeric(type),xlab="Treatment",xaxt="n",ylim=mean(ys[type=="c"])+
axis(1,at=1:4,letters[1:4])
```

```
abline(h=lm.anova$coefficients[1]+lm.anova$coefficients[3],lwd=3,col=3)
abline(h=mean(ys[type=="c"]),lwd=1,col=3,lty=2)
#plot the real mean for type c
abline(h=ms[3],lwd=2,col=3,lty=3)
plot(as.numeric(type),ys,col=as.numeric(type),xlab="Treatment",xaxt="n",ylim=mean(ys[ty
axis(1,at=1:4,letters[1:4])
abline(h=lm.anova$coefficients[1]+lm.anova$coefficients[4],lwd=3,col=4)
abline(h=mean(ys[type=="d"]),lwd=1,col=4,lty=2)
#plot the real mean for type a
abline(h=ms[4],lwd=2,col=4,lty=3)
```




```
#legend("bottomright",c("Estimated","Mean of data","True"),lwd=c(4,1,2),lty=c(1,3,2),i
```

Now we can check how we can obtain the estimated means from the actual parameters of the regression model (yes, that is what the regression does, it calculates the expected mean of the response, conditional on the treatment).

This is the estimated mean per treatment, using function `tapply` (very useful function to get any statistics over a variable, inside groups defined by a second variable, here the treatment)

```
tapply(X=ys,INDEX=type,FUN=mean)
```

```
##        a        b        c        d
## 2.869441 4.948204 5.850078 1.996798
```

and checking these are obtained from the regression coefficients. An important

note. When you fit models with factors (like here), the intercept term will correspond to the mean of the reference level of the factor(s). Hence, to get the other means, you always have to sum the parameter of the corresponding level to the intercept. So we do it below

```
#check ANOVA is just computing the mean in each group
lm.anova$coefficients[1]
```

```
## (Intercept)
##    2.869441
```

```
lm.anova$coefficients[1]+lm.anova$coefficients[2]
```

```
## (Intercept)
##    4.948204
```

```
lm.anova$coefficients[1]+lm.anova$coefficients[3]
```

```
## (Intercept)
##    5.850078
```

```
lm.anova$coefficients[1]+lm.anova$coefficients[4]
```

```
## (Intercept)
##    1.996798
```

and we can see these are exactly the same values.

Chapter 10

Class 9: 21 10 2020 - ANCOVA is (also) just a linear model

We move on to Analysis of Covariance, a.k.a. ANCOVA, which is essentially like an ANOVA to which we add a continuous explanatory covariate. The ANCOVA was traditionally used to compare means of an outcome variable between two or more groups taking into account (or to correct for) variability of other variables, called covariates. In other words, ANCOVA allows to compare the adjusted means of two or more independent groups. It's just… another linear model with a fancy name! Words adapted from this (looooooooooong!) link https://www.datanovia.com/en/lessons/ancova-in-r/!

This is an extremely common situation in biology/ecology data. Consider, as an example, you are trying to explain how the weight of a fish depends on its length, but you want to see if that relationship changes per year or site.

Also, remember the dataset we considered in class 7. The data was simulated via this website: https://drawdata.xyz/ and was named `data4lines.csv`. Those had (about) the same slope in 3 groups, and a different slope in a forth group. That could be analysed as an ANCOVA, and we will look at it that way at the end.

Lets simulate some relevant data and fit the models

10.1 Common slope, different intercepts per treatment

We begin with a situation where there are different intercepts per group, but a common slope across all groups. Contrast this with what we saw under the previous class, under chapter 9.

To make it interesting, assume that we are simulating weights for 4 different species, and that weights depend on length (as they almost always do!).

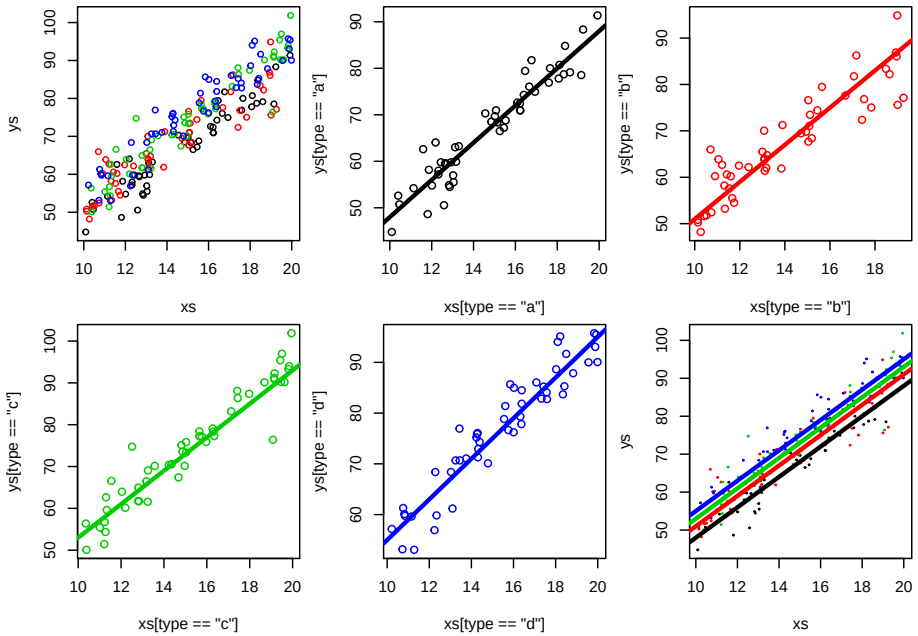
This would be interesting and could be some real data if say one wanted to compare the weights of the fishes of 4 different species, we had captured 50 animals from each species. But we know that the fish lengths across species might be different to begin with, and yet our key interest would be say the weight by species, and in that sense the length was essentially a confounding factor.

```
#all slopes the same, diferent intercepts - no interactions
set.seed(1234)
n<-200
nbygroup<-50
xs <- runif(n,10,20)
tr <- c("a","b","c","d")
type <- rep(tr,each=nbygroup)
cores <- rep(1:4,each=nbygroup)
a<-3
b<-4
error<-4
ys <- a+b*xs+
ifelse(type=="a",5,ifelse(type=="b",8,ifelse(type=="c",10,12)))+rnorm(n,0,4)
```

We plot the data, all together, per group, and at the end adding the generating line to the plot. It's not easy to make sense of it!

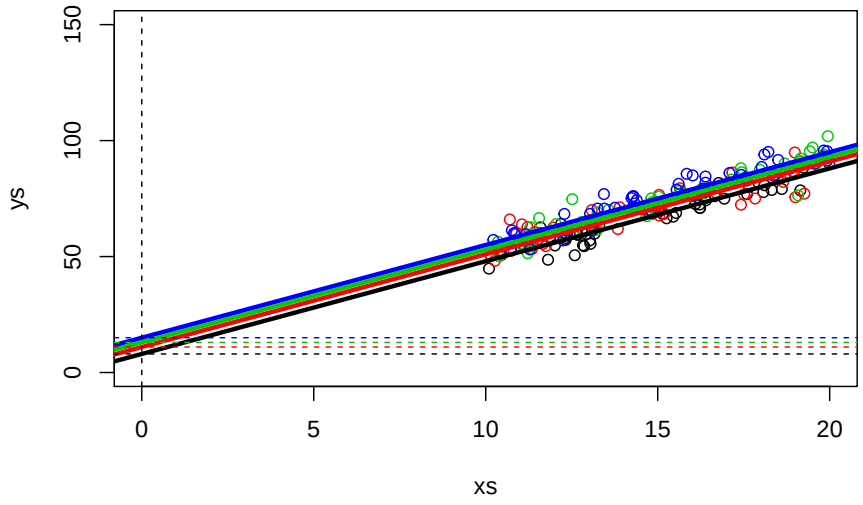
```
par(mfrow=c(2,3),mar=c(4,4,0.5,0.5))
#all the data - uma salganhada!
plot(xs,ys,col=cores,cex=0.8)
#plot the data
#par(mfrow=c(2,2),mar=c(4,4,0.5,0.5))
plot(xs[type=="a"],ys[type=="a"],col=cores[type=="a"])
abline(3+5,4,lwd=3,col=1)
plot(xs[type=="b"],ys[type=="b"],col=cores[type=="b"])
abline(3+8,4,lwd=3,col=2)
plot(xs[type=="c"],ys[type=="c"],col=cores[type=="c"])
abline(3+10,4,lwd=3,col=3)
plot(xs[type=="d"],ys[type=="d"],col=cores[type=="d"])
abline(3+12,4,lwd=3,col=4)
```

```
#the data with each line added to it
#par(mfrow=c(1,1),mar=c(4,4,0.5,0.5))
plot(xs,ys,col=cores,cex=0.2)
abline(3+5,4,lwd=3,col=1)
abline(3+8,4,lwd=3,col=2)
abline(3+10,4,lwd=3,col=3)
abline(3+12,4,lwd=3,col=4)
```



While not the best to look at the data, note that to visually confirm the value of the intercepts we can zoom out on the plot.

```
plot(xs,ys,col=cores,xlim=c(0,20),ylim=c(0,150))
abline(3+5,4,lwd=3,col=1)
abline(3+8,4,lwd=3,col=2)
abline(3+10,4,lwd=3,col=3)
abline(3+12,4,lwd=3,col=4)
abline(h=c(3+5,3+8,3+10,3+12),v=0,col=c(1,2,3,4,1),lty=2)
```



Now we run the corresponding linear model

```
#fit the model
lm.ancova1 <- summary(lm(ys~xs+type))
lm.ancova1
```

```
##
## Call:
## lm(formula = ys ~ xs + type)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.4694  -2.3640   0.2813   2.1063  11.6596
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.24244    1.57462   5.870 1.85e-08 ***
## xs           3.92089    0.09997  39.220  < 2e-16 ***
## typeb        3.11952    0.80410   3.880 0.000143 ***
## typec        5.80393    0.80324   7.226 1.10e-11 ***
## typed        7.36736    0.80434   9.159  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.008 on 195 degrees of freedom
## Multiple R-squared:  0.9011,Adjusted R-squared:  0.8991
```

```
## F-statistic: 444.3 on 4 and 195 DF,  p-value: < 2.2e-16
```

We can check the model intercept coefficients

```
#estimated values of each intercept
lm.ancova1$coefficients[1]
```

```
## [1] 9.242444
```

```
lm.ancova1$coefficients[1]+lm.ancova1$coefficients[3]
```

```
## [1] 12.36196
```

```
lm.ancova1$coefficients[1]+lm.ancova1$coefficients[4]
```

```
## [1] 15.04638
```

```
lm.ancova1$coefficients[1]+lm.ancova1$coefficients[5]
```

```
## [1] 16.60981
```

and the common slope

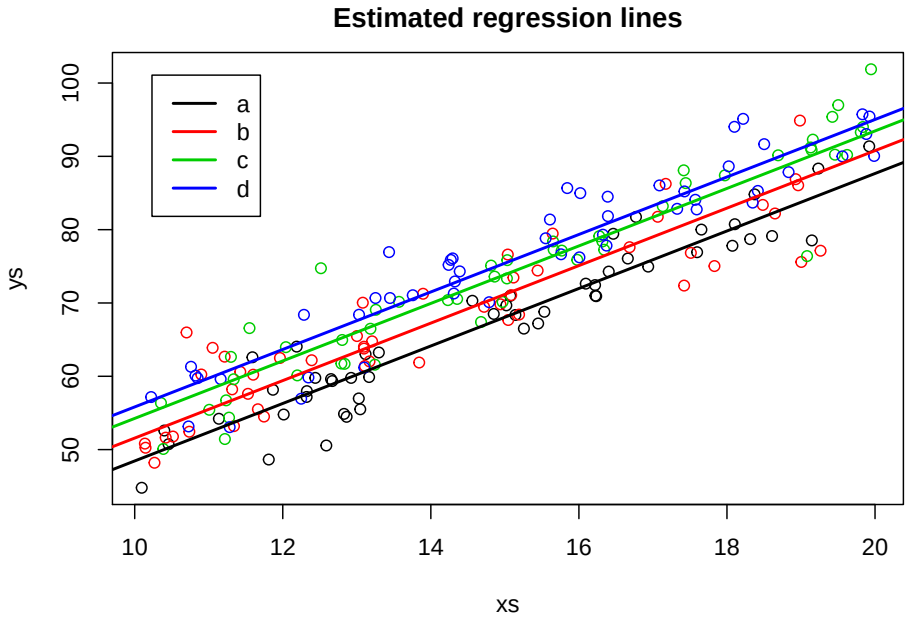
```
lm.ancova1$coefficients[2]
```

```
## [1] 3.920888
```

Check how these values are similar (they are estimates) to those we simulated above, slope was 4, and the intercepts were respectively 3+5,3+8,3+10 and 3+12.

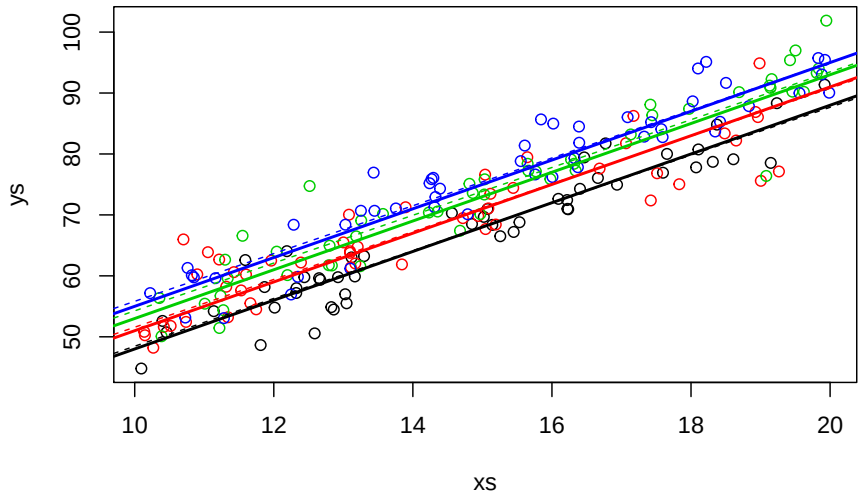
We can plot the estimated regression lines

```
par(mfrow=c(1,1),mar=c(4,4,2.5,0.5))
plot(xs,ys,col=cores,main="Estimated regression lines")
abline(lm.ancova1$coefficients[1],lm.ancova1$coefficients[2],col=1,lwd=2)
abline(lm.ancova1$coefficients[1]+lm.ancova1$coefficients[3],lm.ancova1$coefficients[2],col=2,lwd
abline(lm.ancova1$coefficients[1]+lm.ancova1$coefficients[4],lm.ancova1$coefficients[2],col=3,lwd

abline(lm.ancova1$coefficients[1]+lm.ancova1$coefficients[5],lm.ancova1$coefficients[2],col=4,lwd
legend("topleft",legend = tr,lwd=2,col=1:4,inset=0.05)
```



But because we are in a simulation setting, we can contrast the estimated values agaist the reality (the real model).

```
#In a simulated scenario, we can see we are close to the real values
plot(xs,ys,col=cores)

#plot the lines
abline(a+5,b,lwd=2,col=1)
abline(a+8,b,lwd=2,col=2)
abline(a+10,b,lwd=2,col=3)
abline(a+12,b,lwd=2,col=4)
#gupo a
abline(lm.ancova1$coefficients[1],lm.ancova1$coefficients[2],lwd=1,col=1,lty=2)
#grupo b
# intercept+slope*xs+intercept especifico do grupo b
# (intercept+intercept especifico do grupo b)+ slope
abline(lm.ancova1$coefficients[1]+lm.ancova1$coefficients[3],lm.ancova1$coefficients[2]
#grupo c
abline(lm.ancova1$coefficients[1]+lm.ancova1$coefficients[4],lm.ancova1$coefficients[2]
#grupo d
abline(lm.ancova1$coefficients[1]+lm.ancova1$coefficients[5],lm.ancova1$coefficients[2]
```



As we can see, they are quite close. The error is small compared with the effrect sizes, and the sample size is large enough we can estimate the parameters reasonably well.

But how exactly do we get the predicted intercepts? To understand where they come from we need to see what R does (or, for that matter, what any other software would need to do!) in the background to fit a model with a factor covariate. Remember what the data is

```
#the data
head(data.frame(ys=ys,xs=xs,type=type),10)
```

```
##          ys       xs type
## 1  54.20623 11.13703    a
## 2  70.99310 16.22299    a
## 3  72.63496 16.09275    a
## 4  70.92527 16.23379    a
## 5  79.13262 18.60915    a
## 6  74.28038 16.40311    a
## 7  44.79477 10.09496    a
## 8  57.97476 12.32551    a
## 9  76.06322 16.66084    a
## 10 68.36163 15.14251    a
```

before fitting a factor covariate, we need to replace it by dummy variables (k-1 dummy variables, where k is the number of levels of the factor). Below we look at a set of data lines that allow us to see observations from the diferent `types`

considered

```
#explaining it
data.frame(ys=ys,xs=xs,type=type,typeb=ifelse(type=="b",1,0),typec=ifelse(type=="c",1,0
```

```
##           ys       xs type typeb typec typed
## 1   54.20623 11.13703    a     0     0     0
## 49  59.78224 12.43929    a     0     0     0
## 50  80.00860 17.65460    a     0     0     0
## 51  52.44224 10.73780    b     1     0     0
## 99  64.03652 13.09647    b     1     0     0
## 100 72.37184 17.42120    b     1     0     0
## 101 56.35918 10.35457    c     0     1     0
## 149 93.28892 19.80787    c     0     1     0
## 150 77.14469 15.76813    c     0     1     0
## 151 74.30940 14.39042    d     0     0     1
## 200 81.84481 16.39205    d     0     0     1
```

So R first builds what is known as the design matrix. Notation wise Y= parameters $\times$ design matrix, or Y = βX (see e.g. https://en.wikipedia.org/wiki/Design_matrix)

```
#the design matrix
head(data.frame(xs=xs,typeb=ifelse(type=="b",1,0),typec=ifelse(type=="c",1,0),typed=if
```

```
##         xs typeb typec typed
## 1 11.13703     0     0     0
## 2 16.22299     0     0     0
## 3 16.09275     0     0     0
## 4 16.23379     0     0     0
## 5 18.60915     0     0     0
## 6 16.40311     0     0     0
```

and that is what it uses for the fitting. Therefore, if we want to know the intercept of say `type` c, we need to sum the common intercept with the parameter associates with the dummy variable `typeb`.

This would be an ANCOVA, and here we would conclude that the mean of the response was different for the different levels of z , once accounting for the fact that the xs varied. this is evident since all the coefficients estimates and associated precisions in the summary of the model above would lead to rejecting the null hypothesis that their value was 0, as can be seen by the corresponding very small p-values. Not a surprise, since we simulated them as different and the errors were small.

Taks: Increase the simulated error or lower the coefficients until you get type II errors. Change also sample sizes and effect sizes to see the impacts on the model performance!

Chapter 11

Class 10: 27 10 2020

In class 10 we actually had Miguel Pais talking about Individual Based Models.

In this chapter we look again at the ANCOVA model presented in chapter 10, but under a different perspective. I decided to create this material as a bonus for students, to understand why the ANCOVA is what it is. As a bonus, this also provides a cautionary tale about the dangers of non-random sampling, or more generally, confounding due to unmeasured factors that might affect our response variable.

Therefore,

11.1 Same story, another spin

As we noted above, the ANCOVA would be an useful model to compare means of an outcome variable between two or more groups taking into account (or to correct for) variability of other variables, often called covariates. In other words, ANCOVA allows to compare the adjusted means of two or more independent groups.

Here we tell the same story from chapter 10 under said perspective. We will do so with the help of two unlikely caracters. This is the story of two friends: a biologist that is exploring the weights of lizards, and his friend, a former biology that decided to take an MSc in Ecological Statistics!

The biologist will be the hero of our story. He has a great name. George Ramsey Ernest Armitage Turner. Note that he has 5 names, unusal in the Anglosaxonic world, but not that uncommon is say Portugal. To make it easier, we'll call him just by his initials. So… let's call him Great :) Great's fried, who's a great friend, is simply called John. Boring, but hey, names aren't something you can choose are they, and when you are just the sidekick on the story, you can't complain!

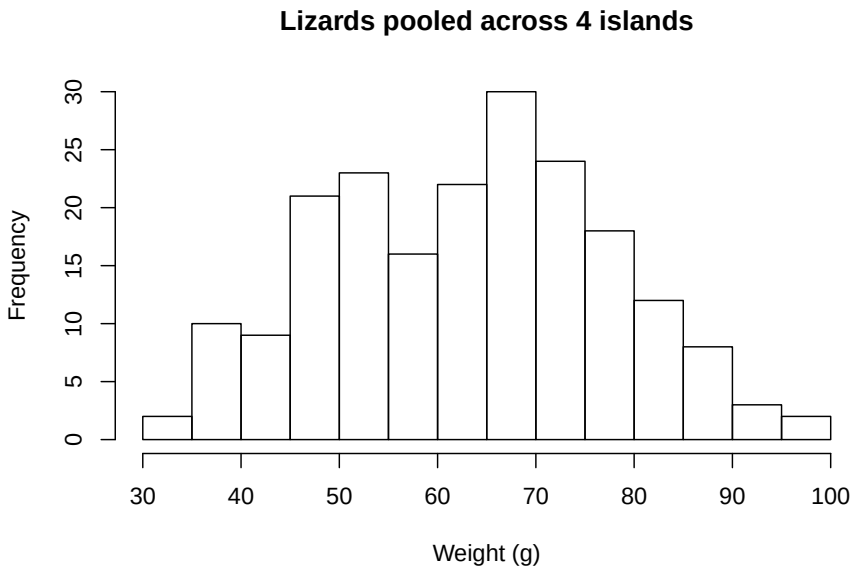
Great when on a journey to a distant archipelago where there are 4 islands, each potentially with a different species of lizard that Great is interested in. The folks providing names to islands where not has imaginative as Great's parents, so the islands are called just A, B, C and D. Imagine that Great did a great job and collected a great sample of lizards in each island. Great is alos interested in the amount of insects available for the lizards in each of the islands. He thinks they might determine the weight of the lizards. Weight is related to condition, condition to fecundity and survival, and so on.

Imagine Great wanted to compare the weights of lizard specimens he collected in each of the 4 islands. He happen to capture a number of animals in each island, and we will label them as A to D, as per the islands.

(note, since this is a story, this time I am not showing you how the data was created (=simulated), for narrative reasons!)

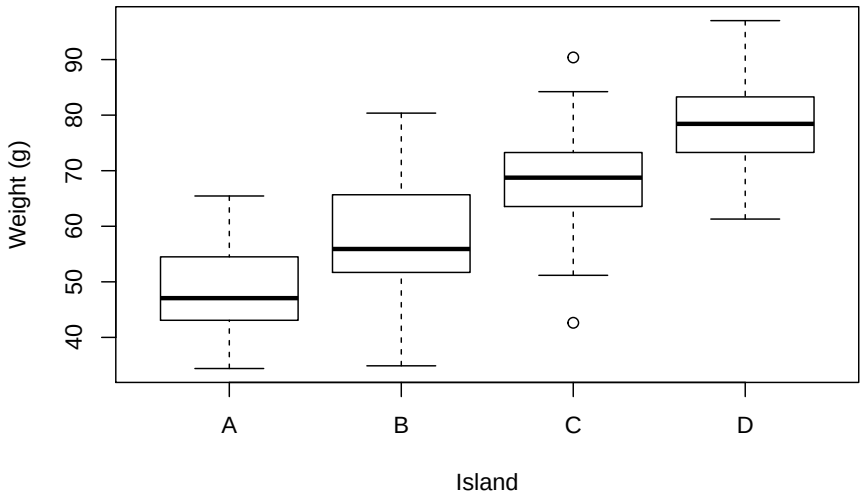
Great ploted the weights, in grams (g), of the captured lizards. These look like this:

```
hist(ys,main="Lizards pooled across 4 islands",xlab="Weight (g)")
```



The distribution is unimodal and about simetrical. When lizards are separated by island, they look like this

```
boxplot(ys~type,ylab="Weight (g)",xlab="Island")
```



There seem to be clear differences in the weights per species, as a standard linear model (e.g. an ANOVA, see 9) will show:

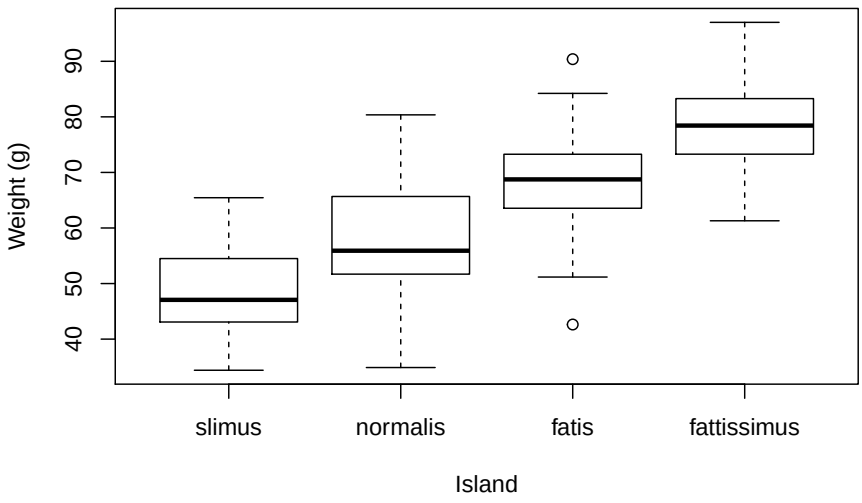
```
summary(lm(ys~type))
```

```
##
## Call:
## lm(formula = ys ~ type)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -25.7066  -5.3458  -0.5474   6.2330  22.9767
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   48.079      1.199  40.105  < 2e-16 ***
## typeB          9.305      1.695   5.489 1.24e-07 ***
## typeC         20.254      1.695  11.947  < 2e-16 ***
## typeD         30.583      1.695  18.039  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.477 on 196 degrees of freedom
## Multiple R-squared:  0.652,Adjusted R-squared:  0.6467
## F-statistic: 122.4 on 3 and 196 DF,  p-value: < 2.2e-16
```

Great is happy, he had seen differents amounts of insects in each island and so he is already thinking about a paper he will write about how the size of the lizards depends on food availability.

Further, he just had a great thought. He calls these GGTs: Great great thoughts. He is thinking about proposing that these correspond to different species in each island, and he is already dreaming about the names of his new species: he is considering naming them "slimus", "normalis", "fatis", "fattissimus", for animals in islands A, B, C and D, respectively. The plot would then read just like this, which looks... you guessed it... great.

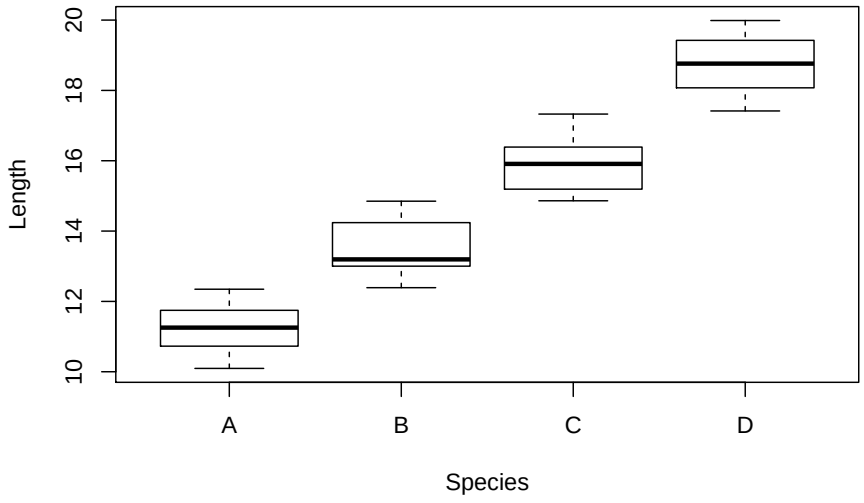
```
spnames <- c("slimus", "normalis", "fatis", "fattissimus")
boxplot(ys~type,ylab="Weight (g)",xlab="Island",
names=spnames)
```



Unfortunately, he goes to the pub and tells John about his findings. John has been doing some modelling courses at the Univeristy and is very interested about sampling. John asks Great a great set of questions: "How did you selected the lizards you captured? What about the lengths of the lizards? Were the animals from each island of about the same length? In other words, did you control the weights for length? Because longer animals will generaly heavier, you know?"

Great had not thought about that yet, indeed. He's feeling dizzy, might be the beers he had, might be the questions he was just asked! He rushes home and looks at the data. And in fact, the different lizards from the different islands have very different lengths to begin with, as we can see in the plot below.

```r
boxplot(xs~type,ylab="Length",xlab="Species")
```




```r
summary(lm(xs~type))
```

```
##
## Call:
## lm(formula = xs ~ type)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.30243 -0.60977 -0.06891  0.52388  1.41528
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.2298     0.1034  108.55   <2e-16 ***
## typeB         2.2413     0.1463   15.32   <2e-16 ***
## typeC         4.6822     0.1463   32.00   <2e-16 ***
## typeD         7.4886     0.1463   51.19   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7315 on 196 degrees of freedom
## Multiple R-squared:  0.9368,Adjusted R-squared:  0.9358
## F-statistic: 968.6 on 3 and 196 DF,  p-value: < 2.2e-16
```

In his mind Great has a vague memory of a teacher in Numerical Ecology that one should explore the data before modelling. He would have avoided this embarassement if he only had done that. Before leaving the pub he heard John saying he should look into ANCOVA's. Something about "you need to test for the weights, accounting for differences in lengths!".

He goes into his books and finds that ANCOVA is just a linear model, where you model a response (weight, he realizes) with a factor (island) and a continous variable (lenght).

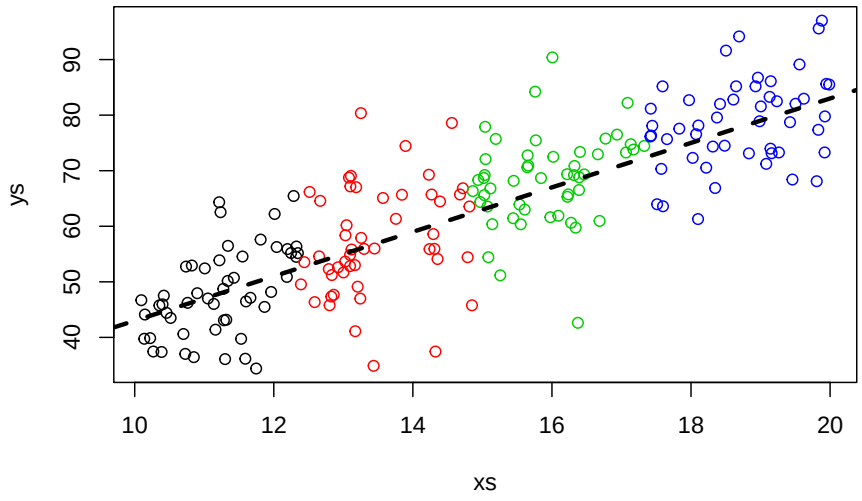
He implements the models and, much to his despair, realizes that, once he you accounts for the length, the weights are not different per island. The damn lizards are exactly the same weight in the different islands once you account for their length... :(

```
summary(lm(ys~xs+type))
```

```
##
## Call:
## lm(formula = ys ~ xs + type)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -27.4613  -4.7367   0.5201   4.2655  23.8079
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.3307     8.8751   0.601    0.549
## xs            3.8067     0.7838   4.857 2.45e-06 ***
## typeB         0.7735     2.3798   0.325    0.746
## typeC         2.4305     4.0058   0.607    0.545
## typeD         2.0764     6.0853   0.341    0.733
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.027 on 195 degrees of freedom
## Multiple R-squared:  0.6896,Adjusted R-squared:  0.6832
## F-statistic: 108.3 on 4 and 195 DF,  p-value: < 2.2e-16
```

He sees his paper further and further farther away. This is what we saw: the same line explains all the data, irrespectively of group. In other words, there is not a different relationship per species between weight and length! His great ecologycal theory goes to the bin!

```
plot(ys~xs,col=cores)
abline(a,b,lwd=3,lty=2)
```



Now, that is dismaying, but interesting. So Great returns to the pub and he asks John: "Would the oposite be possible? Say things looked just the same, yet they were different after accounting for a confounding factor?".

"Yes", John replied: "I have heard about that situation, but have never seen it in a real data set before. Of course that is hard to happen, because *the stars need to align*. But it can happen in theory. Imagine the situation where the relationship between length and weight is different per group. However, out of a strange confounding circumstance, the observed weights happen to be similar, because we sampled (just the right, in this case, wrong!) different lengthts in each species."

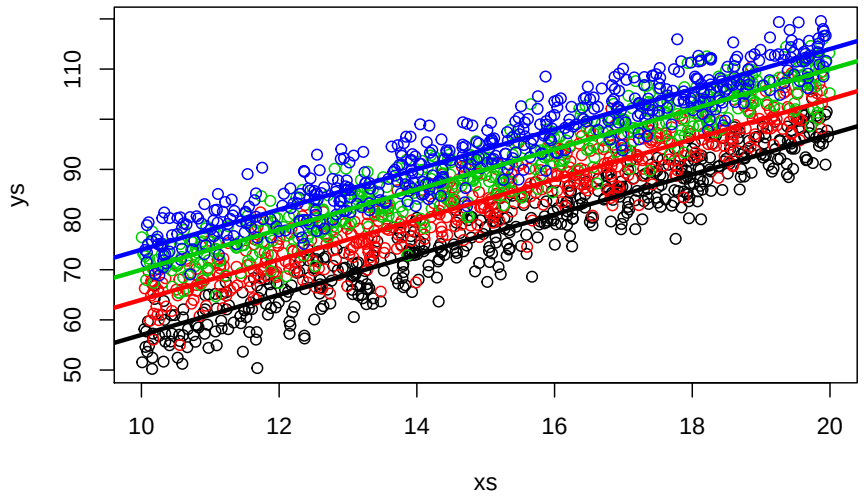
By now Great has a great headache, but he wants to see this with his own eyes, so he goes back home, sits in front of the computer, opens R and decides: "I will simulate this example". That is what we will do here.

Imagine the following example:

The lizards relationship between weight and length is different per island. Say, akin to what we saw before. Just by magin, we have access to all the lizards in the island.

```
#all slopes the same, diferent intercepts - no interactions
set.seed(12345)
n<-2000
nbygroup<-500
xs <- runif(n,10,20)
```

```
island <- c("A","B","C","D")
type <- rep(island,each=nbygroup)
cores <- rep(1:4,each=nbygroup)
a<-12
b<-4
error<-4
ys <- a+b*xs+
ifelse(type=="A",5,ifelse(type=="B",12,ifelse(type=="C",18,22)))+rnorm(n,0,4)
plot(xs,ys,col=cores)
abline(12+5,4,lwd=3,col=1)
abline(12+12,4,lwd=3,col=2)
abline(12+18,4,lwd=3,col=3)
abline(12+22,4,lwd=3,col=4)
```



Now imagine, for the sake of argument, that in all islands we captured lizards with lengths spanning about 2 cm, but in island A we caught animals with about 18 cm, in B with about 16 cm, in c with about 15 cm and in D with about 14 cm, on average. We can simulate that non-random sampling process with respect to lenghth.

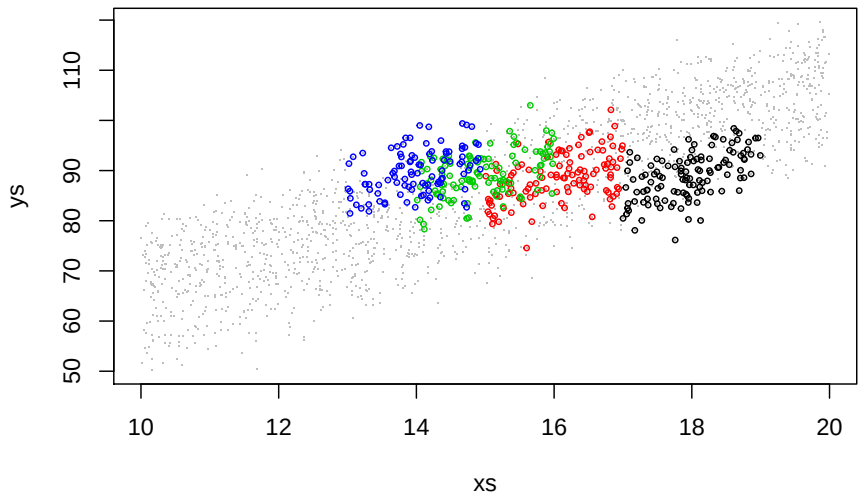
```
sampled.a<-which(xs>17 & xs<19 & type=="A")
sampled.b<-which(xs>15 & xs<17 & type=="B")
sampled.c<-which(xs>14 & xs<16 & type=="C")
sampled.d<-which(xs>13 & xs<15 & type=="D")
sample.all<-c(sampled.a,sampled.b,sampled.c,sampled.d)
```

```
#select a biased sample!
xs2<-xs[sample.all]
ys2<-ys[sample.all]
type2<-type[sample.all]
cores2<-cores[sample.all]
table(type2)
```

```
## type2
##   A   B   C   D
## 119 116 108 100
```

Now, if this is our sample, what happens when we look at the weights alone? First, let's look at the previous plot with the sampled data highlighted in colors and the non sampled data greyed out.

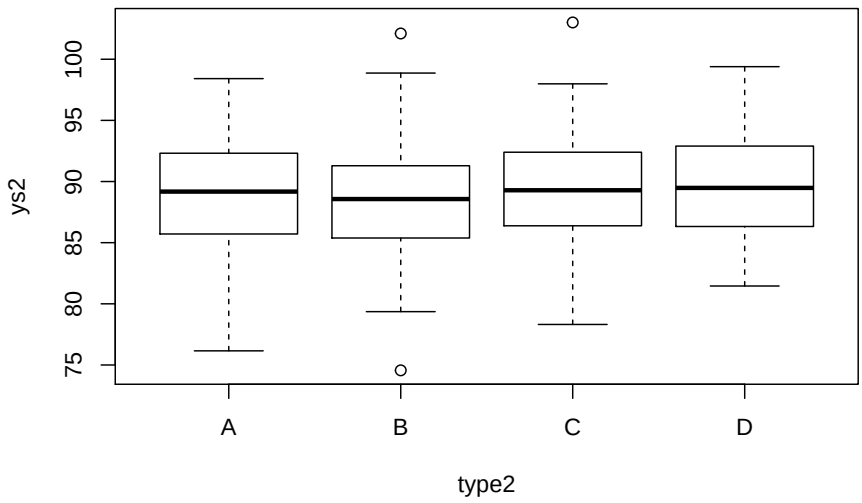
```
plot(xs,ys,pch=".",col="grey")
points(xs2,ys2,col=cores2,cex=0.5)
```




```
#abline(12+5,4,lwd=3,col=1)
#abline(12+12,4,lwd=3,col=2)
#abline(12+18,4,lwd=3,col=3)
#abline(12+22,4,lwd=3,col=4)
```

That was really not a random sample. And non-random samples always ask for trouble. Lets see what happens here. If we look at weights per island, there seems to be no effect:

```
boxplot(ys2~type2)
```



If we test formally for it with an ANOVA, it seems like there is absolutely no effect of weight:

```
summary(lm(ys2~type2))
```

```
##
## Call:
## lm(formula = ys2 ~ type2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.918  -3.137   0.035   3.142  13.648
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  88.9384     0.4194 212.040   <2e-16 ***
## type2B       -0.4589     0.5970  -0.769    0.443
## type2C        0.4200     0.6081   0.691    0.490
## type2D        0.7764     0.6207   1.251    0.212
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.576 on 439 degrees of freedom
```

```
## Multiple R-squared:  0.009992,Adjusted R-squared:  0.003227
## F-statistic: 1.477 on 3 and 439 DF,  p-value: 0.2201
```

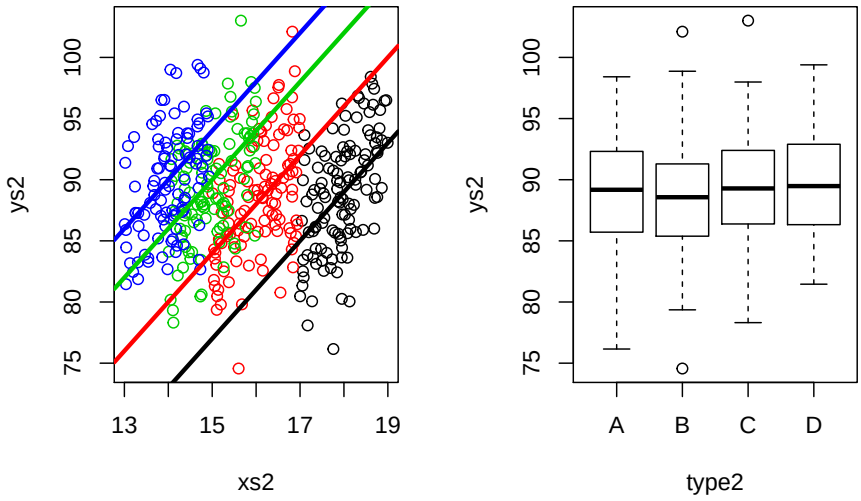
This is where we need to be smart. If we conducts the correct analysis, one that includes and adjusts for the effect of length, the differences in length to weight relationship are clear. The intercepts of the different lines are all different from each other.

```
summary(lm(ys2~type2+xs2))
```

```
##
## Call:
## lm(formula = ys2 ~ type2 + xs2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.1010  -2.8430  -0.0603   2.7281  10.9553
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  18.2503     6.0704   3.006   0.0028 **
## type2B        6.9837     0.8244   8.472  3.7e-16 ***
## type2C       12.1627     1.1384  10.684  < 2e-16 ***
## type2D       16.2717     1.4349  11.340  < 2e-16 ***
## xs2           3.9374     0.3375  11.666  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.001 on 438 degrees of freedom
## Multiple R-squared:  0.2447,Adjusted R-squared:  0.2378
## F-statistic: 35.47 on 4 and 438 DF,  p-value: < 2.2e-16
```

Note this corresponds to comparing weights while not accounting for differences (in lengths), and comparing weights while accounting for those differences. In other words, we are interested in different intercepts in the left plot below, not in the boxplots of the right plot, that ignore the effect of length.

```
par(mfrow=c(1,2))
plot(xs2,ys2,col=cores2)
abline(12+5,4,lwd=3,col=1)
abline(12+12,4,lwd=3,col=2)
abline(12+18,4,lwd=3,col=3)
abline(12+22,4,lwd=3,col=4)
boxplot(ys2~type2)
```

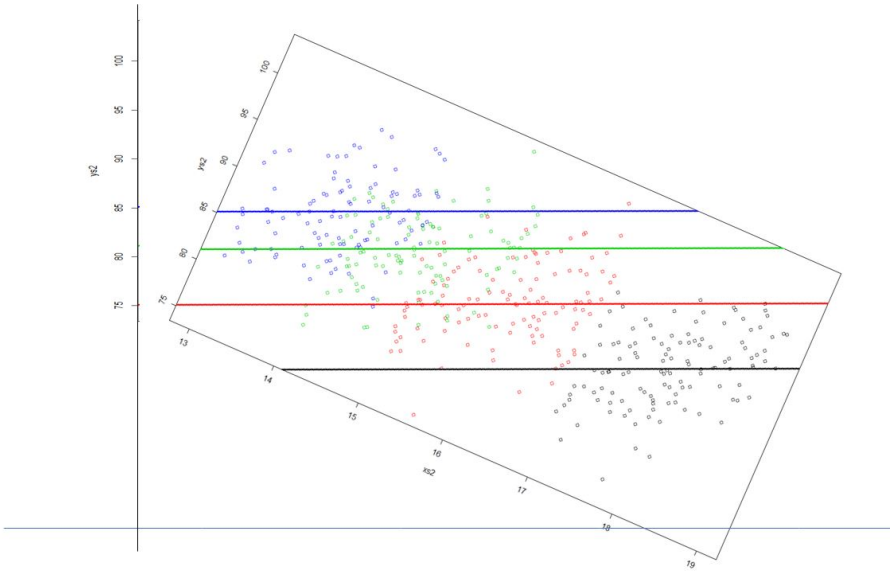


Actually, what the ANCOVA model does is equivalent to looking at the data by rotating the left plot above and see it in the "axis" we care about. That corresponds to the axis such that the slope of the regression lines are aligned with the x-axis of the Cartesian referential.

I want to do this by implementing angular rotation but running out of time. That will involve implementing these transformations:

https://en.wikipedia.org/wiki/Rotation_of_axes

The plot will look just like this!



Naturally these were forced examples, caefully chosen to illustrate a point. But this was really interesting because it:

- illustrates how an ANCOVA is when we test for differences in a response (weight) as a function of a factor (island) while accounting for differences in a quatitative variable (length)
- shows the dangers of testing univariate hypothesis when several (in reality, usually many more than those we can record!) factors have an influence in the response.

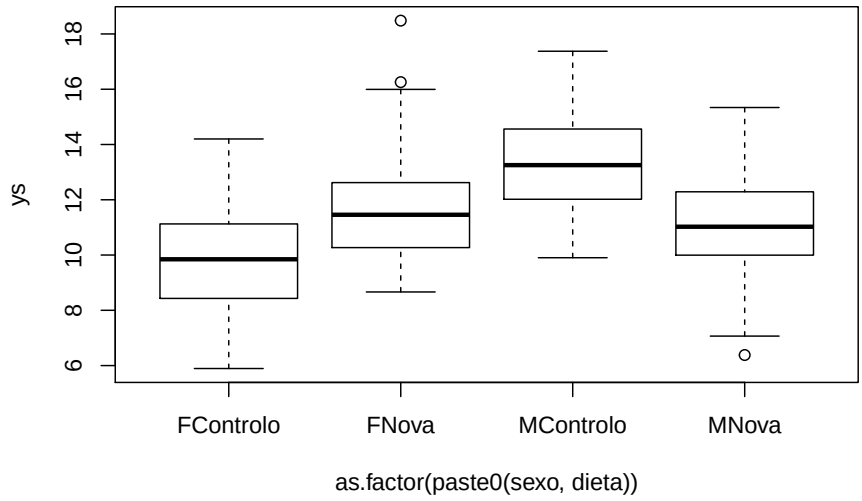
Chapter 12

Class 11: 03 11 2020 ANCOVA with different slopes: interactions

12.1 About interactions

Interactions are useful when the influence of a covariate on the response variable depends on the level of a second covariate. As an example, consider two different diets that we are trying to assess the efficacy on terms of weight gain. We record weight gains for both sexes.

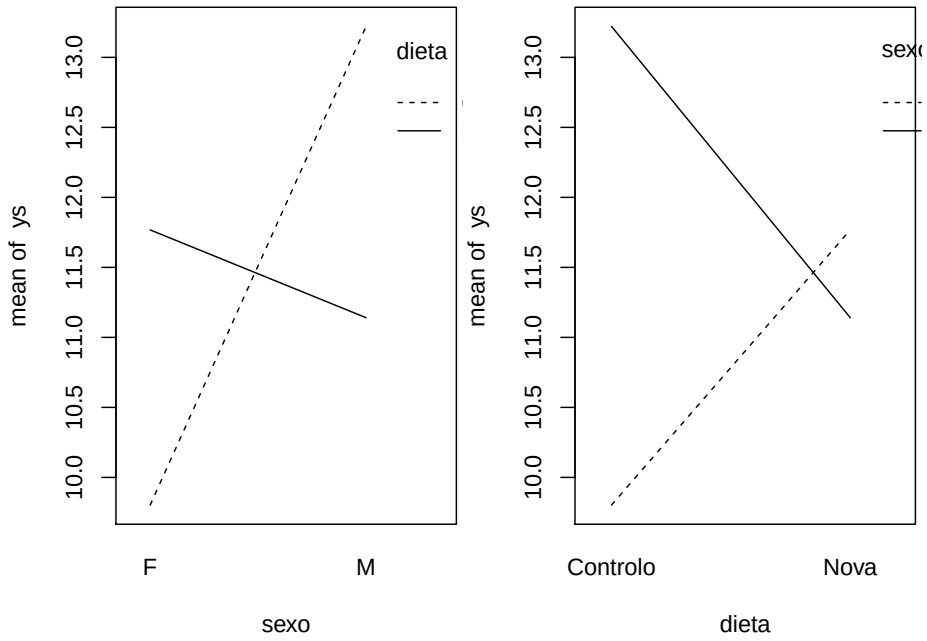
```
#-----------------------------------------------------------
#Interactions
#### with factor covariates
#-----------------------------------------------------------
set.seed(123)
n=100
sexo=rep(c("M","F"),each=n)
dieta=rep(c("Controlo","Nova"),times=n)
ys=10+3*(sexo=="M")+2*(dieta=="Nova")-4*(sexo=="M")*(dieta=="Nova")+rnorm(2*n,mean=0,sd=2)
plot(ys~as.factor(paste0(sexo,dieta)))
```




```
lmSDi=lm(ys~sexo*dieta)
summary(lmSDi)
```

```
##
## Call:
## lm(formula = ys ~ sexo * dieta)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7590 -1.2968 -0.1798  1.1942  6.7145
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       9.8022     0.2673  36.675  < 2e-16 ***
## sexoM             3.4188     0.3780   9.045  < 2e-16 ***
## dietaNova         1.9653     0.3780   5.200 5.00e-07 ***
## sexoM:dietaNova  -4.0457     0.5345  -7.569 1.44e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.89 on 196 degrees of freedom
## Multiple R-squared:  0.3015,Adjusted R-squared:  0.2908
## F-statistic:  28.2 on 3 and 196 DF,  p-value: 3.326e-15
```

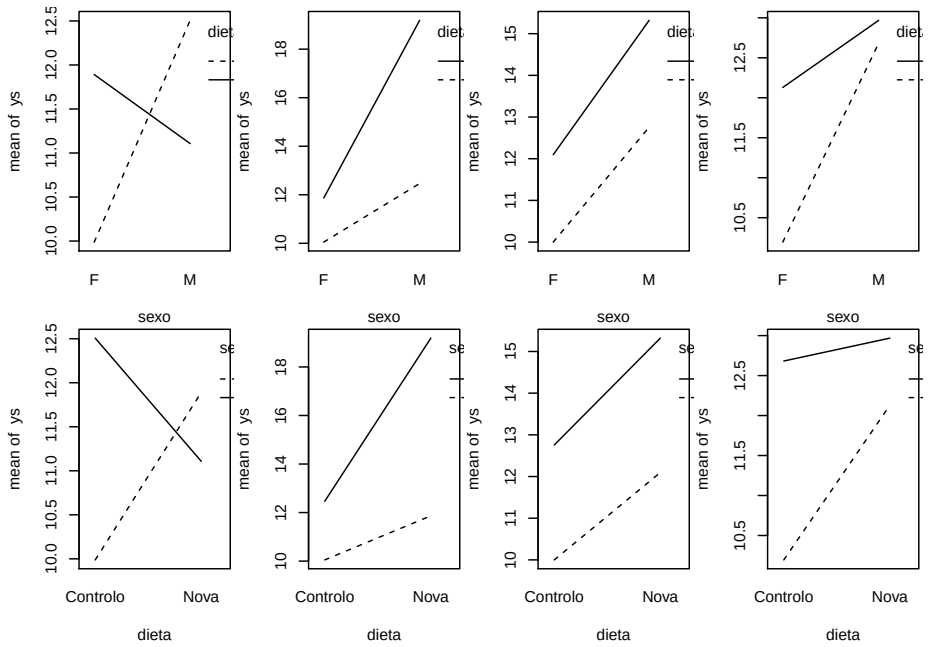
```
par(mfrow=c(1,2),mar=c(4,4,0.2,0.2))
interaction.plot(x.factor=sexo, trace.factor=dieta, response=ys)
interaction.plot(x.factor=dieta, trace.factor=sexo, response=ys)
```



We can see what these interaction plots might look like for different realities.

```
#interaction plots
set.seed(121)
# different interactions and abcense of interaction
par(mfcol=c(2,4),mar=c(4,4,0.2,0.2))
#large negative interaction
ys=10+3*(sexo=="M")+2*(dieta=="Nova")-4*(sexo=="M")*(dieta=="Nova")+rnorm(2*n,mean=0,sd=2)
interaction.plot(x.factor=sexo, trace.factor=dieta, response=ys)
interaction.plot(x.factor=dieta, trace.factor=sexo, response=ys)
#positive interaction
ys=10+3*(sexo=="M")+2*(dieta=="Nova")+4*(sexo=="M")*(dieta=="Nova")+rnorm(2*n,mean=0,sd=2)
interaction.plot(x.factor=sexo, trace.factor=dieta, response=ys)
interaction.plot(x.factor=dieta, trace.factor=sexo, response=ys)
#no interaction
ys=10+3*(sexo=="M")+2*(dieta=="Nova")+rnorm(2*n,mean=0,sd=2)
interaction.plot(x.factor=sexo, trace.factor=dieta, response=ys)
interaction.plot(x.factor=dieta, trace.factor=sexo, response=ys)
#small negative interaction
ys=10+3*(sexo=="M")+2*(dieta=="Nova")-2*(sexo=="M")*(dieta=="Nova")+rnorm(2*n,mean=0,sd=2)
interaction.plot(x.factor=sexo, trace.factor=dieta, response=ys)
```

```
interaction.plot(x.factor=dieta, trace.factor=sexo, response=ys)
```



12.2 Task 1 Implementing the ANCOVA with different slopes

The previous model, explored in Chapters 10 and 11, assumed that the slopes were the same across the different groups. But that might not be the case in many scenarios.

What would change if they were different? We extend the previous case to the scenario where the slope of the relationship is also different per treatment.

We simulate treatments and data, just in the same way as before, but this gives us the option to change things later in this chapter only, and do it separately if we want.

```
#-----------------------------------------------------------------
#all slopes different
set.seed(1234)
xs <- runif(200,10,20)
tr <- c("a","b","c","d")
type <- rep(tr,each=50)
cores <- rep(1:4,each=50)
```

Now we simulate the response

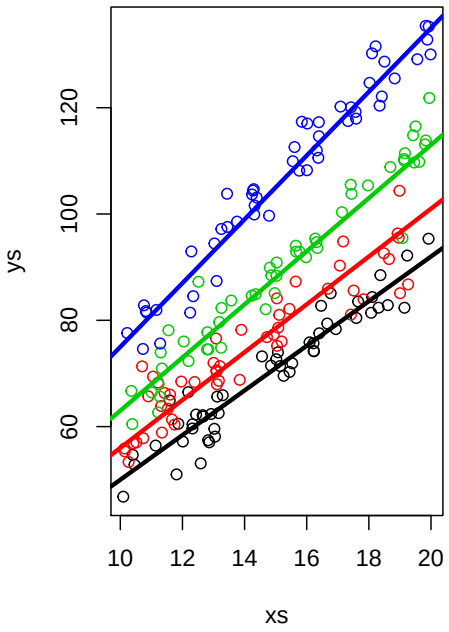
```
ys=3+
ifelse(type=="a",5,ifelse(type=="b",8,ifelse(type=="c",10,12)))+
4*xs+ifelse(type=="a",0.2,ifelse(type=="b",0.5,ifelse(type=="c",1,2)))*xs+
rnorm(200,0,4)
```

If the above code is opaque, we present a different implementation below. Note the code that follows is just the same as the code above, but it might be simpler to understand that the setting implies that we have both different intercepts and slopes per treatment.

```
#same as
intercept=3+ifelse(type=="a",5,ifelse(type=="b",8,ifelse(type=="c",10,12)))
slope=xs*(4+ifelse(type=="a",0.2,ifelse(type=="b",0.5,ifelse(type=="c",1,2))))
ys=slope+intercept+rnorm(200,0,4)
```

We can look at the resulting data, as well as the real model that generated the data (as usual, data is the sistematic component induced by the assumed model to which we add some random error)

```
par(mfrow=c(1,2),mar=c(4,4,0.5,0.5))
plot(xs,ys,col=cores)
abline(3+5,4+0.2,lwd=3,col=1)
abline(3+8,4+0.5,lwd=3,col=2)
abline(3+10,4+1,lwd=3,col=3)
abline(3+12,4+2,lwd=3,col=4)
```



As before, it is actually not that easy to confirm the slopes and intercepts are different, as the intercept is not shown in the above plot. We can zoom out the plot to show us the intercepts (Figure 12.1), which are by definition, where the lines cross the vertical dashed line, i.e., when x=0 in the Cartesian referential.

```
plot(xs,ys,col=cores,xlim=c(0,20),ylim=c(0,150))
abline(3+5,4+0.2,lwd=3,col=1)
abline(3+8,4+0.5,lwd=3,col=2)
abline(3+10,4+1,lwd=3,col=3)
abline(3+12,4+2,lwd=3,col=4)
abline(h=c(3+5,3+8,3+10,3+12),v=0,col=c(1,2,3,4,1),lty=2)
```

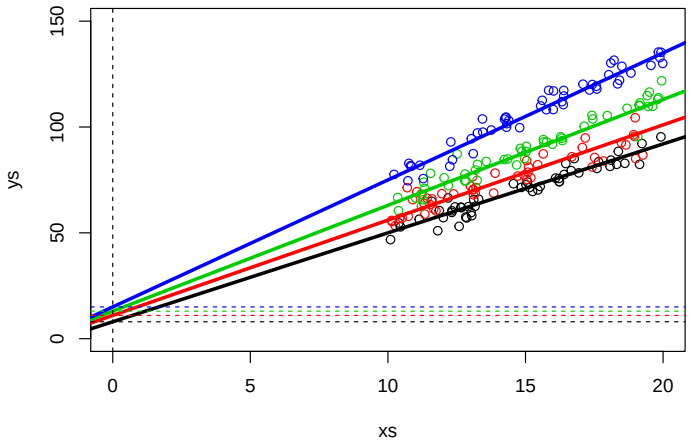


Figure 12.1: Zooming out on the data so that the (fifferent) intercepts are visible

Now, we implement the ANCOVA linear model, but with an *interaction* term between the `type` and `xs`. An interaction between two variables, say `A` and `B`, is defined in R syntax as `A*B`, and so for the corresponding linear model we specify

```
lm.ancova2=lm(ys~xs+type+xs*type)
sum.lm.ancova2=summary(lm.ancova2)
```

We can look at the output of the model

```
sum.lm.ancova2
```

```
##
## Call:
## lm(formula = ys ~ xs + type + xs * type)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.5511  -2.7070   0.2239   2.2766  11.3763
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.9570     3.1985   2.488 0.013707 *
## xs            4.2084     0.2143  19.638  < 2e-16 ***
## typeb        10.6830     4.2375   2.521 0.012513 *
## typec         2.8307     4.3047   0.658 0.511588
## typed         7.5008     4.4644   1.680 0.094556 .
## xs:typeb     -0.2342     0.2889  -0.811 0.418617
## xs:typec      0.9924     0.2836   3.500 0.000579 ***
## xs:typed      1.7875     0.2927   6.106 5.54e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.958 on 192 degrees of freedom
## Multiple R-squared:  0.965,Adjusted R-squared:  0.9637
## F-statistic: 755.2 on 7 and 192 DF,  p-value: < 2.2e-16
```

This is an output similar to the corresponding ANOVA table (implemented via `aov`, the R function that produces ANOVA tables from expressions akin to linear models). The difference is that in such a case the outputs come in terms of the variables, not their levels. This would be

```
summary(aov(ys~xs+type+xs*type))
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## xs            1  49119   49119 3134.87  < 2e-16 ***
## type          3  32682   10894  695.28  < 2e-16 ***
## xs:type       3   1034     345   21.99 2.77e-12 ***
## Residuals   192   3008      16
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that the overall F statistic from the regression model has an F-statistic of 755.2, with 7 and 192 degrees of freedom. That corresponds to the composite test with the null hypothesis "are all parameters equal to 0", which in the ANOVA table, is separated in 3 testes, one for each parameter, with 1, 3 and 3 degrees of freedom each. The residual degrees of freedom are naturally the same in all these tests.

Naturally, we can now evaluate the values of the estimated coefficients, and in particular we can use them to estimate the corresponding regression lines per group. For type `a` we have this

```
#type a
lm.ancova2$coefficients[1];lm.ancova2$coefficients[2]
```

```
## (Intercept)
##    7.957025
```

```
##       xs
## 4.208363
```

in other words, ys=7.9570252+4.2083626 × xs , for type b we have this

```
#type b
lm.ancova2$coefficients[1]+lm.ancova2$coefficients[3];lm.ancova2$coefficients[2]+lm.anc
```

```
## (Intercept)
##    18.64006
```

```
##       xs
## 3.974212
```

in other words, ys=18.640062+3.9742116 × xs, for type c we have this

```
#type c
lm.ancova2$coefficients[1]+lm.ancova2$coefficients[4];lm.ancova2$coefficients[2]+lm.anc
```

```
## (Intercept)
##    10.78774
```

```
##       xs
## 5.200757
```

in other words, ys=10.7877404+5.2007572 × xs, and for type d we have this

```
#type d
lm.ancova2$coefficients[1]+lm.ancova2$coefficients[5];lm.ancova2$coefficients[2]+lm.anc
```

```
## (Intercept)
##    15.45779
```

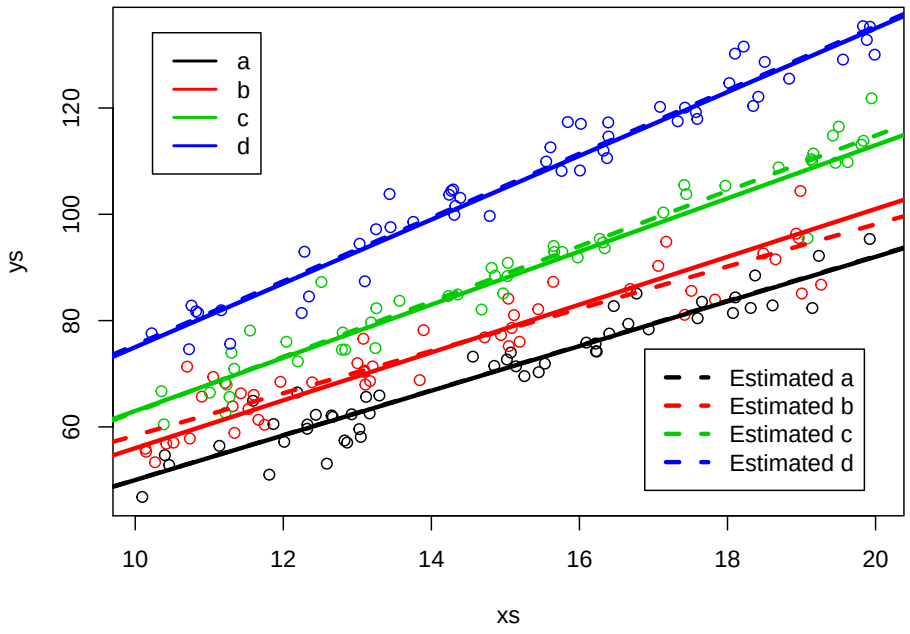
```
##       xs
## 5.995865
```

in other words, ys=15.4577929+5.995865 × xs.

we can now add these to the earlier plots, to see how well we have estimated the different lines per treatment

```
#real lines
par(mfrow=c(1,1),mar=c(4,4,0.5,0.5))
plot(xs,ys,col=cores)
abline(3+5,4+0.2,lwd=3,col=1)
abline(3+8,4+0.5,lwd=3,col=2)
abline(3+10,4+1,lwd=3,col=3)
```

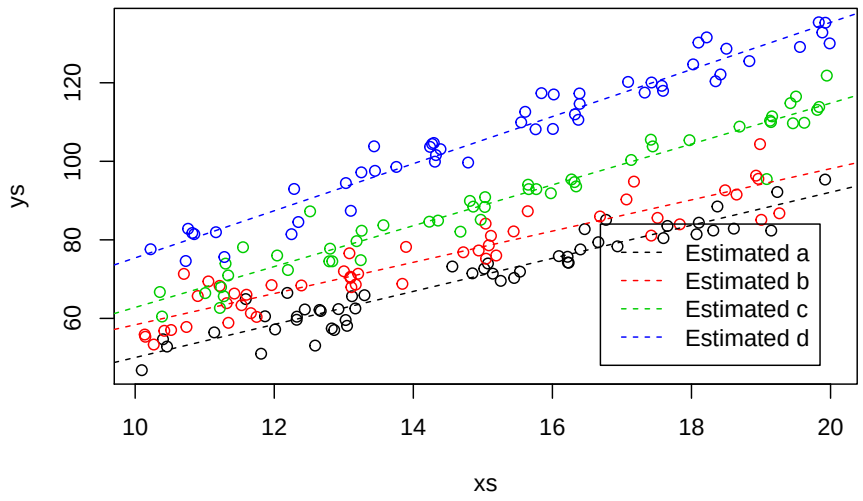
```
abline(3+12,4+2,lwd=3,col=4)
#estimated lines
#type a
abline(lm.ancova2$coefficients[1],lm.ancova2$coefficients[2],lty=2,col=1,lwd=3)
#type b
abline(lm.ancova2$coefficients[1]+lm.ancova2$coefficients[3],
lm.ancova2$coefficients[2]+lm.ancova2$coefficients[6],lty=2,col=2,lwd=3)
#type c
abline(lm.ancova2$coefficients[1]+lm.ancova2$coefficients[4],
lm.ancova2$coefficients[2]+lm.ancova2$coefficients[7],lty=2,col=3,lwd=3)
#type b
abline(lm.ancova2$coefficients[1]+lm.ancova2$coefficients[5],
lm.ancova2$coefficients[2]+lm.ancova2$coefficients[8],lty=2,col=4,lwd=3)
legend("topleft",legend = tr,lwd=2,col=1:4,inset=0.05)
legend("bottomright",legend =paste("Estimated",tr),lwd=3,lty=2,col=1:4,inset=0.05)
```



Remember, if this was a real analysis, you would not know the truth, so at best, you would be able to see the predicted lines, but not the real lines, just as in the plot below

```
# In real life, we only see this
plot(xs,ys,col=cores)
#plot the lines
abline(lm.ancova2$coefficients[1],lm.ancova2$coefficients[2],lwd=1,col=1,lty=2)
abline(lm.ancova2$coefficients[1]+lm.ancova2$coefficients[3],lm.ancova2$coefficients[2]+lm.ancova
```

```
abline(lm.ancova2$coefficients[1]+lm.ancova2$coefficients[4],lm.ancova2$coefficients[2]
abline(lm.ancova2$coefficients[1]+lm.ancova2$coefficients[5],lm.ancova2$coefficients[2]
legend("bottomright",legend =paste("Estimated",tr),lwd=1,lty=2,col=1:4,inset=0.05)
```



It is interesting to note that the slopes were in general closer to the true values than the intercepts.

```
library(knitr)
#true intercepts
TI<-unique(3+ifelse(type=="a",5,ifelse(type=="b",8,ifelse(type=="c",10,12))))
#true slopes
TS<-unique(4+ifelse(type=="a",0.2,ifelse(type=="b",0.5,ifelse(type=="c",1,2))))
#estimated intercepts
EI<-round(c(lm.ancova2$coefficients[1],lm.ancova2$coefficients[1]+lm.ancova2$coefficie
#estimated slopes
ES<-round(c(lm.ancova2$coefficients[2],lm.ancova2$coefficients[2]+lm.ancova2$coefficie
#pooled table
kable(cbind(c("Type","Intercept (true)","Intercept (estimated)","Slope (true)","Slope
```

		(Intercept)	(Intercept)	(Intercept)	(Intercept)
	Type	a	b	c	d
TI	Intercept (true)	8	11	13	15
EI	Intercept (estimated)	8	18.6	10.8	15.5
TS	Slope (true)	4.2	4.5	5	6
ES	Slope (estimated)	4.2	4	5.2	6

Can you think about an intuitive reason for that being the case? The answer lies in figure 12.1. There is no data near the intercept, so there is actually more information abot the slope than about the intercept. Things could have been different if the data for the continous response was also available around the value 0.

12.3 Task 2 Modeling a data set

In a given dataset we might want to explain a continous variable as a function of a couple of explanatory variables, specifically a continous variable and a factor. As we have seen before, this could be attempted via an ANCOVA, with or without an interaction term.Therefore, a relevant question might be to know if the interaction is needed or not, or in other words, if the different lines expressing the relationship between the countinous covariate and the response might be have different slopes per level of the factor covariate, or if on the other hand, a comon slope might be possible, with different per level of the factor covariate. A third even simpler model is the one where the factor covariate is not relevant at all, or, in other words, single line would be the most parcimonious way to model the data.

We illustrate that with the data set `data4lines.csv` that we considered before in chapter 8, which was created using the website https://drawdata.xyz/.

To make our story a bit more intersting we again come up with a narrative. A biologist is interested in estimating what might have been the weight of a fish for which he found an operculum in the in the stomach of a dead otter (*Lutra lutra*). The operculum was 300 mm in diameter. He knows the fish was a barbel (genera: *Barbus*). We will call this biologist Carlos. His wife is called Conceição, and Conceição enjoys thinking about hard questions. She thinks *stongly* about all the hard questions, and when she does her eyebrows raise in a way that Carlos fears. Carlos found the dead otter in the margins of the Guadiana river in Portugal. It seems hard to believe that the otter might have eaten such a large fish. It might have been the largest fish ever eaten by a *Lutra lutra*, which the biologist would love to report in the next Ichtiology congress. To answer his question he visited the museum and he was able to find a sample of fish from the genus *Barbus*. Therefore he labouriosly weighed fish and measured opeculuns… The resulting data are our data!

Carlos reads the data in

```
folder<-"extfiles/"
#folder<-"../Aula7 14 10 2020/"
d4l <- read.csv(file=paste0(folder,"data4lines.csv"))
n <- nrow(d4l)
```

He plots the data ad he sees a pattern that seems to be somewaht linear. He knows about linear regression and he fits a line to the data, and then shows the

resulting plot with data, model and predictions to Conceição.

```
lm0 <- lm(y~x,data=d4l)
```

Carlos is winning, claiming that unfortunately, he suspects that the fish was not the largest ever reported. A Spanish ictiologist reported an otter having eaten a fish with 220 grams. And with his model, he predicts the opperculum belonged to a fish with a weight of about 213.7 gr (12.2).

```
#plot all the data
plot(y~x,xlab= "Operculum diameters (mm)",ylab="Fish length (gr)",data=d4l)
abline(lm0,lty=1,lwd=3)
abline(v=sizeo,h=lm0$coefficients[1]+lm0$coefficients[2]*sizeo,lty=2)
```

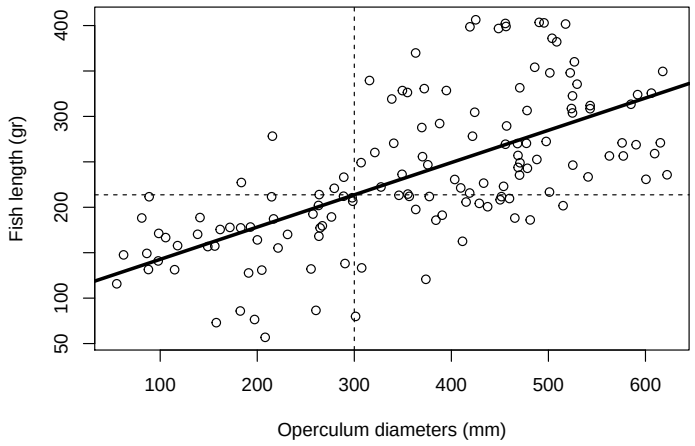
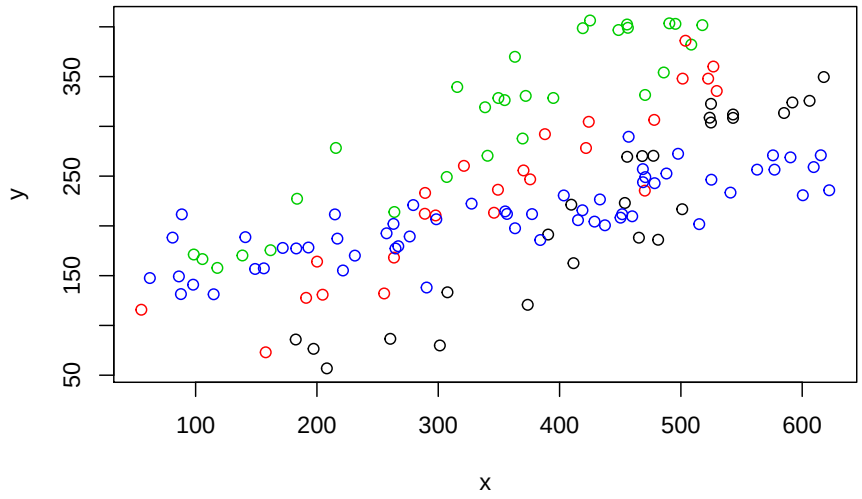


Figure 12.2: Pooled model of fish weight as a function of operculum diameter. Dashed lines represent predictions for a fish with a 300 mm operculum.

Conceição looks at the plots, wonders once again why she married Carlos, and asks: "What was the actual species of the operculum in your dead otter? And by the way, please, for the last time, can you put the dead otter in the garbage? It is the second time that I confuse it with a rabbit in the freezer...! One day you might have a bad surprise for dinner!". Carlos thinks about this as says, it was *Barbus sclateri*. And Maria says: "Well, is the relationship between operculum diameter and fish length the same for all the species?". Carlos - perhaps not the sharpest tool in the box as we by now have come to realize - nonetheless manages to realize what Conceição is asking about. He knows the animals in the museum were identified to the species level, and he redoes the plot coloring the points by species.

```
plot(y~x,col=as.numeric(as.factor(z)),data=d4l,pch=1)
```



And the question that now arises in Carlos mind is, what might the best model to represent this data set, where he has measurements of two numeric variables, `weight~diameter` across 4 groups of observations, defined by `species`.

Carlos decides he can fit a linear model with both `x` and `z` as independent variables, without an interaction. Note this is the conventional ANCOVA model, just as we did before.

```
#fit model per group
lmANC<-lm(y~x+z,data=d4l)
summary(lmANC)
```

```
##
## Call:
## lm(formula = y ~ x + z, data = d4l)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -90.01 -35.01   2.54  35.51 108.10
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  45.51813   13.83069   3.291  0.00128 **
## x             0.39657    0.02516  15.763  < 2e-16 ***
```

```
## zb           54.92376   12.11597   4.533 1.28e-05 ***
## zc          128.20339   11.72572  10.934  < 2e-16 ***
## zd           22.82412   10.26509   2.223  0.02787 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 42.5 on 133 degrees of freedom
## Multiple R-squared:  0.7332,Adjusted R-squared:  0.7252
## F-statistic: 91.38 on 4 and 133 DF,  p-value: < 2.2e-16
```

Maria reminds him again that life is more complicated than what he'd like to imagine: "Why don't you try a model with an interaction term between opperculum diameter and species?". And so he does.

```
#fit model per group, with interaction
lmlinesI<-lm(y~x+z+x:z,data=d4l)
summary(lmlinesI)
```

```
##
## Call:
## lm(formula = y ~ x + z + x:z, data = d4l)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -77.569 -15.787   2.848  17.776  60.875
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -84.72064   19.98946  -4.238 4.24e-05 ***
## x             0.69346    0.04380  15.831  < 2e-16 ***
## zb          109.96907   25.87243   4.250 4.04e-05 ***
## zc          187.01780   24.87546   7.518 8.03e-12 ***
## zd          223.27435   21.82006  10.233  < 2e-16 ***
## x:zb         -0.08161    0.06222  -1.312    0.192
## x:zc         -0.08902    0.05954  -1.495    0.137
## x:zd         -0.49630    0.04926 -10.075  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.09 on 130 degrees of freedom
## Multiple R-squared:  0.8861,Adjusted R-squared:   0.88
## F-statistic: 144.5 on 7 and 130 DF,  p-value: < 2.2e-16
```

It seems like, based on AIC, the model with the interaction is the most parsimonious (remember: most parcimonious model is the one with lowest AIC)!

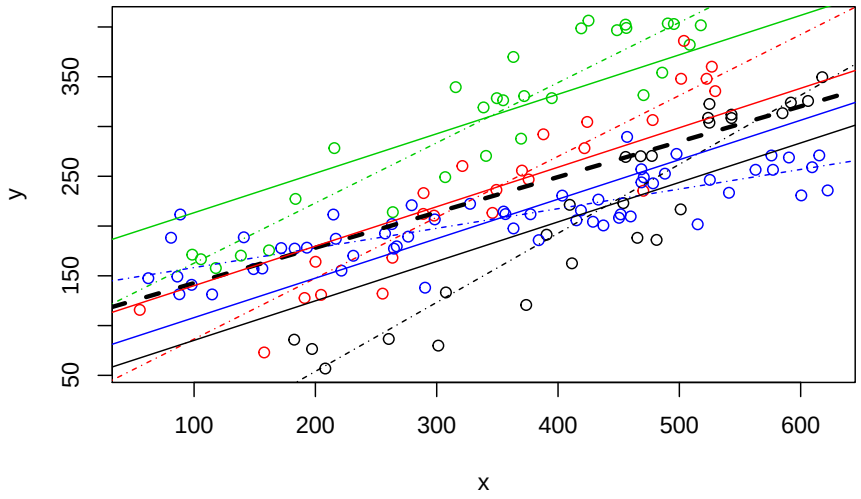
```
AIC(lm0,lmANC,lmlinesI)
```

```
##           df      AIC
## lm0        3 1533.741
## lmANC      6 1433.414
## lmlinesI   9 1321.927
```

Which if you ignore the story for a moment, makes total sence, since that indeed we had one line for one of the groups (`z`) that had a different slope! And that is the significant interaction term above, indicating it is different from the slope of group `a`.

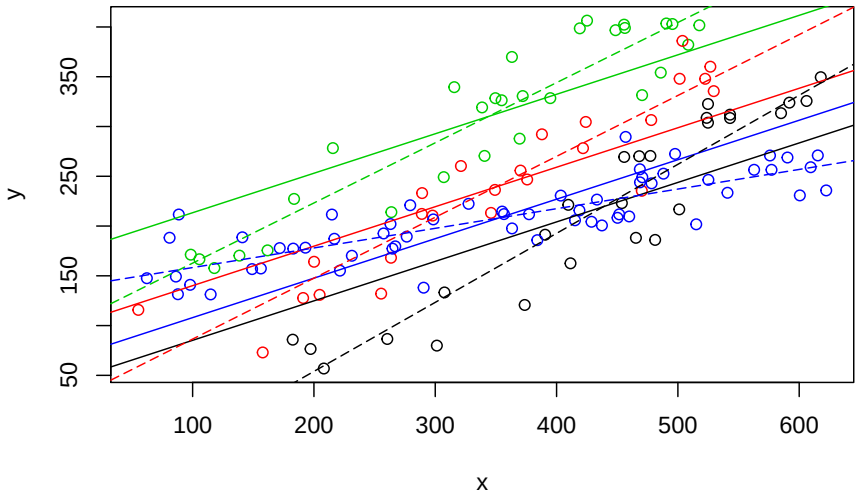
Now lets go back to the data. Remember a plot we had on this dataset before?We noted the plot was messy, including the pooled regression (the thick black line), the regressions fitted to independent data sets, one for each species (museums before!) (the solid lines), and the regressions resulting from the model with species as a factor covariate (dotted-dashed lines).

```
#plot all the data
plot(y~x,col=as.numeric(as.factor(z)),data=d4l,pch=1)
#completely independet regression lines
abline(lm(y~x,data=d4l[d4l$z=="a",]),col=1,lty=4)
abline(lm(y~x,data=d4l[d4l$z=="b",]),col=2,lty=4)
abline(lm(y~x,data=d4l[d4l$z=="c",]),col=3,lty=4)
abline(lm(y~x,data=d4l[d4l$z=="d",]),col=4,lty=4)
#fit model to pooled data
lmlinesG<-lm(y~x,data=d4l)
#these are the wrong lines... why?
abline(lmlinesG,lwd=3,lty=2)
abline(lmANC$coefficients[1],lmANC$coefficients[2],col=1)
abline(lmANC$coefficients[1]+lmANC$coefficients[3],lmANC$coefficients[2],col=2)
abline(lmANC$coefficients[1]+lmANC$coefficients[4],lmANC$coefficients[2],col=3)
abline(lmANC$coefficients[1]+lmANC$coefficients[5],lmANC$coefficients[2],col=4)
```



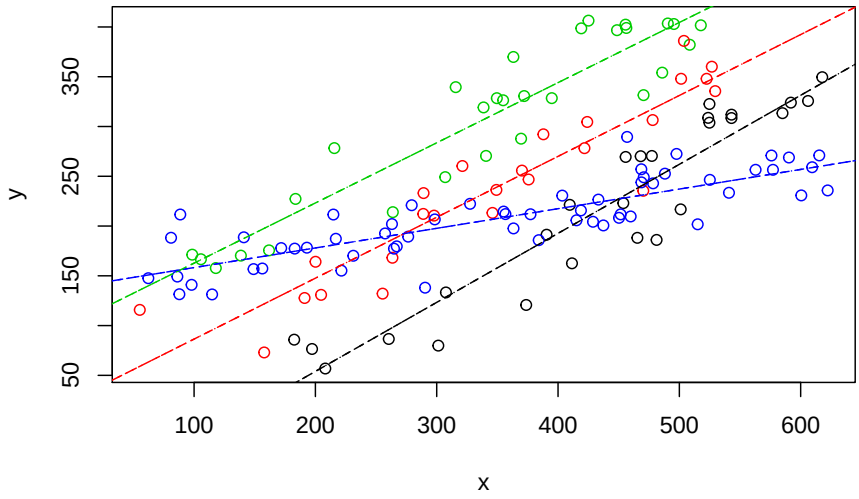
Now we can remove the independent lines (to clean it up a bit!) and just leave the no interaction model estimated values, then add the estimated lines from the interaction model below. Remember, AIC says its the ones from the interactioon model that are the best representation of the data.

```
#plot all the data
plot(y~x,col=as.numeric(as.factor(z)),data=d4l,pch=1)
# #completely independent regression lines
# abline(lm(y~x,data=d4l[d4l$z=="a",]),col=1,lty=4)
# abline(lm(y~x,data=d4l[d4l$z=="b",]),col=2,lty=4)
# abline(lm(y~x,data=d4l[d4l$z=="c",]),col=3,lty=4)
# abline(lm(y~x,data=d4l[d4l$z=="d",]),col=4,lty=4)
# no interaction lines
abline(lmANC$coefficients[1],lmANC$coefficients[2],col=1)
abline(lmANC$coefficients[1]+lmANC$coefficients[3],lmANC$coefficients[2],col=2)
abline(lmANC$coefficients[1]+lmANC$coefficients[4],lmANC$coefficients[2],col=3)
abline(lmANC$coefficients[1]+lmANC$coefficients[5],lmANC$coefficients[2],col=4)
# model with interaction lines
abline(lmlinesI$coefficients[1],lmlinesI$coefficients[2],col=1,lty=5)
abline(lmlinesI$coefficients[1]+lmlinesI$coefficients[3],lmlinesI$coefficients[2]+lmli
abline(lmlinesI$coefficients[1]+lmlinesI$coefficients[4],lmlinesI$coefficients[2]+lmli
abline(lmlinesI$coefficients[1]+lmlinesI$coefficients[5],lmlinesI$coefficients[2]+lmli
```



Likewise, we could compare the lines from independent lines to those of the interaction model.

```
#plot all the data
plot(y~x,col=as.numeric(as.factor(z)),data=d4l,pch=1)
#completely independent regression lines
abline(lm(y~x,data=d4l[d4l$z=="a",]),col=1,lty=4)
abline(lm(y~x,data=d4l[d4l$z=="b",]),col=2,lty=4)
abline(lm(y~x,data=d4l[d4l$z=="c",]),col=3,lty=4)
abline(lm(y~x,data=d4l[d4l$z=="d",]),col=4,lty=4)
# model with interaction lines
abline(lmlinesI$coefficients[1],lmlinesI$coefficients[2],col=1,lty=5)
abline(lmlinesI$coefficients[1]+lmlinesI$coefficients[3],lmlinesI$coefficients[2]+lmlinesI$coeffi
abline(lmlinesI$coefficients[1]+lmlinesI$coefficients[4],lmlinesI$coefficients[2]+lmlinesI$coeffi
abline(lmlinesI$coefficients[1]+lmlinesI$coefficients[5],lmlinesI$coefficients[2]+lmlinesI$coeffi
```

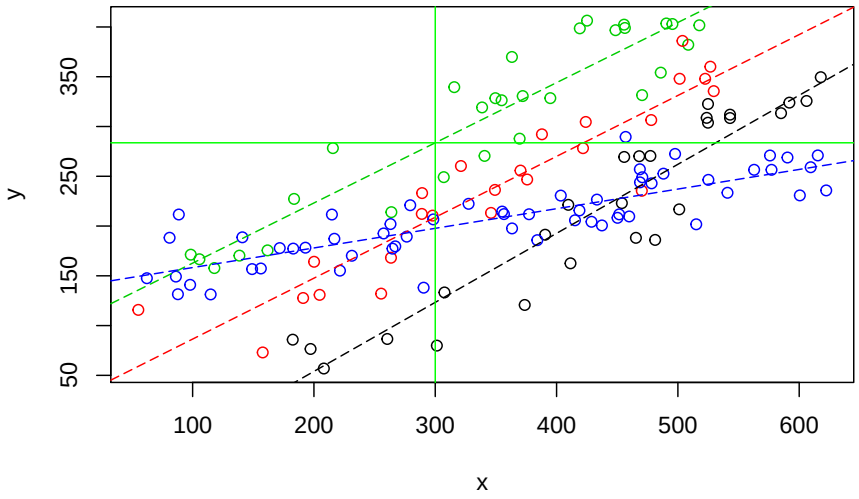


It is interesting to see that they are not very different, which is perhaps surprising but... actually... not surprising. Both use exaclty 8 parameters to describe the data... it's the same thing!!! Linear models are cool :)

OK, I have completely forgoten about Carlos and Conceição... or maybe it's too late and I am just sleepy. But if the right model for *Barbus sclatery*, which happens to correspond to the "green" species, is the one based on the interaction term, then... the fish weight might be 283.6, as shown below.

Carlos might have something to present at the congress after all! He goes to sleep, happy. As he is starting to dream with his "best talk" award he still manages to hear Conceição saying something in the distance like "... have you... considered... variability ... prediction... maybe not realy larger than 220 gr...". The last thought that goes through his mind before the dream turns into a nightmare where he is on that congress naked is... "Why did I married that woman"?

```
#plot all the data
plot(y~x,col=as.numeric(as.factor(z)),data=d4l,pch=1)
#completely independent regression lines
abline(lmlinesI$coefficients[1],lmlinesI$coefficients[2],col=1,lty=5)
abline(lmlinesI$coefficients[1]+lmlinesI$coefficients[3],lmlinesI$coefficients[2]+lmli
abline(lmlinesI$coefficients[1]+lmlinesI$coefficients[4],lmlinesI$coefficients[2]+lmli
abline(lmlinesI$coefficients[1]+lmlinesI$coefficients[5],lmlinesI$coefficients[2]+lmli
abline(v=300,h=lmlinesI$coefficients[1]+lmlinesI$coefficients[4]+(lmlinesI$coefficients
```


y
50
150
250
350
100
200
300
400
500
600
x

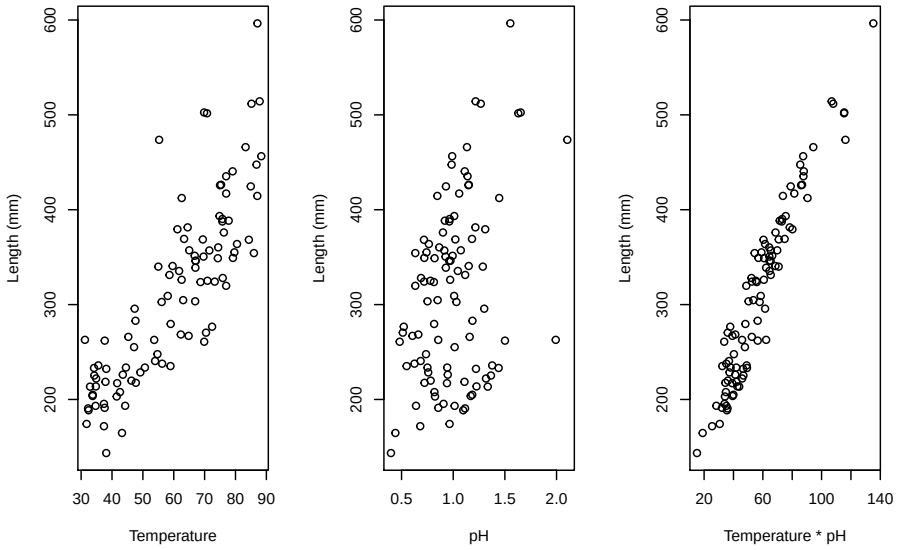
Chapter 13

Class 12: 04 11 2020 Interactions between continous covariates

The examples above were interactions with factors, but interactions can occur also across strictly continous variables. An example follows. Imagine it corresponds to the length of fishes as a function of water temperature `temp` and `pH` (if those were real values of some water pH… all the fish would be dissolved, but any way!). We make up some example data below.

```
#-----------------------------------------------------
#Interactions
#### With continous covariates
#-----------------------------------------------------
#sample size
set.seed(121)
n=100
#get a response variable
xs1=runif(n,30,90)
#get a second variable
xs2=rgamma(n,10,10)
#define the linear predictor
ys=20+2*xs1-4*xs2+3*xs1*xs2+rnorm(n,2)

#to make it easier
xs12=xs1*xs2

par(mfrow=c(1,3))
plot(xs1,ys,ylab="Length (mm)",xlab="Temperature")
```

```
plot(xs2,ys,ylab="Length (mm)",xlab="pH")
plot(xs12,ys,ylab="Length (mm)",xlab="Temperature * pH")
```



We can fit different models to such data, namely those that consider just the `xs1` (temperature), just the `xs2` (pH), or even just a new variable called `xs1xs2` (temperature $\times$ pH), and those with both variables and the interaction. Note `xs1xs2` variable is just the product of the other two, and in fact that fitting that product alone is equivalent to fitting the interaction alone!

```
#-----------------------------------
#models with interaction
#model with interaction
mx1x2I=lm(ys~xs1+xs2+xs1:xs2)
#just the interaction term
mI=lm(ys~xs1:xs2)
#same as mx1x2I
mx1x2I.b=lm(ys~xs1*xs2)
#same as just the interaction term
mI.b=lm(ys~xs12)
#-----------------------------------
#models without the interaction term
mx1x2=lm(ys~xs1+xs2)
mx1=lm(ys~xs1)
mx2=lm(ys~xs2)
```

To begin with, check what we commented above is true. Models `mx1x2I` and

mx1x2I.b are equivalent,

```
summary(mx1x2I)
```

```
##
## Call:
## lm(formula = ys ~ xs1 + xs2 + xs1:xs2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7590 -0.6134  0.1005  0.6850  2.5857
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 23.21760    1.22479  18.956  < 2e-16 ***
## xs1          1.97682    0.02030  97.368  < 2e-16 ***
## xs2         -5.06833    1.09787  -4.617 1.21e-05 ***
## xs1:xs2      3.01992    0.01876 160.986  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9462 on 96 degrees of freedom
## Multiple R-squared:  0.9999,Adjusted R-squared:  0.9999
## F-statistic: 3.272e+05 on 3 and 96 DF,  p-value: < 2.2e-16
```

```
summary(mx1x2I.b)
```

```
##
## Call:
## lm(formula = ys ~ xs1 * xs2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7590 -0.6134  0.1005  0.6850  2.5857
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 23.21760    1.22479  18.956  < 2e-16 ***
## xs1          1.97682    0.02030  97.368  < 2e-16 ***
## xs2         -5.06833    1.09787  -4.617 1.21e-05 ***
## xs1:xs2      3.01992    0.01876 160.986  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9462 on 96 degrees of freedom
## Multiple R-squared:  0.9999,Adjusted R-squared:  0.9999
## F-statistic: 3.272e+05 on 3 and 96 DF,  p-value: < 2.2e-16
```

and `mI` and `mI.b` are equivalent.

```
summary(mI)
```

```
##
## Call:
## lm(formula = ys ~ xs1:xs2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -66.780 -19.984   1.668  19.739  57.329
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  82.7735     7.4909   11.05   <2e-16 ***
## xs1:xs2       3.9363     0.1196   32.90   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27.28 on 98 degrees of freedom
## Multiple R-squared:  0.917,Adjusted R-squared:  0.9161
## F-statistic:  1083 on 1 and 98 DF,  p-value: < 2.2e-16
```

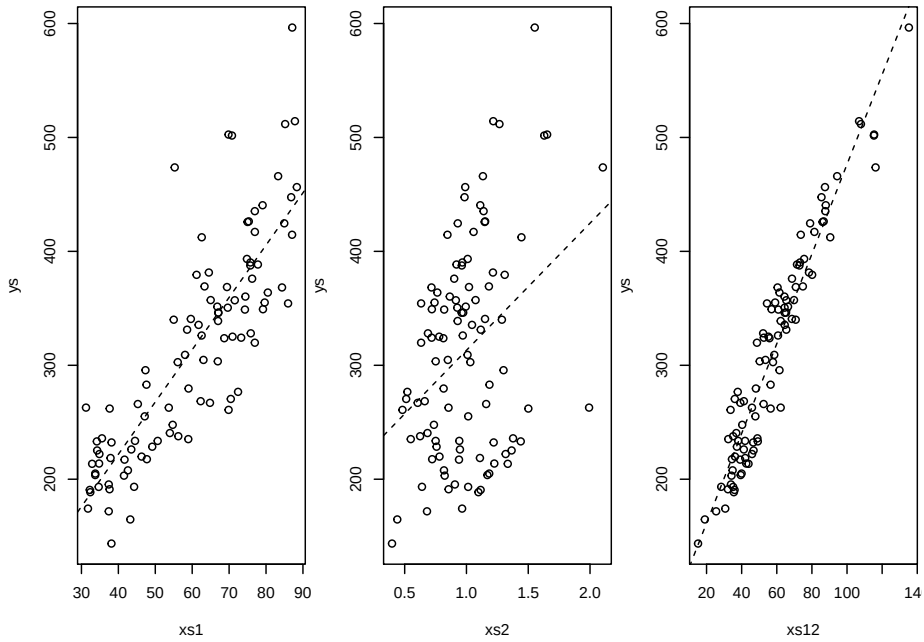
```
summary(mI.b)
```

```
##
## Call:
## lm(formula = ys ~ xs12)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -66.780 -19.984   1.668  19.739  57.329
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  82.7735     7.4909   11.05   <2e-16 ***
## xs12          3.9363     0.1196   32.90   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27.28 on 98 degrees of freedom
## Multiple R-squared:  0.917,Adjusted R-squared:  0.9161
## F-statistic:  1083 on 1 and 98 DF,  p-value: < 2.2e-16
```

Now, we plot the single variable models

```
#ploting the data and (single variable) models
par(mfrow=c(1,3),mar=c(4,4,0.2,0.2))
```

```
plot(xs1,ys)
abline(mx1,lty=2)
plot(xs2,ys)
abline(mx2,lty=2)
plot(xs12,ys)
abline(lm(mI),lty=2)
```



Note that if we ignore the interaction, we make the wrong conclusion, we conclude that `xs2` has the wrong effect compared to reality: it seems to have a positive influence, when we know that influence is negative!

```
summary(mx1x2)
```

```
##
## Call:
## lm(formula = ys ~ xs1 + xs2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -67.854  -6.369   0.761   7.561  52.010
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -156.08567    8.34374  -18.71   <2e-16 ***
## xs1            5.11738    0.09208   55.57   <2e-16 ***
```

```
## xs2          164.10708    5.20386   31.54   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.49 on 97 degrees of freedom
## Multiple R-squared:  0.9735,Adjusted R-squared:  0.973
## F-statistic:  1782 on 2 and 97 DF,  p-value: < 2.2e-16
```

When we look at the model that really makes sense here, given the true model, we get the right decisions for all the terms in the model: xs1 has a positive effect and xs2 has a negative effect in the response, with a significant positive effect on the interation between the two.

```
summary(mx1x2I)
```

```
##
## Call:
## lm(formula = ys ~ xs1 + xs2 + xs1:xs2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7590 -0.6134  0.1005  0.6850  2.5857
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 23.21760    1.22479  18.956  < 2e-16 ***
## xs1          1.97682    0.02030  97.368  < 2e-16 ***
## xs2         -5.06833    1.09787  -4.617 1.21e-05 ***
## xs1:xs2      3.01992    0.01876 160.986  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9462 on 96 degrees of freedom
## Multiple R-squared:  0.9999,Adjusted R-squared:  0.9999
## F-statistic: 3.272e+05 on 3 and 96 DF,  p-value: < 2.2e-16
```

Failing to include significant interactions could lead one to wrongly conclude that a variable that in reality has a negative effect on the response happens to have a significant positive impact on the response!

Note that, reassuringly, this would be the model preferred by AIC, by a long shot. All is good when all ends well, which if we have lots of data and low error, is not unexpected. But the real danger lies in real life data, where we do not know what reality is.

```
#compare models using say AIC
AIC(mx1,mx2,mI,mx1x2,mx1x2I)
```

```
##        df       AIC
```

```
## mx1      3 1076.8927
## mx2      3 1183.9971
## mI       3  949.0211
## mx1x2    4  836.8333
## mx1x2I   5  278.6349
```

Therefore, exploring important interactions is important, and failing to include relevant ones might cause errors (while including spurious ones might also mask some real effects, see next slides!).

This was a 2-way interaction.

One can think about 3-way interactions or even higher order interactions, but no one can interpret those models anymore!

13.1 Larger order interactions

Just for fun (yeah.. fun !) we look at a model where we have more variables available to fit. In fact, we have 4 covariates, meaning we could even fit a 4th order interaction. Don't try that at hope folks, only trained professionals should (and to so so with a 4th order interaction, probably trained profesionals that went a bit crazy at some point!).

Our true model wll be of the form

$ys = 20 + 2 * xs1 - 4 * xs2 + 3 * xs1 * xs2 + xs3 + xs4$

```
#-------------------------------------------
# A 4 way interaction model
#but in reality there is only 1 second order interaction
#-------------------------------------------
set.seed(123)
#get a response variable
xs1=runif(n,30,90)
#get a second variable
xs2=rgamma(n,10,10)
#get a response variable
xs3=runif(n,3,6)
#get a second variable
xs4=rgamma(n,4,4)
#define the linear predictor
ys=20+2*xs1-4*xs2+3*xs1*xs2+xs3+xs4+rnorm(n,2)
```

After having created the data, we can fit a couple of models to it. One with all the interactions, one with just the correct (=true, since we know the data generating model) 2 way interaction between `xs1` and `xs2`.

```
modW=lm(ys~xs1*xs2*xs3*xs4)
modR=lm(ys~xs1+xs2+xs3+xs4+xs1:xs2)
```

If we look at the 4 way interaction model

```
summary(modW)
```

```
##
## Call:
## lm(formula = ys ~ xs1 * xs2 * xs3 * xs4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.38930 -0.54310  0.05557  0.65213  3.01609
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      1.687e+01  2.363e+01   0.714    0.477
## xs1              1.897e+00  3.753e-01   5.056 2.47e-06 ***
## xs2              6.907e+00  2.383e+01   0.290    0.773
## xs3              2.413e+00  5.412e+00   0.446    0.657
## xs4              2.543e+01  2.372e+01   1.072    0.287
## xs1:xs2          3.024e+00  3.697e-01   8.180 2.64e-12 ***
## xs1:xs3          1.438e-02  8.635e-02   0.167    0.868
## xs2:xs3         -2.400e+00  5.403e+00  -0.444    0.658
## xs1:xs4         -1.958e-01  3.457e-01  -0.567    0.573
## xs2:xs4         -3.026e+01  2.485e+01  -1.218    0.227
## xs3:xs4         -5.713e+00  5.582e+00  -1.024    0.309
## xs1:xs2:xs3     -6.552e-04  8.407e-02  -0.008    0.994
## xs1:xs2:xs4      2.623e-01  3.558e-01   0.737    0.463
## xs1:xs3:xs4      4.987e-02  8.132e-02   0.613    0.541
## xs2:xs3:xs4      6.737e+00  5.736e+00   1.175    0.243
## xs1:xs2:xs3:xs4 -6.164e-02  8.242e-02  -0.748    0.457
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.053 on 84 degrees of freedom
## Multiple R-squared:  0.9999,Adjusted R-squared:  0.9999
## F-statistic: 6.03e+04 on 15 and 84 DF,  p-value: < 2.2e-16
```

we see several errors, namely type II errors associated with `xs2`, `xs3` and `xs4`! In other words, including interactions which are not real can mask the true influence of relevant variables!

When we look at the right model, these go away, and all variables and the correct second order variable are considered significant (at the usual significance levels)

```
summary(modR)
```

```
##
## Call:
```

```
## lm(formula = ys ~ xs1 + xs2 + xs3 + xs4 + xs1:xs2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6674 -0.5841  0.1184  0.6866  2.7579
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 21.15925    1.46383  14.455  < 2e-16 ***
## xs1          1.99914    0.02291  87.247  < 2e-16 ***
## xs2         -3.30066    1.29741  -2.544  0.01259 *
## xs3          1.19397    0.12449   9.591 1.36e-15 ***
## xs4          0.76218    0.24032   3.171  0.00205 **
## xs1:xs2      2.99210    0.02131 140.432  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.054 on 94 degrees of freedom
## Multiple R-squared:  0.9999,Adjusted R-squared:  0.9999
## F-statistic: 1.806e+05 on 5 and 94 DF,  p-value: < 2.2e-16
```

Reassuringly, the right model is preferred in terms of AIC

```
AIC(modW,modR)
```

```
##      df      AIC
## modW 17 310.6238
## modR  7 302.0554
```

What would happen if we had not recorded some of the variables? Or if we considered different second order or even 3 order interactions? I experiment below just as an example creating additional 10 wrong models.

```
modW1=lm(ys~xs1+xs2*xs3)
modW2=lm(ys~xs2+xs3*xs4)
modW3=lm(ys~xs1*xs2*xs3)
modW4=lm(ys~xs1*xs3*xs4)
modW5=lm(ys~xs1+xs2+xs3)
modW6=lm(ys~xs1+xs3+xs4)
modW7=lm(ys~xs2+xs3+xs4)
modW8=lm(ys~xs1*xs2)
modW9=lm(ys~xs1*xs3)
modW10=lm(ys~xs1*xs4)
```

```
library(knitr)
kable(AIC(modW,modW1,modW2,modW3,modW4,modW5,modW6,modW7,modW8,modW9,modW10,modR))
```

	df	AIC
modW	17	310.6238
modW1	6	835.1165
modW2	6	1180.6098
modW3	9	311.4269
modW4	9	1086.9135
modW5	5	833.2009
modW6	5	1089.0771
modW7	5	1180.0438
modW8	5	368.2706
modW9	5	1088.0494
modW10	5	1089.2458
modR	7	302.0554

If we had all variables, we would always get the correct model apparently (even if I did not a full search over the possible model space - a task for you !) but if some of the variables were not recorded, some errors might occur. Your task is to find them, just for fun… have fun!

Chapter 14

Conlusion on linear regression

14.1 Conclusion

The material in previous 4 lectures allows you to fully understand the outputs of simple regression models, and to see how some statistical models that you know by other names, like an ANOVA or a the t-test, are just simple special cases of a linear model.

It also helps you understand how the parameter values represent just features of the data and its generating process, and how we can recover estimates of the original relationships between the variables from said set of parameters.

I recommend you explore the code and output above, and that in particular you experiment with changing means (parameter values for the real models), variances (the precision of how you would measure variables) and sample sizes (which gives you an indication of how much information you have to estimate the underlying reality). Understanding the outputs under these new scenarios is fundamental for progressing towards more complex regression models, like GLMs or GAMs, of which the above cases are just particular cases.

Many additional interesting links on linear models exist online. This is just one of them: https://data-flair.training/blogs/r-linear-regression-tutorial/

Chapter 15

Class 13: 10 11 2020 Maximum likelihood and all that

We have been fitting regression models, using say function `lm`. While this might seem rather ordinary and uninteresting to a seasoned statistician, it is quite remarkable for the average person. So remarkable that I would suggest the analogy of a cell phone to most of us. We do not really think about it, since we do it all the time, but wait a second: can you imagine all the things that must happen inside that little device so that your cousin Maria João having her honey moon in Hawaii can share with you a 2-second delay live of the fantastic romantic dinner she is having, while you are actually 10000 meters above ground on a plane preparing to land in Siberia? For about 99.9 % of you, you do not! And I do not plan on telling you here - I hope by now you have realized that is beyond the purpose of this book. On the contrary, telling you exactly what happens behind the scenes when a function like `lm` reports some maximum likelihood estimates of a given model parameters is the task that lays ahead. "Brace brace!", as they might say when facing strong turbulence on your plane that currently is landing in Siberia.

What `lm` does under the hood is, based on a model, estimate the best value of the parameters, given the data. We illustrate it using the standard linear model, with a single covariate x to explain the response y

$$y = \alpha + \beta x + e_i$$

where e_i $Gau(0, \sigma^2)$. The `lm` function finds the best values of α, β and σ, given the data. Those we call estimators, and denote them by $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\sigma}$.

After collecting a sample, we fit the model and we get the estimates. Remember estimates are observed values, or realizations based on the sample, of estimators.

The way `lm` finds the estimates is via maximum likelihood. Actually, this happens despite the fact that the line is widely know as the minimum squares line. Why is that? Because as we saw in chapter ?, that line is the line that minimizes the sum of the squares of the deviations between the observations and the predictions conditional on the best line. Formally, that is the line that minimizes the following quantity

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - (\hat{\alpha} + \hat{\beta}x_i))^2.$$

But what is the likelihood and does it work? We will introduce the concept using an example. Imagine a biologist, lets call her Andreia. Andreia is interested in estimating the probability that a pair of jays will produce eggs before the first of June.

She sets out to find a random sample of blue jay nests, and defines a random variable X representing the egg status of a nest on the 1^{st} of June. We assume that all eggs laid before 1 June will not have fledged yet). Andreia decides that X will take:

- the value 1 if eggs are present, which she will call a success, and she assumes that happens with probability θ,
- the value 0, which she calls a failure, representing no eggs present, with probability 1-θ.

Assuming that the probability of different nests having eggs is independent, each of these is a Bernoulli trial, and there are N trials, of which we could say n will be successes, and $n - n$ will be failures. The Bernoulli distribution is a special case of a Binomial random variable, with a single trial and probability of success θ. In fact, you you consider all the nests together that is indeed a Binomial with parameters N and p. A small detour to justify this statement: there is a theoretical result that demonstrates that the sum of K independent Binomials X_k, each with N_k trials, with constant probability of success p, is a Binomial(N, p), where $N = \sum_{k=1}^{K} N_k$. Therefore, the sum of k Bernoulli trials, i.e. k Binomial(1,p) independent random variables, is a Binomial(K, p).

So this is a model with a single parameter, θ. (since we know N, the number of trials!)

Andreia goes out and about in the field and finds 5 nests. The first has eggs, the second and third do not, the forth does, and the fifth does not. By this time Andreia is tired and decides to call it a day, with her sample x collected: $x = c(1, 0, 0, 1, 0)$.

```
#a 1 is a nest with eggs, a 0 is a nest without eggs
nests=c(1,0,0,1,0)
```

Note that in this case the number of successes n is 2 and the number of failures `N-n` is 3.

Then she asks a friend doing an MSc in biostatistics how she can estimate the value of θ. Unfortunately, her friend has just started her classes, and she too is a bit unsure about what to do too. But she does know how to calculate the probability of the observed sample.

$$P(x|\theta) = \theta(1-\theta)(1-\theta)\theta(1-\theta) = \theta^2(1-\theta)^3$$

If only we knew what the value of was we could evaluate this probability. Imagine that it was 0.2, then the probability of the sample would be $0.2^20.8^3$=0.02048. What if it was 0.8, then the probability of the data would be $0.8^20.2^3$=0.00512. This is a considerably lower probability.

And here's when Andreia's friend has a great idea. What if we turn it around and look at this as a function of theta, conditional on the data

$$P(\theta|x) = \theta(1-\theta)(1-\theta)\theta(1-\theta) = \theta^2(1-\theta)^3$$

Then we could evaluate the expression for a set of possible values for θ, and the largest probability will intuitively correspond to the most likely value of θ.

```
library(knitr)
thetas<-seq(0.05,0.95,by=0.1)
pthetas<-thetas^2*(1-thetas)^3
kable(cbind(thetas,pthetas),col.names = c("theta","P(theta)"))
```

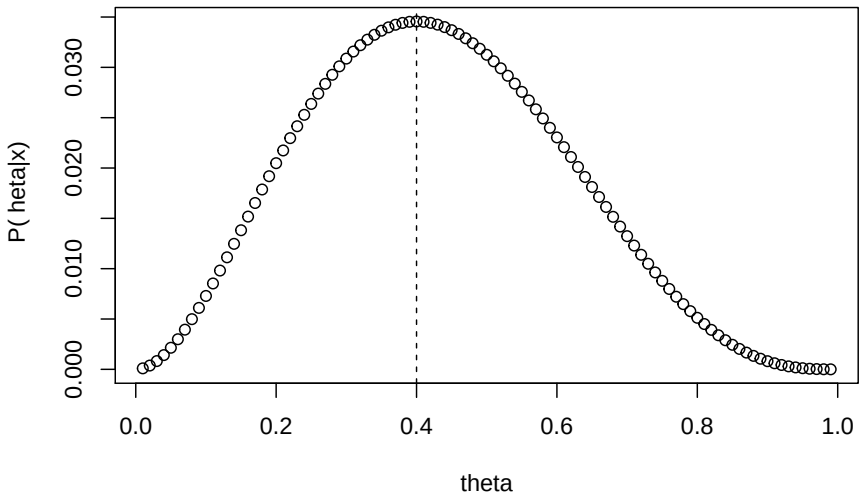
theta	P(theta)
0.05	0.0021434
0.15	0.0138178
0.25	0.0263672
0.35	0.0336416
0.45	0.0336909
0.55	0.0275653
0.65	0.0181147
0.75	0.0087891
0.85	0.0024384
0.95	0.0001128

The largest values are observed for the trial values of θ of 0.35 and 0.45. What if we calculate that probability for a fine grid of values possible for θ and represent it in a plot. This is what follows, and we add to the plot a dashed vertical line representing the value of θ for which that function is maximized.

```
thetas <- seq(0.01,0.99,by=0.01)
pthetas<-thetas^2*(1-thetas)^3
plot(thetas,pthetas,ylab="P(\theta|x)",xlab="theta")
```

```
## Warning in title(...): font width unknown for character 0x9
```

```
abline(v=thetas[pthetas==max(pthetas)],lty=2)
```



You will probably not be too surprised to find out that $\hat{\theta}$=0.4 is indeed the maximum likelihood estimate (MLEe), and that for a proportion $\hat{\theta}$=successes/trials is the maximum likelihood estimate (MLE). Therefore, note that it will only be from the context that one can say if MLE stands for an estimate, which corresponds to a random realization of the estimator, or for the estimator itself. This should not have come as a surprise. Remember we had 2 successes in 5 trials, and that corresponds to 0.4.

Andreia asks her friend what was the point. If the MLE was just the empirical proportion, 2/5=0.4, why going through all this trouble? There's at least 3 good reasons for that:

- This way we understand why an MLE is
- If you look at the figure above, we not only have an estimate of the parameter θ, but we also have an idea about the precision around that estimate. That comes from the shape of the likelihood profile. We get back to this below.
- by embedding it in the concept of a likelihood, we open the door to gener-

alize this procedure to any other far more complicated situation for which closed form analytic estimators do not exist. As examples of additional sophistication, we could easily:

- consider several parameters at once; as an example, we could be considering instead of a Bernoulli a complex model that describes how a whale dives, with 17 parameters that we want to maximize at once. Rarely closed form estimators will be available then;
- make the parameters a function of observed covariates. In the case of our nests, the height of the nest could be a relevant covariate to model the probability of success of a nest, say. In such a case, we could have an estimate for θ that would be dependent of the height `h`, e.g by defining that $\theta_h = f(h)$. Naturally we would choose the link function f such that theta would be constrained to be between 0 and 1, the possible values for a probability. The logit link function comes to mind here. But that will be a story for another day.

To illustrate the point above regarding being able to estimate the precision around the parameter estimate from the likelihood function, lets consider that we had not 5 samples, but man more. In the figure below we contrast the small sample size to a set of increasing sample sizes: 50, 100 or 200.

```
par(mfrow=c(4,1),mar=c(4,4,0.5,0.5))
thetas <- seq(0.01,0.99,by=0.01)
pthetas<-thetas^2*(1-thetas)^3
plot(thetas,pthetas,ylab="P(\theta|x)",xlab="theta (n=5)")
```

```
## Warning in title(...): font width unknown for character 0x9
```

```
abline(v=thetas[pthetas==max(pthetas)],lty=2)
thetas <- seq(0.01,0.99,by=0.01)
pthetas<-thetas^20*(1-thetas)^30
plot(thetas,pthetas,ylab="P(\theta|x)",xlab="theta  (n=50)")
```

```
## Warning in title(...): font width unknown for character 0x9
```

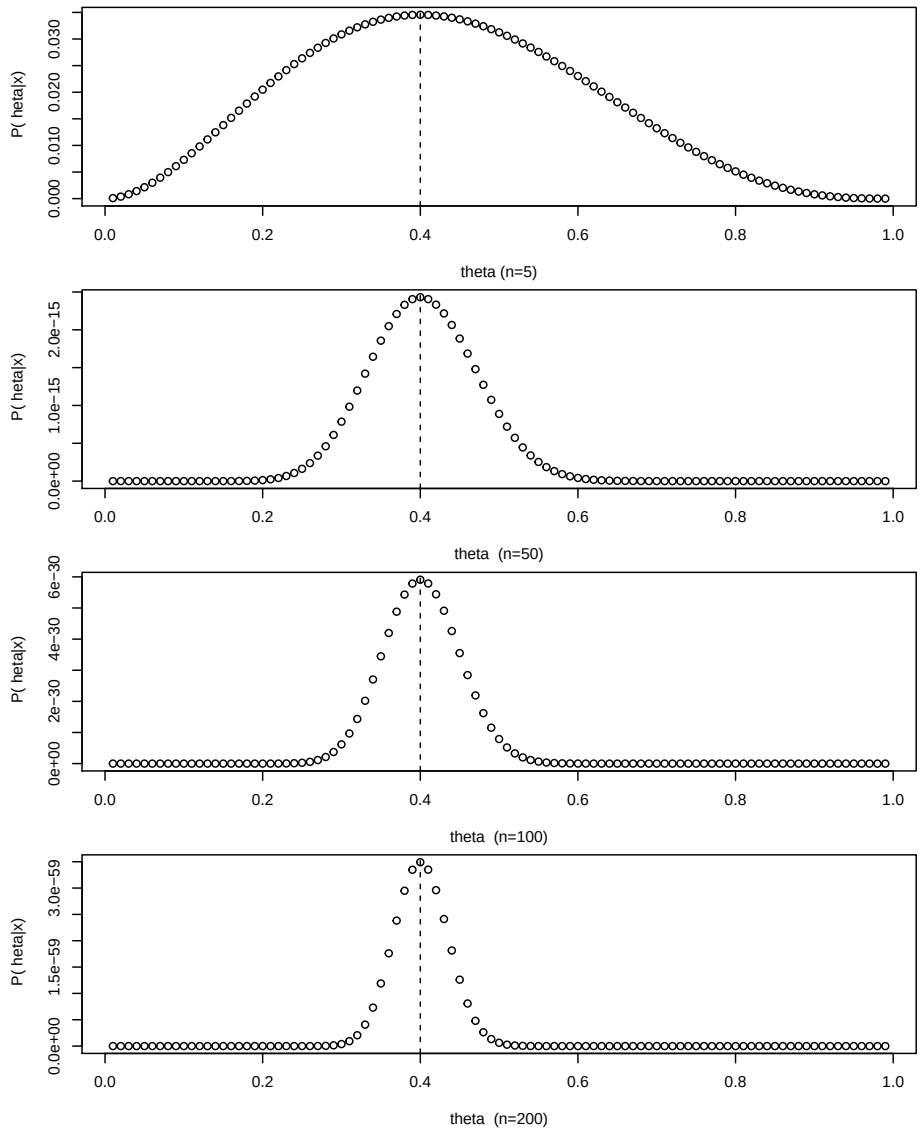
```
abline(v=thetas[pthetas==max(pthetas)],lty=2)
thetas <- seq(0.01,0.99,by=0.01)
pthetas<-thetas^40*(1-thetas)^60
plot(thetas,pthetas,ylab="P(\theta|x)",xlab="theta  (n=100)")
```

```
## Warning in title(...): font width unknown for character 0x9
```

```
abline(v=thetas[pthetas==max(pthetas)],lty=2)
thetas <- seq(0.01,0.99,by=0.01)
pthetas<-thetas^80*(1-thetas)^120
plot(thetas,pthetas,ylab="P(\theta|x)",xlab="theta  (n=200)")
```

```
## Warning in title(...): font width unknown for character 0x9
```

```
abline(v=thetas[pthetas==max(pthetas)],lty=2)
```



As we increase the sample size, and hence we increase the amount of information available to estimate θ, the likelihood profile becomes more spiky. It can be demonstrated that the curvature of the likelihood profile allows us to quantify the precision on our estimate of the parameter. Naturally, the steeper the curve, the better, in the sense that the more certain we are. On the other hand, when the likelihood surface is very flat, we might hit problems in terms of the numerical maximization of the likelihood.

15.1 Maximizing a likelihood algebraically

While above we were able to maximize the likelihood function via a "grid" search. We divided the possible range of values that the parameter could take, also know as the parameter space, into a large number of candidate values. Then we evaluated the likelihood at each one of these, and picked the value of the parameter for which the function was maximum: the maximum likelihood estimate.

Grid search can become very inefficient very fast, and hence there are other ways to maximize a likelihood. One is to analytically find what is the maximum of that function. How can we do that. Straightforwardly for our example. You differentiate the function, find the point at which the first derivative is 0, and by definition that point is a maximum or a minimum. If you are unsure the second derivative would tell you which. Considering the above

$$\frac{d(f(\theta))}{d\theta} = \frac{d(\theta^n(1-\theta)^{N-n})}{d\theta}$$

Then by solving

$$\frac{d(f(\theta))}{d\theta} = 0$$

we get that $\hat{\theta} = n/N$, which is just the empirical proportion (i.e. the observed proportion of successes in the sample).

(note to self: add detail to these derivations above)

However, like the grid search, this is not a problem free procedure. The above expression was simple enough that derivation was trivial. That might not be the rule, but the exception, so we need an alternative approach for when models are more complex than our Bernoulli example. That will be the norm in real ecological models.

15.2 Numerically Maximizing a likelihood

Here we look at using a numerical maximization procedure, which means that we will derive a procedure, and algorithm, that will find the maximum of a function computationally. The analogy with the real world is simple. Imagine that you were somewhere in the most boring country in the world, Boredomnesia. It happens to be a square with a single mountain at the center, as depicted in the image below, and you wanted to start walking and reaching the highest point in the country. Boredomnesia happens to also be the foggiest country in the world, so you manage to see about 3 meters around you, at most!

```
# need mvtnorm package
library("mvtnorm")
range = seq(-5,5,0.1)
mean = c(0,0)
Sigma = matrix(c(1, .5, .5, 1), 2)
out = matrix (rep(0,101*101),101)

for (i in 1:length(range)){
        for (j in 1:length(range)){
                out[i,j] = dmvnorm(c(range[i],range[j]),mean=mean,sigma=Sigma)
        }
}
 persp(out,theta = 30,phi = 20,col="lightblue",xlab = "Latitude",ylab="Longitude",zlab=
```

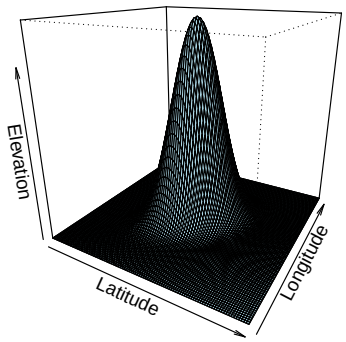


Figure 15.1: Ilustrating a likelihood. This would be Boredomnesia, the most boring country in the world. You want to to find a way to reach its highest point amidst the dense fog that characterizes it. How can you do it? Just keep moving up!

```
#add an emoji that is a small man to it!
```

But you actually have no idea about the orography of the country, and you can only see so much as the country is always a bit foggy. To make sure you manage, you could set up a set of rules for yourself:

1. starting where you are (this seems like a lame first step, but you will see surprisingly that is actually one of the hardest for a computer!). Then, until you can't find a higher point, repeat the following steps:

2. evaluate the height where you are currently
3. evaluate the height at 8 directions around you (like North, South, East, West and the 4 intermediate directions, say)
4. move toward the steepest highest of those directions
5. if the difference in elevation (i.e. the mountain slope) is
   - high: move 3 meters
   - low: move 1 meter

If you do these steps above, when you stop you are at most 1 meter from the top. Well done, you are the king of the world.

Naturally, this assumes the terrain of the country you are in is relatively simple. More precisely, that there is only one mountain in the country, and there are no valleys (or in a likelihood world, no local maxima). Basically, you would not like to be Dane, or Dutch, as there are no mountains there to begin with, and definitely you would not want to be near the Grand Canyon (Figure X) or in Scotland (Figure X), where the Munro's would certainly defeat you. As we will see below, this has very important implications in the likelihood world!

Figure 15.2: The nightmare place for our example task of finding the highest place using the move-towards-higher-ground algorithm, given all the plateaus

Now… what happens inside a computer? The above example makes more sense if we are maximizing a likelihood with respect to two parameters, so that the likelihood surface is a bi-dimensional surface. Imagine a Gaussian, for which we want to estimate the mean μ and the standard deviation σ. In the real world the analogy is latitude is equivalent to μ, longitude is equivalent to σ, and the altitude is the likelihood. So now we look at how a computer does it!

There are many algorithms one could use, and here we will use some standard R functions to do the job for us. We will consider a couple, `optimize`, when we are only considering a single parameter, and `optim`, for when more than one parameter is at stake. An alternative to `optim` might be `nlm` (from package `stats`). There are many other options in and outside R!

The first thing we need to do is to write up the likelihood function. This will be

Figure 15.3: The nightmare place for our example task of finding the highest place using the move-towards-higher-ground algorithm, given all the local maxima

often the hardest part. That would be like having a detailed map of the country. We know that takes a lot of work to do.

This must be a function which the first argument is the parameter(s), typically the second is the data. Then other additional parameters might follow, or not.

Let us build, step by step, a likelihood for the example of the Bernoulli case for the nests we were looking at in the previous section.

Recall the probability of θ, given the data 1,0,1,0,0.

$$P(\theta|x) = \theta(1-\theta)(1-\theta)\theta(1-\theta) = \theta^2(1-\theta)^3$$

We can write a bespoke function of theta to evaluate this probability

```
liktheta1=function(theta){
  lik<-theta^2*(1-theta)^3
  return(lik)
}
```

Now we use it, job done

```
liktheta1(0.35)
```

```
## [1] 0.03364156
```

just as in the table above, we are good. But this is not really what we want, because the data, our sample was hardwired, we need a function that could cope

with any sample. We need to be able to compute the relevant statistics from the sample. Andreia realizes that she can do that easily by summing successes and failures in the table

```
sum(nests==1)
```

```
## [1] 2
```

```
sum(nests==0)
```

```
## [1] 3
```

and hence she suggests this new formulation

```
liktheta2=function(theta,samp){
  lik<-theta^sum(samp==1)*(1-theta)^sum(samp==0)
  return(lik)
}
```

She tries it out

```
liktheta2(0.35,nests)
```

```
## [1] 0.03364156
```

and she gets the same value as above. She's happy, as she can now calculate the likelihood for (1) any parameter value and (2) any sample. Excited, she shows how this would be the case for 11 nests, with just 1 success.

```
liktheta2(0.35,c(1,rep(0,10)))
```

```
## [1] 0.00471196
```

Now, she's really excited and she has a dream where she samples 1000 eggs, and 300 successes. She wakes up and wants to know the likelihood under that scenario

```
dreameggs<-c(rep(1,300),rep(0,700))
liktheta2(0.35,dreameggs)
```

```
## [1] 1.818708e-268
```

Ups, something went terribly wrong, the likelihood is now… 0. This is unhelpful, I can't climb a mountain if there is no mountain! She scratches her head for a while and she realizes what is going on. She's multiplying 5000 probabilities, even if those were high, the computer will round them to 0.

Andreia calls a friend, and he says that he will give her two clues that might help. And then says:

1. If you apply the log to a function, the logged function will have the same function as the untransformed function, and
2. log(a*b)=log(a+b)

Andreia hangs up the phone and takes a mental note: "I need to find better, more useful friends"! But during the night she has an epiphany. If she logs the function, a product of probabilities, she will get a sum of log probabilities. Log probabilities are smaller than probabilities, but there's a small miracle in the process. The sum of small numbers does not tend (does not converge to 0!). And so she tries a new function, where she adds the log probabilities

```
logliktheta=function(theta,data){
  loglik=sum(log(theta^sum(data==1))+sum(log((1-theta)^sum(data==0))))
  return(loglik)
}
```

She calculates the function that gave her grief above, and the egg dream meets the epiphany

```
logliktheta(0.35,dreameggs)
```

```
## [1] -616.4947
```

Then, she just needs to call the `optimize`, where `interval` defines the plausible parameter space, and we make sure that `maximum` is TRUE because by default the function `optimize` minimizes (That is why we sometimes use a function that is `-log(likelihood)`, that means the minimum is the point we want!) the function `f` with respect to its first parameter, given any other arguments provided to `f`. In this case those other parameters are just the `data`, the second argument for `liktheta`. Those you will recognize as our data.

```
MLEtheta<-optimize(f=logliktheta,interval=c(0.01,0.99),data=c(0,1,0,1,0),maximum=TRUE)
MLEtheta
```

```
## $maximum
## [1] 0.399996
##
## $objective
## [1] -3.365058
```

Now we can actually calculate the MLE for θ in the case of Andreia's dream sample.

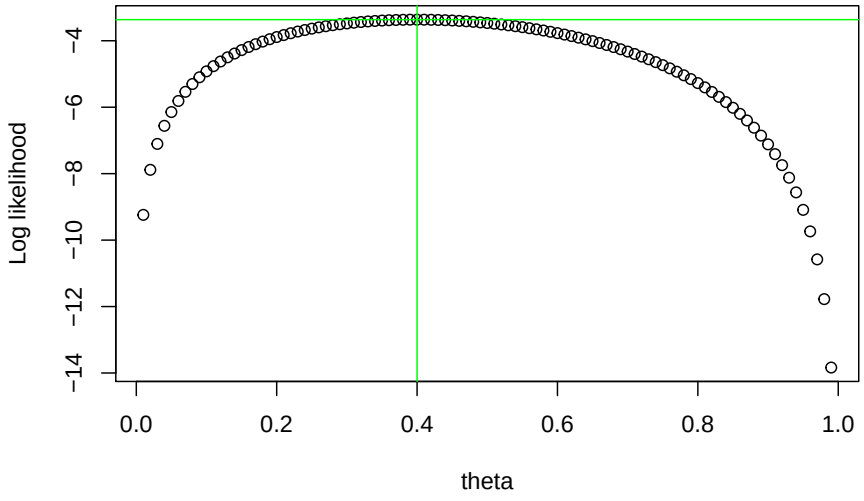
```
MLEthetadream<-optimize(f=logliktheta,interval=c(0.01,0.99),data=dreameggs,maximum=TRU
MLEthetadream
```

```
## $maximum
## [1] 0.2999964
##
## $objective
## [1] -610.8643
```

So the output of `optimize` has two components, *maximum* and *objective*. What are these? The two components of this object are the MLE of the parameter,

in this case 0.399996 and the value of the function at that point for θ, in this case -3.3650583. This will be the actual value of the likelihood at this point and might be useful later, but for now we ignore it. Note that 0.399996 is just a numeric approximation of the real value, that we know analytically to be 0.4. These are illustrated below:

```
#valores possiveis para thetas
thetas<-seq(0.01,0.99,by=0.01)
#object to hold the values of the likelihood
nthetas <- length(thetas)
logliksthetas<-numeric(nthetas)
#para cada theta
for(i in 1:nthetas){
  logliksthetas[i] <- logliktheta(thetas[i],nests)
}
plot(x=thetas,y=logliksthetas,ylab="Log likelihood",xlab="theta")
abline(v=MLEtheta$maximum,h=MLEtheta$objective,col="green")
```



As a task, what would it require to change the above code to calculate the value of theta if we had 78 trials and 43 sucesses? You got it, just need to change the data

```
MLEtheta<-optimize(f=logliktheta,interval=c(0.01,0.99),data=c(rep(1,43),rep(0,78-43)),maximum=TRU
MLEtheta
```

```
## $maximum
## [1] 0.5512824
```

```
##
## $objective
## [1] -53.6545
```

So, now we know all about likelihoods, but Andreia wonders. Why all the trouble, if I could just have calculated the exact value of the MLE as the observed proportion? To that we need to continue with Andreia's explorations.

15.3 The case of a Gaussian

Lets now look at situation where Andreia is interested in characterizing how far away from the nearest river, in a straight line, are the nests from water. She assumes that these might be hypothetically described by a Gaussian random variable. That will be the basis for constructing a likelihood. For her 5 nests, those distances in kilometers are

```
dists<-c(0.78,1.73,1.32,0.54,2.12)
```

Then she thinks about what might the likelihood look like for a Gaussian. She knows R can calculate the density of a Gaussian via `rnorm`, and so she suggests the following minus log likelihood function:

```
minuslogliknorm=function(pars,data){
  media=pars[1]
  desvio=pars[2]
  minusloglik=-sum(log(dnorm(data,mean=media,sd=desvio)))
  return(minusloglik)
}
```

She testes the function on simulated data, 10000 fake distances with mean 2 and standard deviation 2.7. Either a small miracle happened, or she got it right at first try:

```
#simulated sample size
n<-10000
# simulated mean
mG<-2
# simulated standard deviation
sdG<-0.7
# simulated sample
xs=rnorm(n,mean=mG,sd=sdG)
# MLE of the parameters
MLEGau<-optim(par=c(1,1),fn=minusloglik norm,data=xs)
MLEGau
```

```
## $par
## [1] 1.9994644 0.6987295
##
```

```
## $value
## [1] 10604.8
##
## $counts
## function gradient
##       65       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Note she now has more than 1 parameter, so `optimize` was not an option, and had to use function `optim`. This requires starting values via argument `par`, but the rest is similar to `optimize`, the `data` and the function to maximize is `fn`. The output is a bit messier, and while the resulting object components are all relevant to know about, we ignore them for now for simplicity. If we evaluate this likelihood using a brute force grid approach, this is what we get

```
#to implement later
```

Notice that this image reminds us of Boredomnesia! Now, we know how to do this for more than one parameter, but why would we. After all, if I wanted to estimate the MLE of a Gaussian, actually, the sample mean `mean(xs)`=1.999534 and the sample standard deviation `sd(xs)`=0.6987875 are what I want. So the above values obtained by `optim` for μ and σ, the MLEs, 1.9994644 and 1.9994644 respectively, are really just numerical approximations of the real analytically obtainable MLE's 1.999534 and 0.6987875, respectively (note: need, strictly, to refer to minor detail regarding denominator of the standard deviation, considering n, the MLE, or n-1, not MLE but unbiased; ME students can ignore detail for now!). These are themselves, in this case where we know reality, estimates of the true simulated values generating our data, 2 and 0.7, respectively. So all quite reasonable and close, really, what is not surprising given a sample size of 10^4!

Now, what about based on the distances Andreia had for the 5 nests

```
dists
```

```
## [1] 0.78 1.73 1.32 0.54 2.12
```

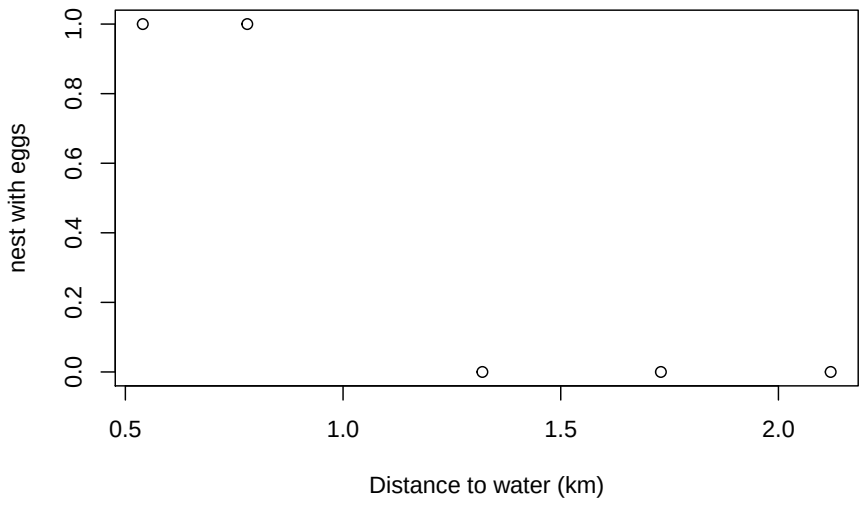
```
MLEGaud<-optim(par=c(1,1),fn=minuslogliknorm,data=dists)
MLEGaud
```

```
## $par
## [1] 1.2979771 0.5840272
##
## $value
```

```
## [1] 4.406008
##
## $counts
## function gradient
##       53       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Note that, as it might be interesting for later, that nests with eggs were closer to the water. But remember also that with such a small sample size, believing in that as being some indication of reality, rather than just a fluke, is a matter of faith.

```
plot(dists,nests,ylab="nest with eggs",xlab="Distance to water (km)")
```



We get some estimates, despite the fact that it is yet unclear why we should do it this way and not just use `mean` and `sd`. To see why, we continue our story, and the plot thickens…

15.4 The case of a linear model

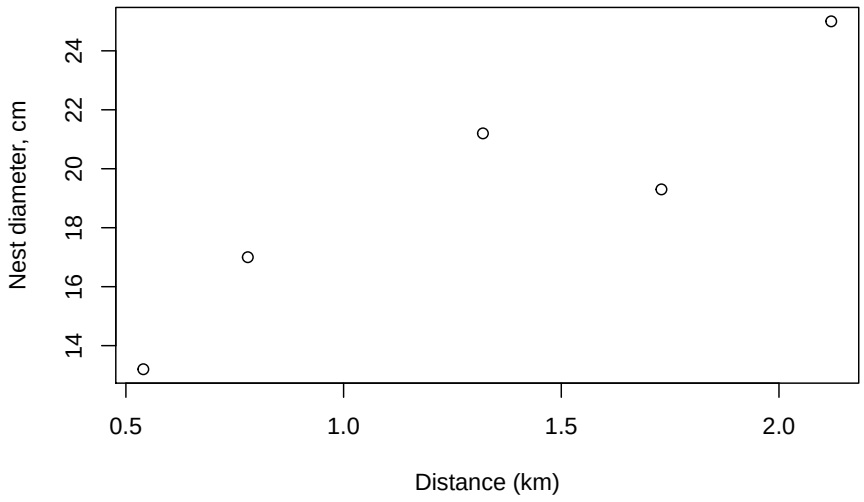
We started by talking about `lm`, so, what is happening behind `lm`.

Let's us imagine that for each nest, Andreia also related the size of the nest with the distance to the water. The size of the nest, diameter, in cm, was as below, She is interested in describing, modeling, explaining how nest size changes as a function of distance to the water.

```
size<-c(17,19.3,21.2,13.2,25)
```

We can visualize the relationship between the distance to the water and the nest diameter

```
plot(dists,size,xlab="Distance (km)",ylab="Nest diameter, cm")
```



So now Andreia needs a likelihood. Since the above relation seems linear, she remembers that the linear model is given by

$$y_i = a + bx_i + e_i$$

where the e_i are a Gaussian with mean 0 and constant variance σ^2. And then she has another epiphany and realizes that she can construct data for which a likelihood can be derived. Because if she rearranges the above, you have

$$e_i = y_i - (a + bx_i) = y_i - \hat{y}_i$$

where, remember, e_i are a Gaussian with mean 0 and constant variance σ^2. And so we can build a likelihood that exploits that Gaussian density for the observed errors, as

```
liklm=function(pars,data){
  #data must be a data.frame with columns y and x
  a=pars[1]
  b=pars[2]
  sigma=pars[3]
  ps=dnorm(data$y-(a+b*data$x),mean=0,sd=sigma)
  #minus loglik
  loglik=-sum(log(ps))
  return(loglik)
}
```

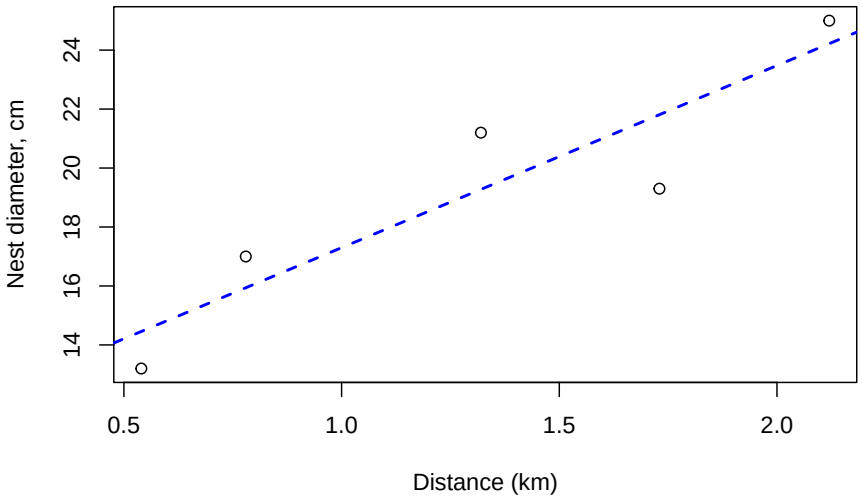
and we use it over our sample

```
lmMLE<-optim(par=c(2,1,1),fn=liklm,data=data.frame(y=size,x=dists))
lmMLE
```

```
## $par
## [1] 11.120181  6.178854  1.631731
##
## $value
## [1] 9.542754
##
## $counts
## function gradient
##      192       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

and so we get as estimates of a, b and σ of 11.1201813, 6.1788541 and 1.6317308, respectively. So finally, Andreia uses her estimated values for a and b and puts them over the above plot of the data

```
plot(dists,size,xlab="Distance (km)",ylab="Nest diameter, cm")
abline(lmMLE$par[1],lmMLE$par[2],lty=2,lwd=2,col="blue")
```



Now, we know that `lm` does this kind of stuff very efficiently, so how do these compare across? We can look at the outcome of the `lm` call

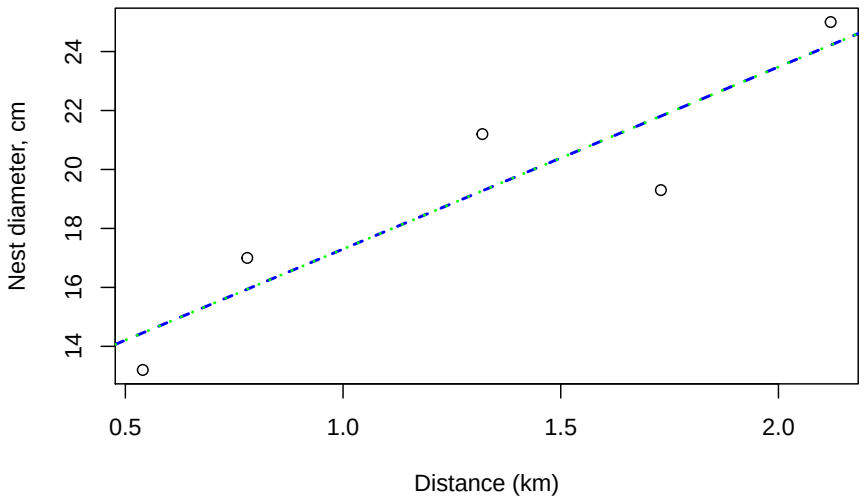
```
lm0<-lm(size~dists)
summary(lm0)
```

```
##
## Call:
## lm(formula = size ~ dists)
##
## Residuals:
##       1       2       3       4       5
##  1.0616 -2.5101  1.9240 -1.2550  0.7794
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   11.117      2.296   4.843   0.0168 *
## dists          6.181      1.613   3.832   0.0313 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.106 on 3 degrees of freedom
## Multiple R-squared:  0.8304,Adjusted R-squared:  0.7738
## F-statistic: 14.68 on 1 and 3 DF,  p-value: 0.03132
```

and when we add these to the above plot, we see we were bang on: the two lines

are indistinguishable by eye.

```
plot(dists,size,xlab="Distance (km)",ylab="Nest diameter, cm")
abline(lmMLE$par[1],lmMLE$par[2],lty=2,lwd=2,col="blue")
abline(lm0,lty=3,lwd=2,col="green")
```



So, but still, why would we do it this way, since `lm` does it with less hassle, faster, and probably better? For a number of reasons, including:

1. because it allows us a framework that is generalizable to any model for which we can define the likelihood, so it works for more than standard regression models
2. because it allows us to really understand what is happening in the background, as an example, we can relate the profile of the likelihood to the variance around the parameter estimates

So to complete this chapter, lets see an example for which a dedicated function like `lm` is not available off the shelf, and we would really need to write down our own likelihood to get meaningful ecological inferences.

15.5 The really interesting case

Imagine now that we were interested in relating the probability for a nest being successful with the distance of the nest to a body of water. A possible ecological explanation for there being a negative relationship between the distance and the success probability might be that near water bodies there are usual more

insects, and hence more food which means improved body condition and hence incentives for attempting reproduction.

We could therefore hypothesize that

$$\theta_i = f(d_i)$$

and that this function is such that for a given nest i, the larger the distance d_i the smaller the probability of success θ_i.

A possible way to conceptualize that relationship might be by choosing f in a way that forces θ to be in the plausible range for a probability. One such way is to assume a logistic relationship between the probability and the distance

$$\theta_i = \frac{1}{1 + exp(-(\alpha + \beta d_i))}$$

which we could easily code up in R as

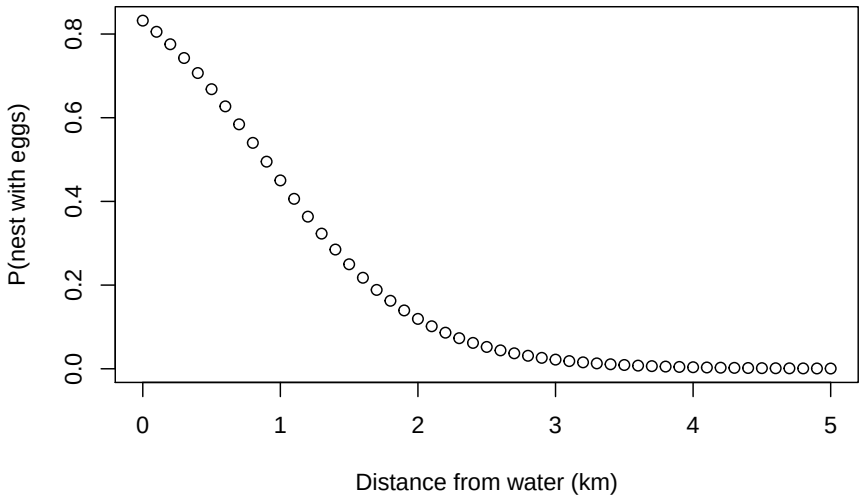
```
thetafd<-function(alpha,beta,d){
  res<-1/(1+exp(-(alpha+beta*d)))
  return(res)
}
```

We can visualize what that function might look like for some arbitrary values for α and β.

```
alpha<-1.6
beta <- -1.8
```

These values imply that the probability of a nest near a body of water having eggs is relatively high, in fact, if just by the water ($d = 0$) then around 0.83, and around 1km it will be 0.45 but by about 5km from the water the probability is down at effectively 0.

```
alldists<-seq(0,5,by=0.1)
plot(alldists,thetafd(alpha,beta,alldists),ylab="P(nest with eggs)",xlab="Distance from water (km
```



Having done this, then it is relatively simple to modify the likelihood for theta that we used for θ, to replace the θ by a function of the relevant covariate d

```
logliktheta d=function(pars,data){
  nests<-data[,1]
  dists<-data[,2]
  alpha<-pars[1]
  beta<-pars[2]
  theta<-thetafd(alpha,beta,dists)
  minusloglik=-sum(log(theta^sum(nests==1))+sum(log((1-theta)^sum(nests==0))))
  return(minusloglik)
}
```

and so we could maximize this likelihood based on the data, both the successes and the distances to the water observed above

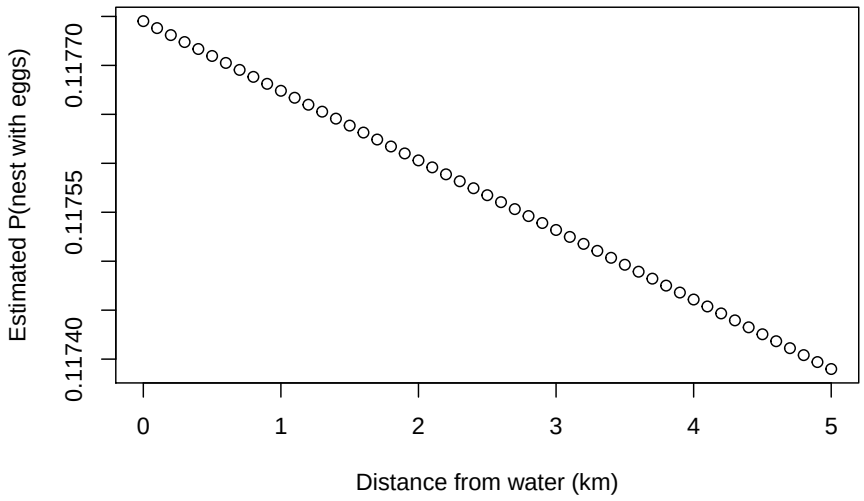
```
thetadMLE<-optim(par=c(1,1),fn=logliktheta d,data=data.frame(nests=nests,dists=dists))
thetadMLE
```

```
## $par
## [1] -2.0139580650 -0.0006849422
##
## $value
## [1] 30.7879
##
## $counts
## function gradient
```

```
##       81       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Now, this provides an interesting ecological insight. Unlike the `naive` estimator for the probability of a nest having eggs we obtained above, of 0.4, we now can relate that probability to the distance from the water. And what we see is that the probability of a random nest having nests might be well below that. We just happened to consider a small sample for which two out of five nests were near the water. While the `naive` estimate might change considerably depending on the relative amount of nests near the water versus nests away from the water, if the main determinant of nest success was the distance from the water, our final model would allow a better estimate of the true probability of success of a nest.

```
alldists<-seq(0,5,by=0.1)
plot(alldists,thetafd(thetadMLE$par[1],thetadMLE$par[2],alldists),ylab="Estimated P(nest with egg
```



15.6 Likelihood, above and beyond

The examples presented in this chapter should illustrate a couple of fundamental points regarding likelihoods.

The likelihood is a concept that allows one to obtain parameter estimates for model parameters that might allow ecological insights, provided that the model parameters are interpretable

The likelihood allows to establish a full framework that can be extended an generalizable to become as complicated as a researcher might want. However, if you complicate the model enough, not surprisingly, the likelihood might become hard to write down and even harder to maximize, and then one needs to find alternatives.

One such alternative might actually be to change the inferential framework. The likelihood is actually the basis of the Bayesian inferential paradigm, but there you combine the information from the likelihood with a prior to provide a posterior distribution from which inferences can be made. The prior will represent previous knowledge about unknown quantities, like the parameters in a model. That opens a world of possibilities, like the possibility of including information from previous studies to make more reliable inferences based on the data collected in the current study. is a story for another book, however.

Most of the statistical methods that you have used are probably based on a likelihood. That is the case for most parametric statistical tests like t-tests and ANOVA's (despite the fact that ANOVA's are typically implemented not exploiting the likelihood but using decompositions of the sources of variation into sums of squares associated to different components of the underlying linear models. That was a story we touch upon briefly, from a different perspective (see chapter **??**aula8).

Chapter 16

Class 14: 10 10 2020 GLMs

16.1 What are GLMs

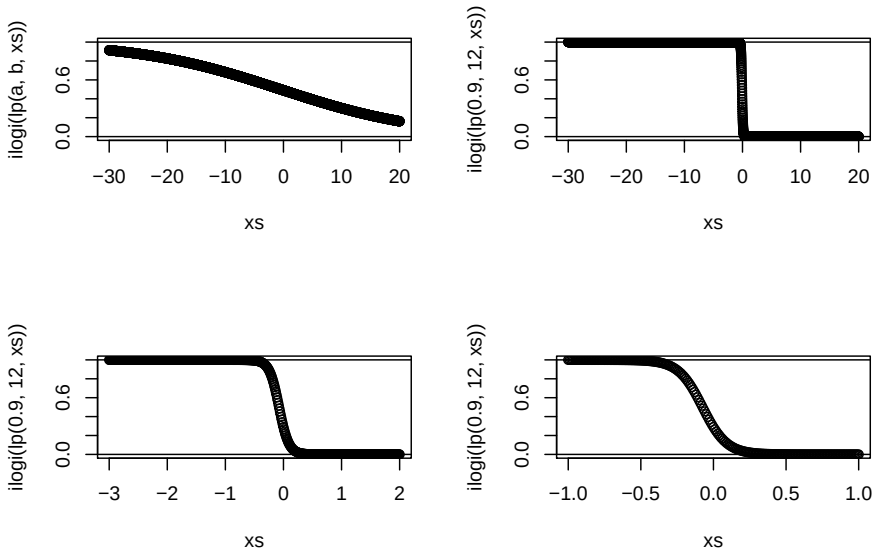
Here we talk about link functions

```
lp<-function(a,b,x){
  res<-a+b*x
  return(res)
}

ilogi<-function(lp){
  res<-1/(1+exp(lp))
  return(res)
}
```

Illustrate how a link function can be used to ensure that a prediction only returns admissible values

```
a<-0.04
b<-0.08
par(mfrow=c(2,2))

xs<-seq(-30,20,by=0.01)
plot(xs,ilogi(lp(a,b,xs)),ylim=c(0,1))
abline(h=0:1,lty=1)


xs<-seq(-30,20,by=0.01)
plot(xs,ilogi(lp(0.9,12,xs)),ylim=c(0,1))
abline(h=0:1,lty=1)
```

```
xs<-seq(-3,2,by=0.01)
plot(xs,ilogi(lp(0.9,12,xs)),ylim=c(0,1))
abline(h=0:1,lty=1)


xs<-seq(-1,1,by=0.01)
plot(xs,ilogi(lp(0.9,12,xs)),ylim=c(0,1))
abline(h=0:1,lty=1)
```



16.2 An example analysis

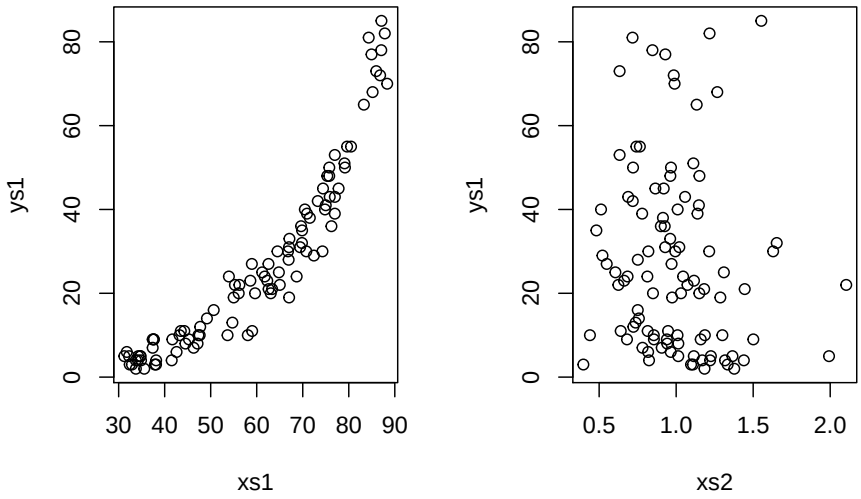
Here we talk about GLMs

```
set.seed(121)
#define sample size
n<-100
#get a predictor variable
low<-30
high<-90
xs1<-runif(n,low,high)
#get a second potentially predictor
xs2<-rgamma(n,10,10)
#define linear predictor
lp1<-0.01+0.05*xs1
```

```
#get the mean value
Eys1<-exp(lp1)
#get actual data
ys1<-rpois(n,Eys1)
```

We can plot the data

```
par(mfrow=c(1,2))
plot(ys1~xs1)
plot(ys1~xs2)
```



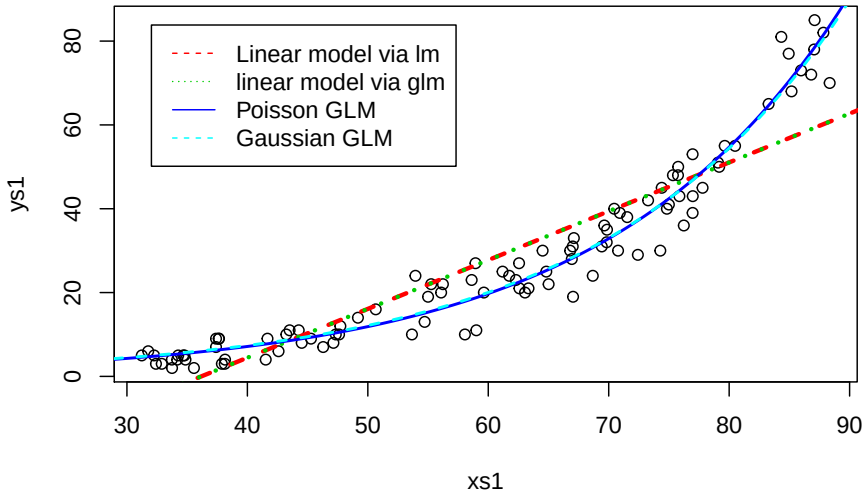
Now, we can fit some models to data

```
lm1A<-lm(ys1~xs1)
lm1B<-glm(ys1~xs1)
glmPoi<-glm(ys1~xs1,family=poisson(link="log"))
glmGau<-glm(ys1~xs1,family=gaussian(link="log"))
```

and we can show these overlayed on data - wtf is going on...?

```
plot(ys1~xs1)
xs<-seq(low-5,high+5,by=0.5)
novosdados<-data.frame(xs1=xs)
predslm1A<-predict(lm1A,newdata=novosdados)
predslm1B<-predict(lm1B,newdata=novosdados)
predsglmPoi<-predict(glmPoi,newdata=novosdados,type="response")
predsglmGau<-predict(glmGau,newdata=novosdados,type="response")
```

```
lines(xs,predslm1A,col=2,lty=2,lwd=3)
lines(xs,predslm1B,col=3,lty=3,lwd=3)
lines(xs,predsglmPoi,col=4,lwd=2)
lines(xs,predsglmGau,col=5,lty=2,lwd=2)
legend("topleft",legend=c("Linear model via lm","linear model via glm","Poisson GLM","G
```



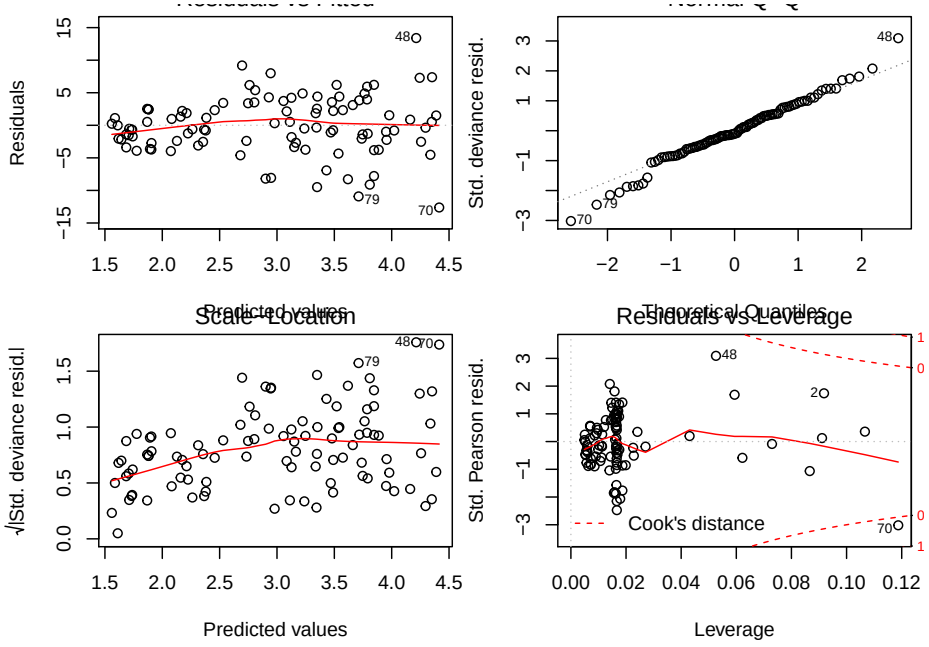
Just for fun

```
AIC(lm1A,lm1B,glmGau,glmPoi)
```

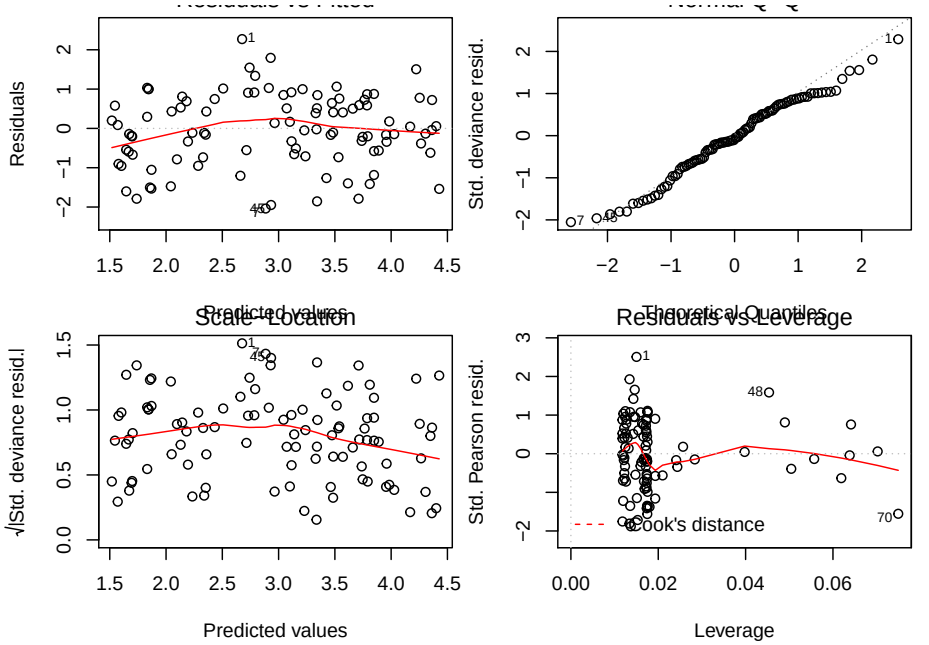
```
##        df      AIC
## lm1A    3 718.4863
## lm1B    3 718.4863
## glmGau  3 586.4525
## glmPoi  2 564.2675
```

Diagnostics

```
par(mfrow=c(2,2),mar=c(4,4,0.5,0.5))
plot(glmGau)
```




```
par(mfrow=c(2,2),mar=c(4,4,0.5,0.5))
plot(glmPoi)
```



Chapter 17

Class 15: 11 11 2020

Chapter 18

Class 16: 17 11 2020

Chapter 19

Final Words

This is a work in progress. If you have read it I hope you enjoyed it. Please send me any feedback you might have on it. Suggestions, questions, comments, edits, requests, etc will be taken seriously. I would be really very happy to use them and to aknowledge your contribution to make this more useful for students and readers.

Bibliography

Faraway, J. J. (2006). *Extending the Linear Model with R.* Chapman & Hall / CRC.

Zuur, A. F., Ieno, E. N., and Meesters, E. H. (2009a). *A Beginner's Guide to R.* Springer.

Zuur, A. F., Ieno, E. N., and Smith, G. M. (2007). *Analyzing Ecological Data.* Springer.

Zuur, A. F., Ieno, E. N., Walker, N., Saveliev, A. A., and Smith, G. M. (2009b). *Mixed Effects Models And Extensions In Ecology With R.* Springer.