

Notes for Ecological Modelling

Tiago A. Marques, Susana França, Ana Sofia Reboleira

2020-10-29

Contents

1	Introduction	5
2	Introduction	7
2.1	What is a regression?	7
2.2	The general linear model	7
2.3	Bookdown template leftovers	8
3	Class 6 13 10 2020	11
3.1	Implementing a regression	11
3.2	Simulating regression data	21
4	Class 7 14 10 2020	37
4.1	Task 1	37
4.2	Task 2	45
5	Class 8 20 10 2020 - t-test and ANOVA are just linear models	55
5.1	The t-test	55
5.2	ANOVA	62
6	Class 9: 21 10 2020 - ANCOVA is (also) just a linear model	71
6.1	Common slope, different intercepts per treatment	72
7	Class 10: 27 10 2020	79
7.1	Same story, another spin	79
8	Class 11: 03 10 2020 ANCOVA with different slopes: interactions	93
8.1	Modeling a data set	99
8.2	Conclusion	104
9	Final Words	105
10	Agradecimientos	107

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 latter became obvious that with a bit of extra effort put into it these might become material that would be useful to others beyond the ME students.

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

Chapter 2

Introduction

This is being written as a bookdown project. Maybe one day it will become a book, for now, these are notes I am using for my course on “Modelação Ecológica” at FCUL.

2.1 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 of covariates (a.k.a. predictors, explanatory or independent variables).

2.2 The general linear model

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

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

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** $\beta_0 + \beta_1 x_1 + \dots + \beta_k x_k$ (μ , a linear function of k covariates). In general books tend to represent this as

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

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

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

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

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

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

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

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

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

being Gaussian, i.e. $e_i \sim \text{Gau}(0, \sigma)$, and where the link function is the identity (i.e. $f(E(y)) = 1 \times E(y) = E(y)$).

2.3 Bookdown template leftovers

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter 2. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter ??.

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

```
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 2.1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 2.1.

```
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

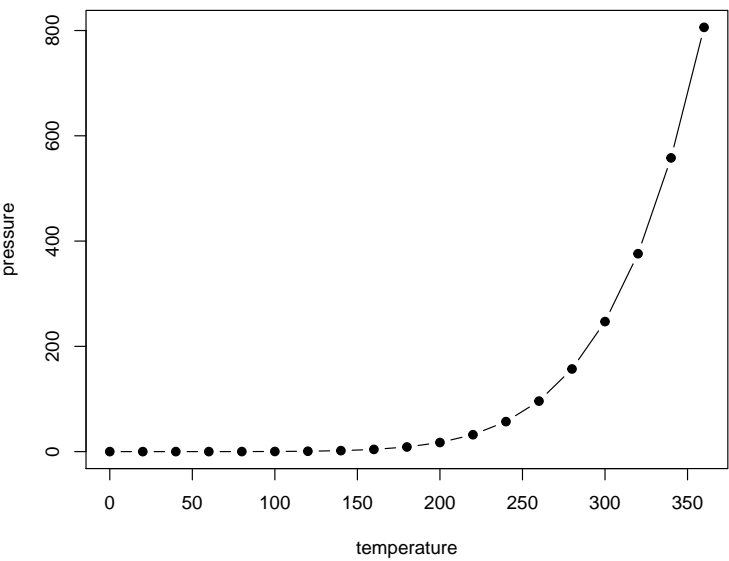



Figure 2.1: Here is a nice figure!

Table 2.1: Here is a nice table!

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa

You can write citations, too. For example, we are using the **bookdown** package (Xie, 2020) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015).

Chapter 3

Class 6 13 10 2020

3.1 Implementing a regression

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

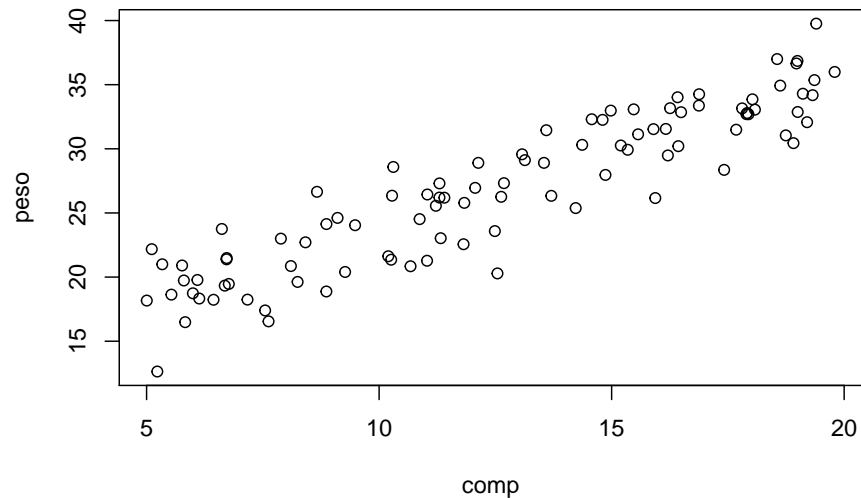
Read the data in

```
#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:

```
glm1ag <- glm(peso~comp,data=lagartos,family=gaussian(link="identity"))
summary(glm1ag)
```

```
##
## 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

$$\text{peso} = 11.72 + 1.2 \times \text{comp}$$

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 difference 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 concept that is hard to explain in this applied context, but relates to the number of available independent bits of information available. So trust me when I say that we lose 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))
```

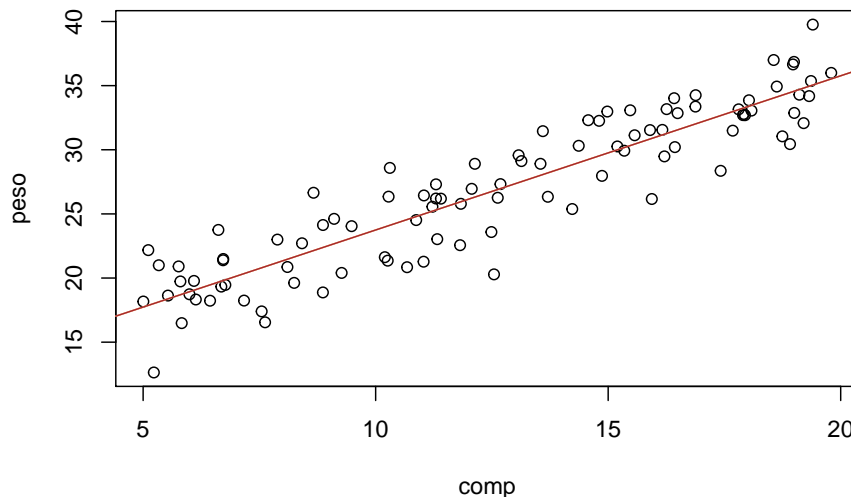
```
## [1] 2.414641
```

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`

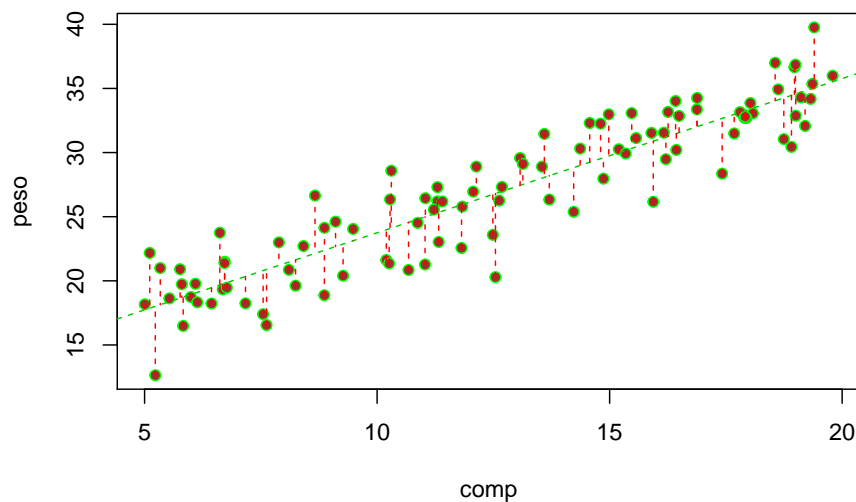
```
summary(lmlag)$coefficients
```

```
##           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)
```

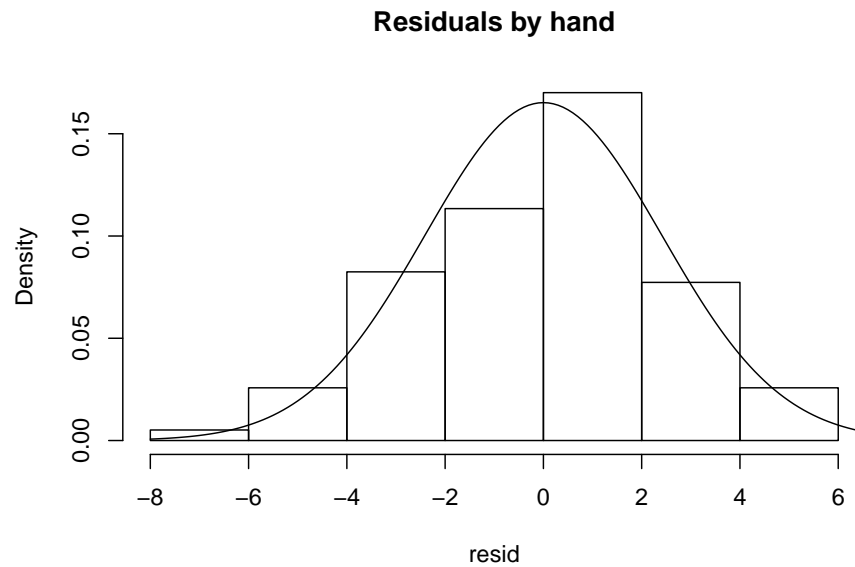
```
#plot the data
with(lagartos,plot(peso~comp,pch=21,bg="brown",col="green"))
abline(lmlag,col=3,lty=2)
#add residuals
with(lagartos,segments(x0 = comp,y0 = peso, x1= comp, y1=estimated,lty=2,col="red"))
```



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).

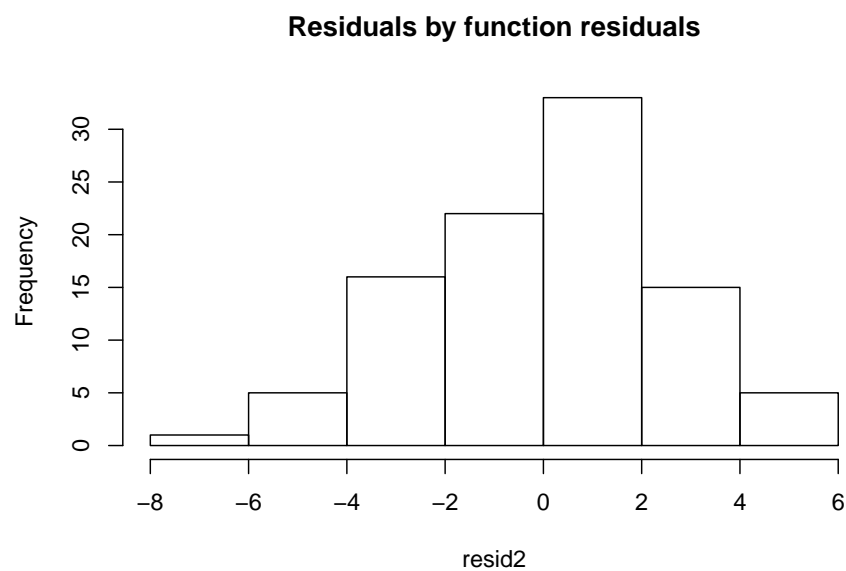
The residuals should, if the model is reasonable - and here that should be the case - be well approximated by a Gaussian distribution. Note we can get the values of the residuals by the difference between the observations and the estimated values, as we did above. We can look at their histogram below

```
hist(resid,main="Residuals by hand",freq=FALSE)
#adding the theoretical 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")
```



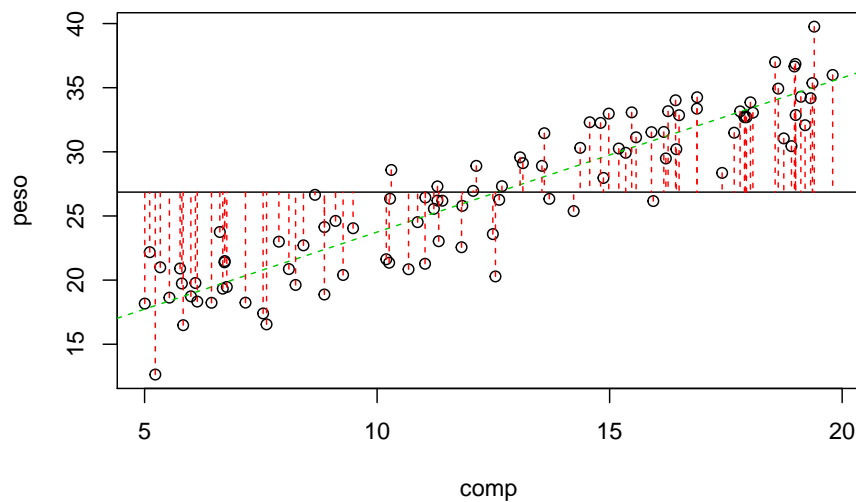
It is often said that the R^2 represents the amount of variation in the response that the regression explains. Why is that?

Because if you assume that all the variability in the response data, the y_i the difference 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
```

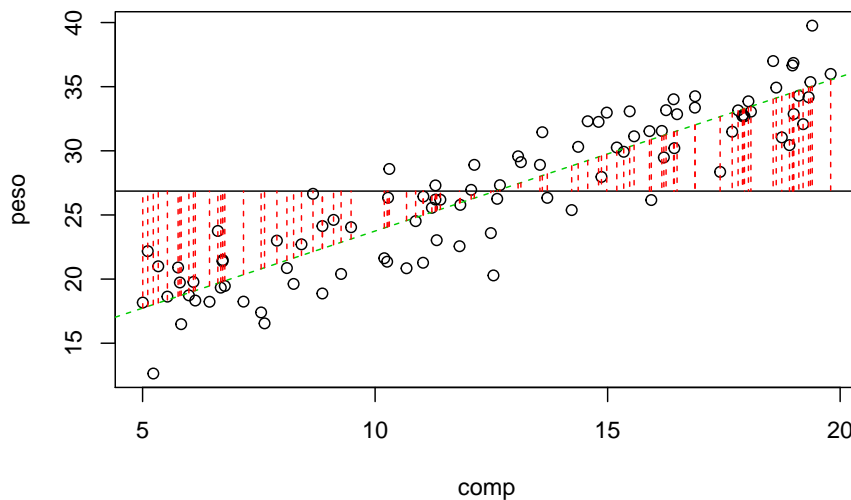
And that... as noted above... is the $R^2=0.8391$. This comes from the fact that all of the variability in the data (the y , the response, here the `peso`) can be decomposed into the variability explained by the model, and the unexplained variability, that of the errors. In a formula

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

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 reason 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=aQ32qTjqjJM> 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 !

3.2 Simulating regression data

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

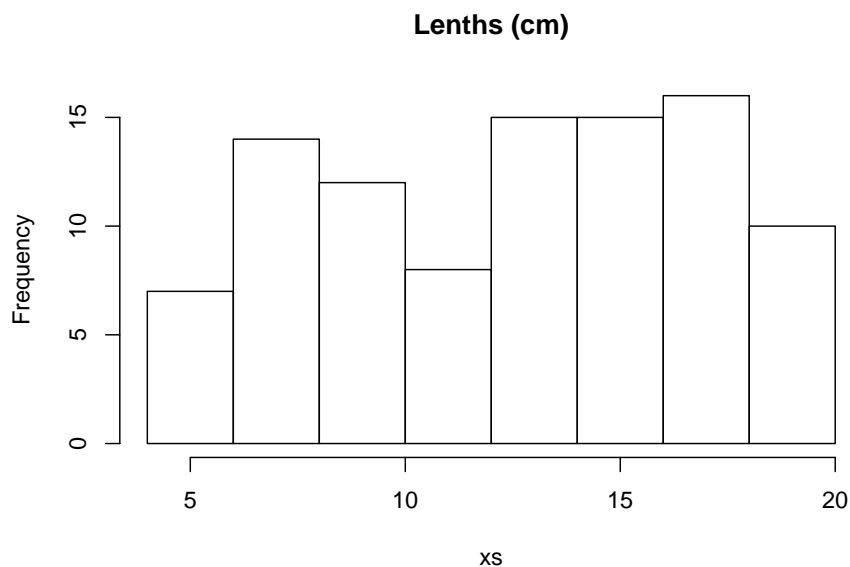
$$peso = 12 + 1.2 * comp$$

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

As in the data we will have 97 lizzards

Then you were told to create the lengths:

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

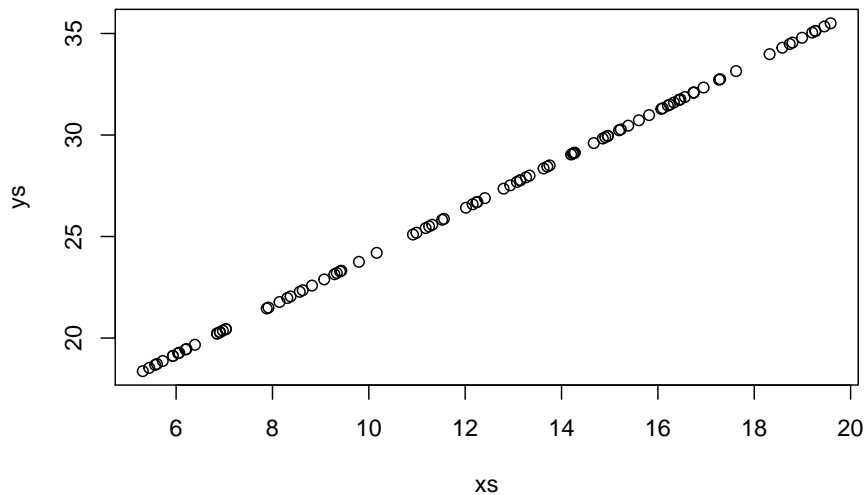


and then to create weights of lizards

```
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))
```

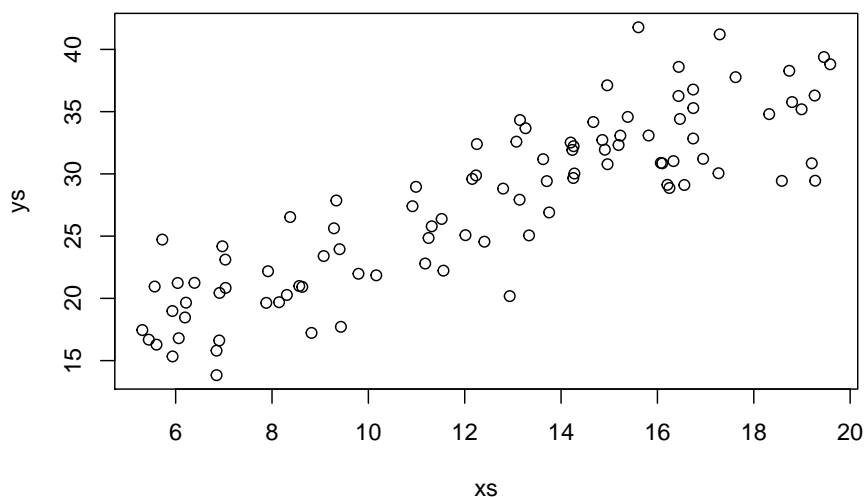
```
## Warning in summary.lm(lm(ys ~ xs)): essentially perfect fit: summary may be
## unreliable

##
## 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 model 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 it 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 compliments
comp.sim <- runif(n,5,20)
```

```

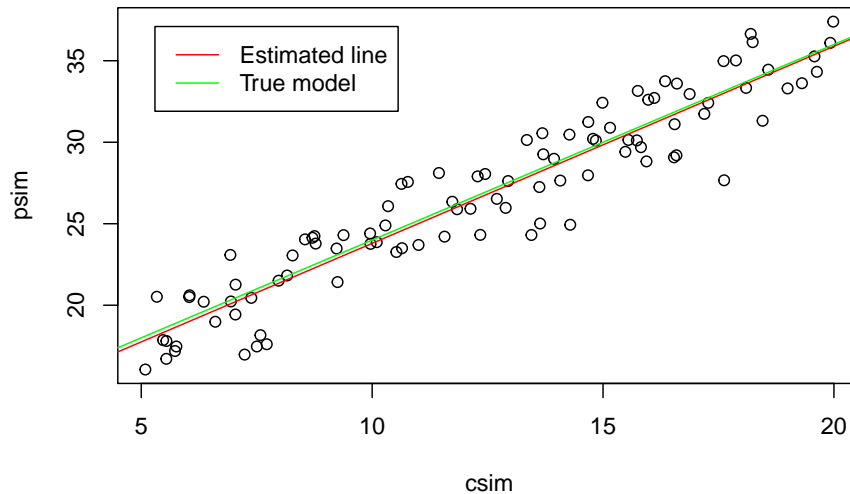
a<-12
b<-1.2
#similar pesos
peso.sim<-a+b*comp.sim+rnorm(n,mean=0,sd=2)
data.sim=data.frame(csim=comp.sim,psim=peso.sim)
plot(psim~csim,data=data.sim)
mod.sim<-lm(psim~csim,data=data.sim)
abline(mod.sim,col="red")
summary(mod.sim)

```

```

##
## 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

```

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

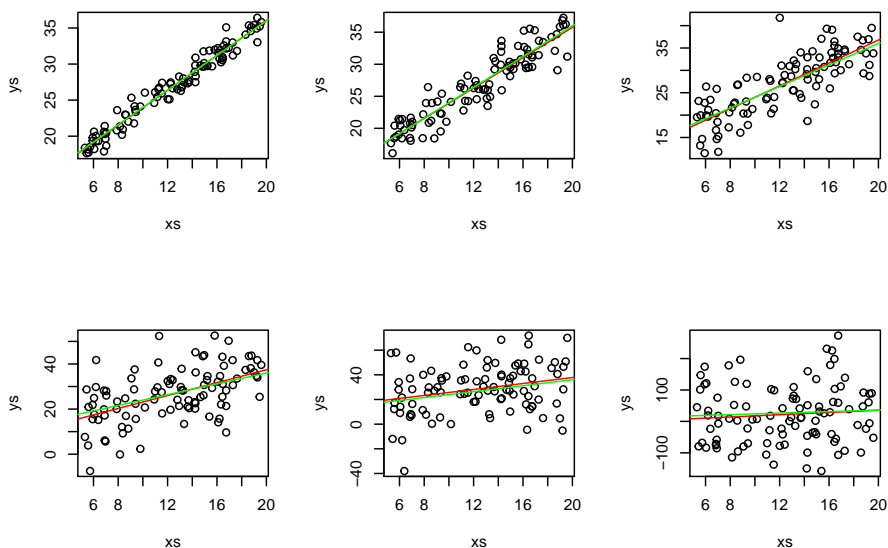
Now, let's consider there's more and less variance. We also add to each plot the real line (that with the true parameter values) and the one with the estimated parameter values.

```
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")

```



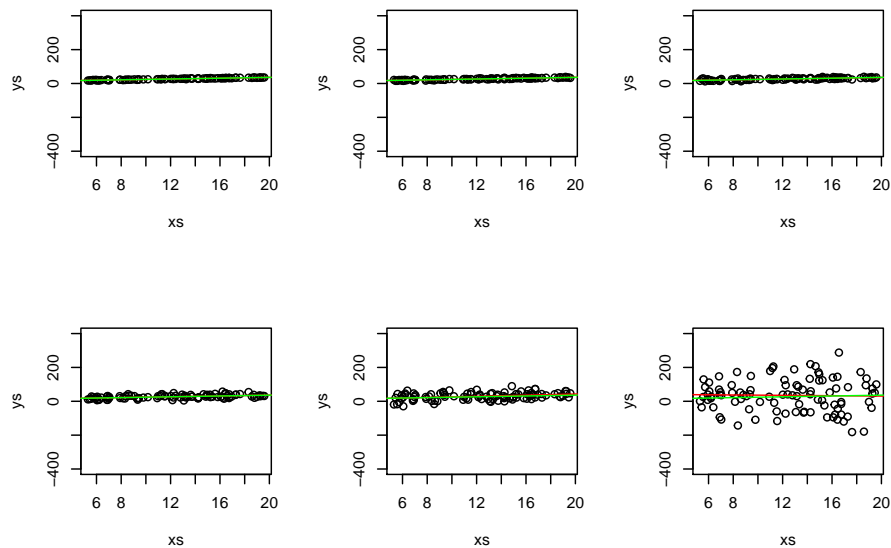
Not surprisingly, as the variance increases, we get data that more and more looks like it is not coming from a real linear process.

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
```

```
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
```

```
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
```

```
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
```

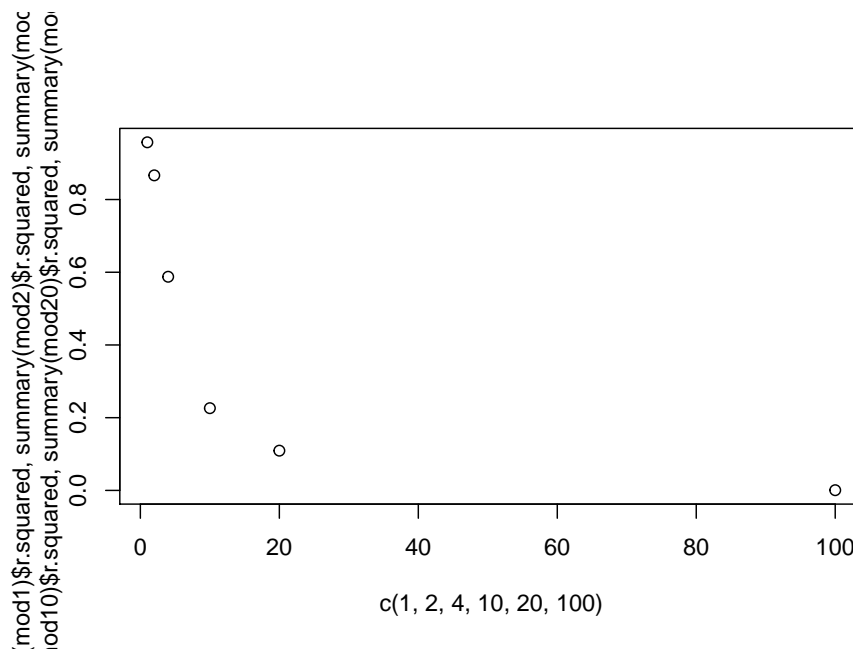
```
summary(mod100)
```

```
##
## 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 deviations, i.e., variances, as variance is standard deviation squared), so hard to tell...

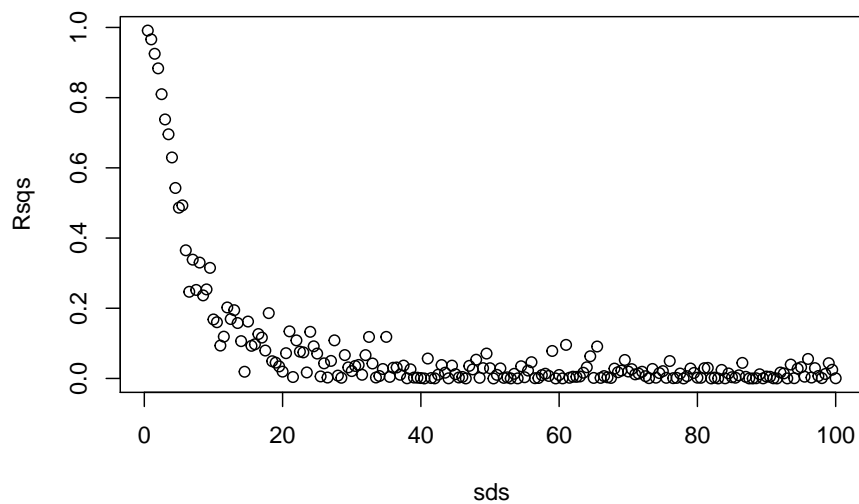
Let's show off in R...

```
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)

```



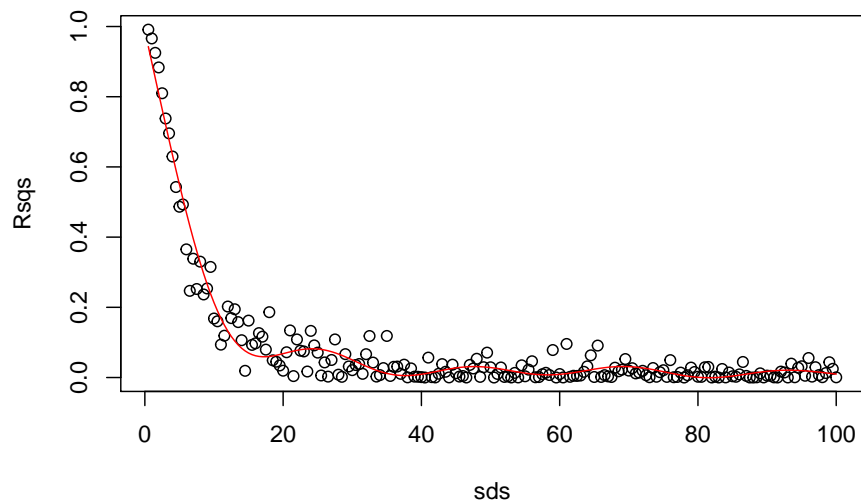
How cool is that!! Actually, this means we can model the R^2 as a function of the original variance! But we would not be able to model it using a linear model...

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

```

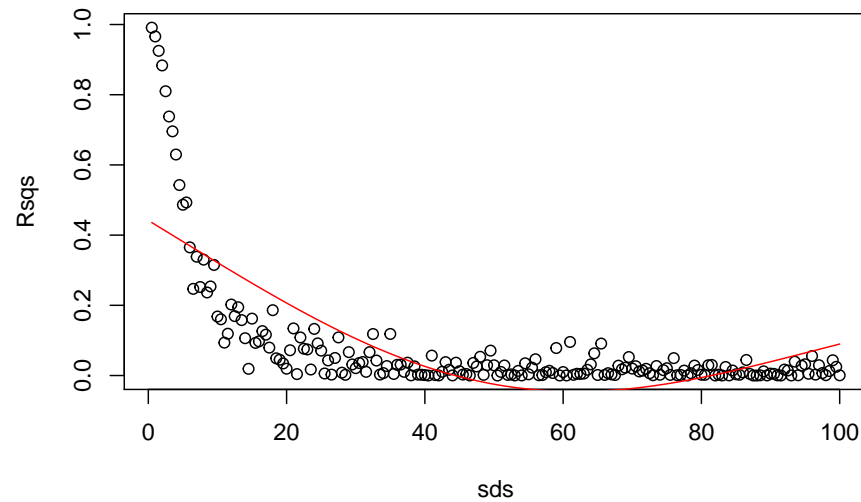
library(mgcv)
gam1=gam(Rsqs~s(sds),link=log)
#make predictions to plot the estimated GAM model
predRsqs=predict.gam(gam1,newdata = list(sds=sds),type="response")
plot(sds, Rsqs)
lines(sds, predRsqs, col="red")

```

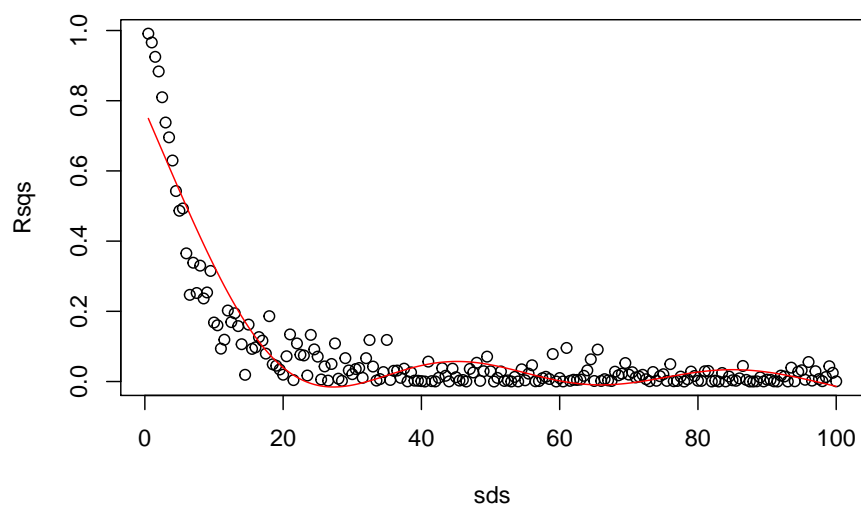
Aha... remember what we talked in class today? It seems like we have overfitted. Then, I constrain the GAM.

```
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 4

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.

4.1 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 the 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))

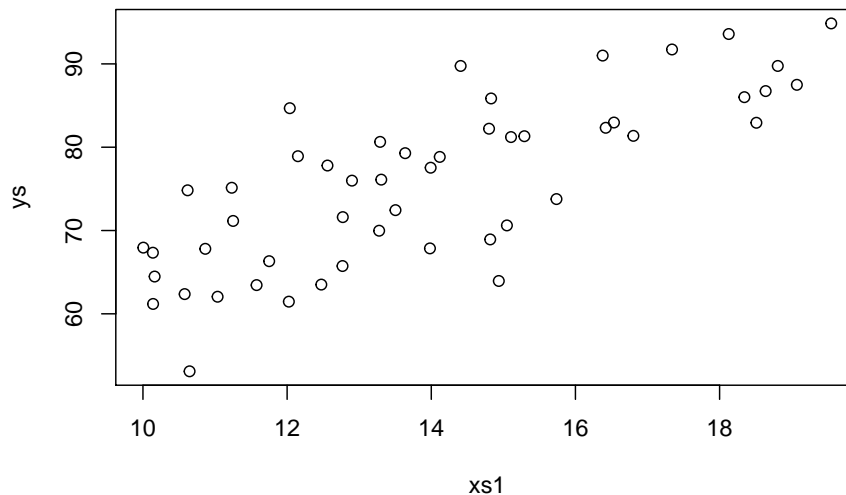
```

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)

```



```

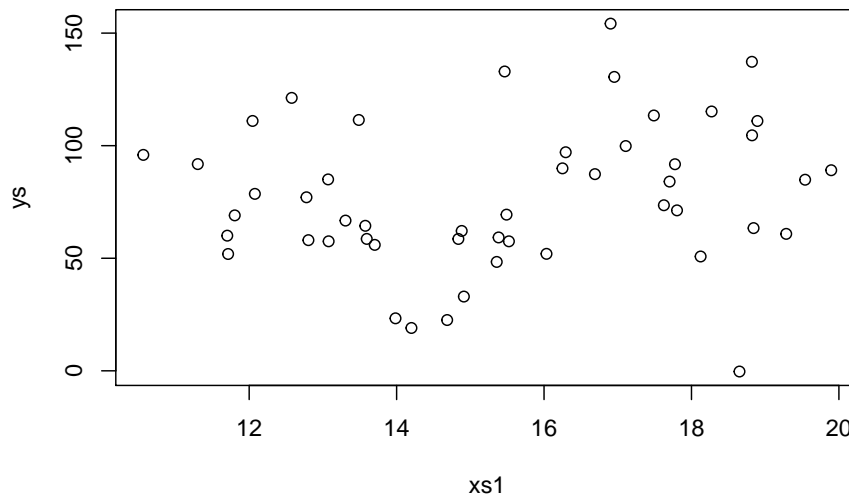
#look at model summary
summary(lm(ys~xs1+xs2+xs3+xs4))

```

```
##
## 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 up 10 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))
```

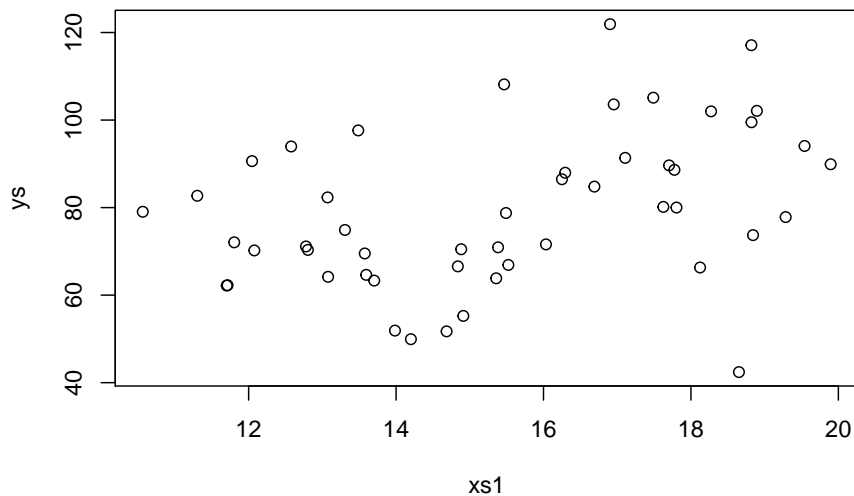
```
##
## Call:
## lm(formula = ys ~ xs1 + xs2 + xs3 + xs4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -73.25 -17.86  -6.76   21.25   68.42
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   22.8661    41.3133   0.553   0.583
## xs1           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))

```

```

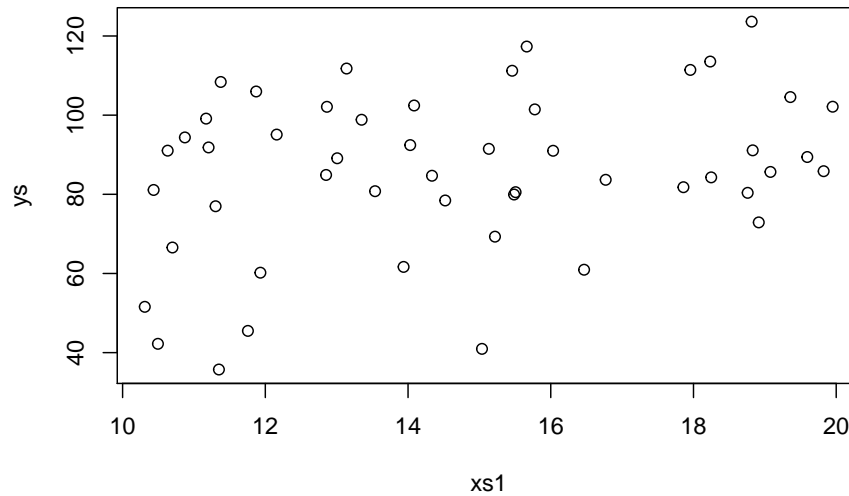
##
## Call:
## lm(formula = ys ~ xs1 + xs2 + xs3 + xs4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -36.624  -8.927  -3.380   10.623   34.211
##

```

```
## 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, b_1 , b_2 , to increase and decrease 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 remember 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

$$y = \beta_0 + \beta_1 x s_1 + \beta_2 x s_2$$

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

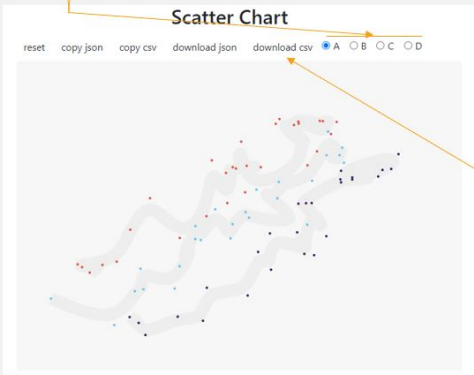
```
folder<-"extfiles/"
#folder<-"../Aula7 14 10 2020/"
d41 <- read.csv(file=paste0(folder,"data4lines.csv"))
n <- nrow(d41)
```

4.2 Task 2

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

Task 2:

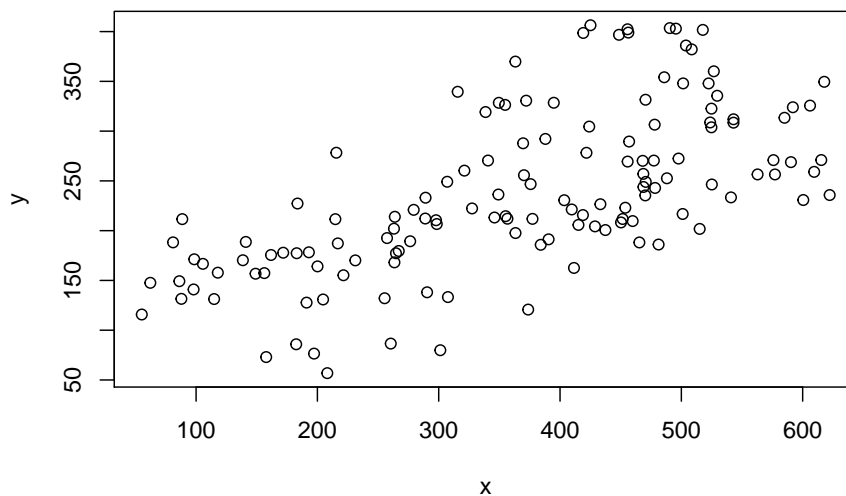
1. Go to <https://drawdata.xyz/>
2. Draw 4 lines 3 with about the same slope and one with a different slope (select A, B, C and D)
3. Download data: data4lines.csv (note data will have y, x, and z, for groups A, B, C, D)
4. Read the data into R and represent in a plot with the group z in different colors
5. Add a legend (tip: explore function `legend`)
6. Fit a multiple regression model to the data, considering $y \sim x + z$, & explore the results



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

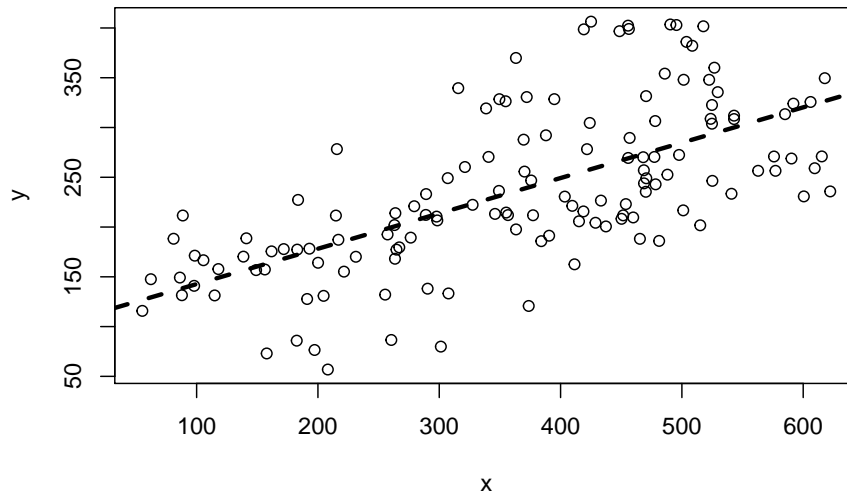
```
#read the data
#folder<-"../Aula7 14 10 2020/"
folder<-"extfiles/"
data4lines <- read.csv(file=paste0(folder,"data4lines.csv"))
#plot all the data
plot(y~x,data=data4lines)
```



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

These correspond to observations from weights and lengths 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
lm1linesG<-lm(y~x,data=data4lines)
abline(lm1linesG,lwd=3,lty=2)
```



```
summary(lmlinesG)
```

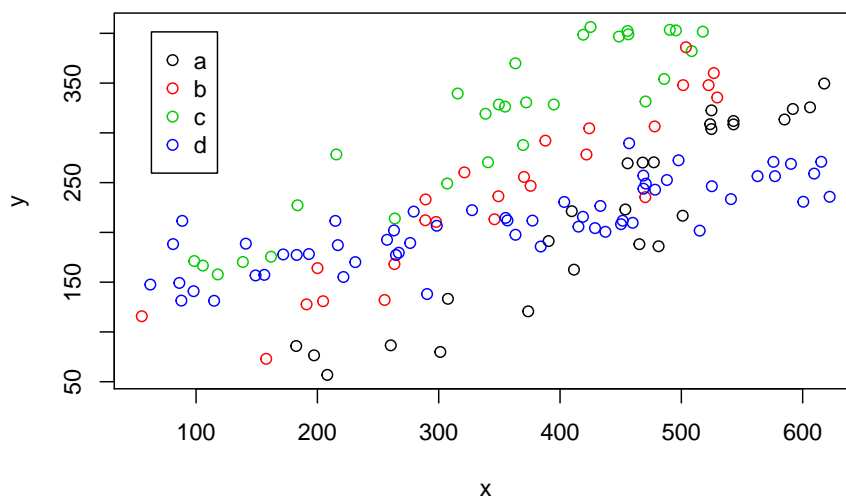
```
##
## Call:
## lm(formula = y ~ x, data = data4lines)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -134.304  -44.188   -1.995   29.432  148.202
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 107.17590   14.03069   7.639 3.51e-12 ***
## 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 highly 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 scientist decides to look at whether 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 pattern 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 boring, 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
lm1lines<-lm(y~x+z,data=data4lines)
summary(lm1lines)
```

```
##
## 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 mystery that we shall unfold now!

While the model we are fitting might be represented by `weight~length+museum`, the design 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

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

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 mystery, 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 order, 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 intercept of the other museums? 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

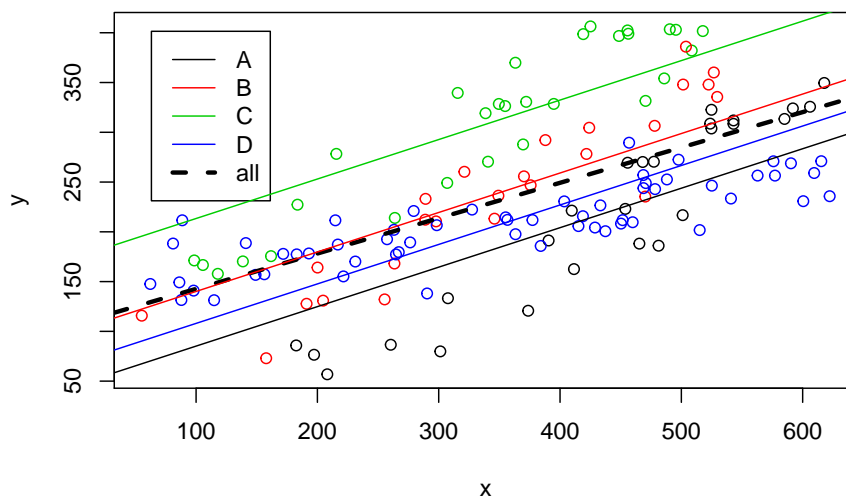
$$y = \beta_0 + \beta_1 \times \text{length} + \beta_{2c} \times zc$$

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

and so, from the above output, that equates to $y = \text{lmlines\$coefficients}[1] + \text{lmlines\$coefficients}[4] \times \text{length}$ or $173.7215247 + 0.3965715 \times \text{length}$.

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 $\text{length}=x=0$) of all lines are indeed easy to get from the model's output

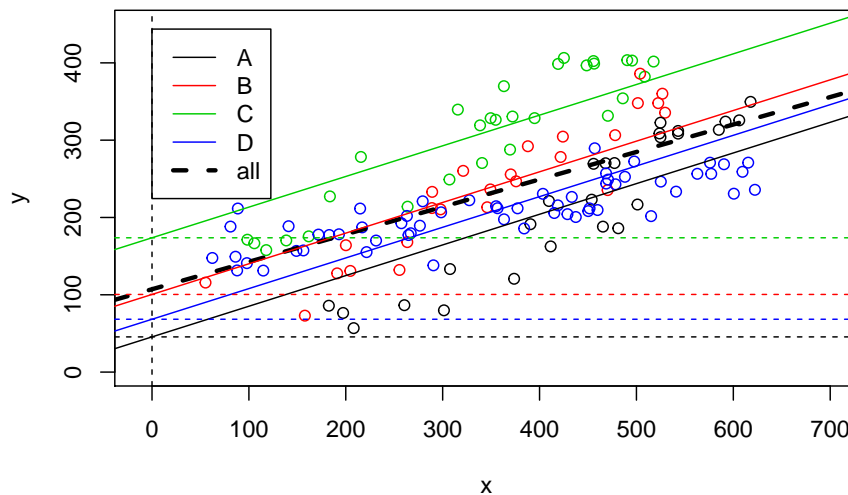
```
#plot all
plot(y~x,xlim=c(-10,700),ylim=c(0,450),col=z,data=data4lines)
legend("topleft",inset=0.05,legend=c(LETTERS[1:4],"all"),col=c(1:4,1),lty=c(rep(1,4),2))
#these are the wrong lines... why?
```

```

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)

```

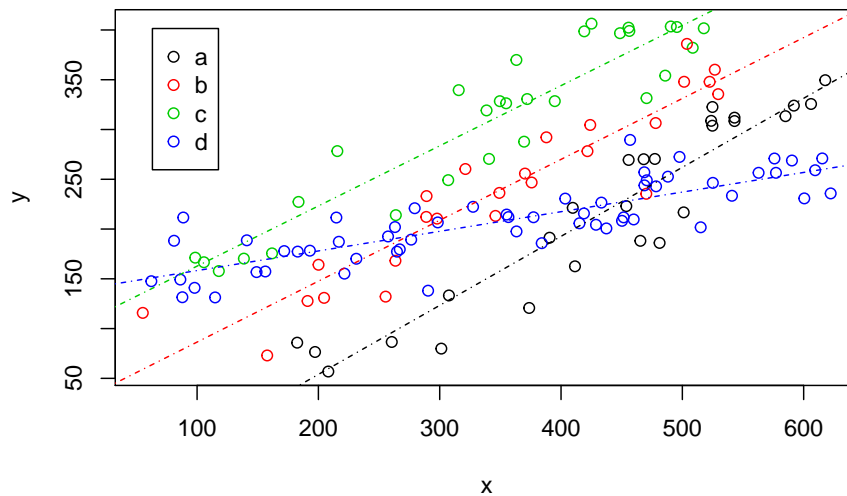


Now, the smart biologist then says that he could also fit a separate line to each museum's data. And so he does, and that looks like this:

```

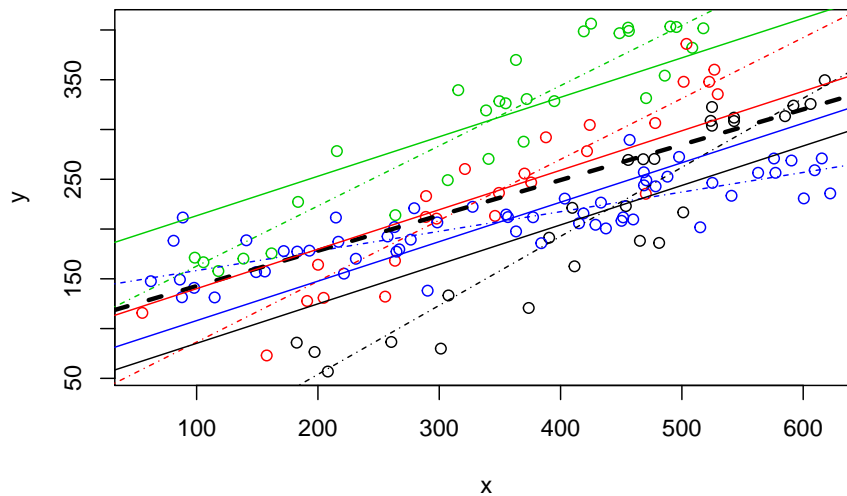
#plot all the data
plot(y~x,col=as.numeric(as.factor(z)),data=data4lines,pch=1)
#completely independet regression lines
abline(lm(y~x,data=data4lines[data4lines$z=="a",]),col=1,lty=4)
abline(lm(y~x,data=data4lines[data4lines$z=="b",]),col=2,lty=4)
abline(lm(y~x,data=data4lines[data4lines$z=="c",]),col=3,lty=4)
abline(lm(y~x,data=data4lines[data4lines$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 mystery that will remain to unfold. For that we will need an 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 `length` 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 5

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.

5.1 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)
```

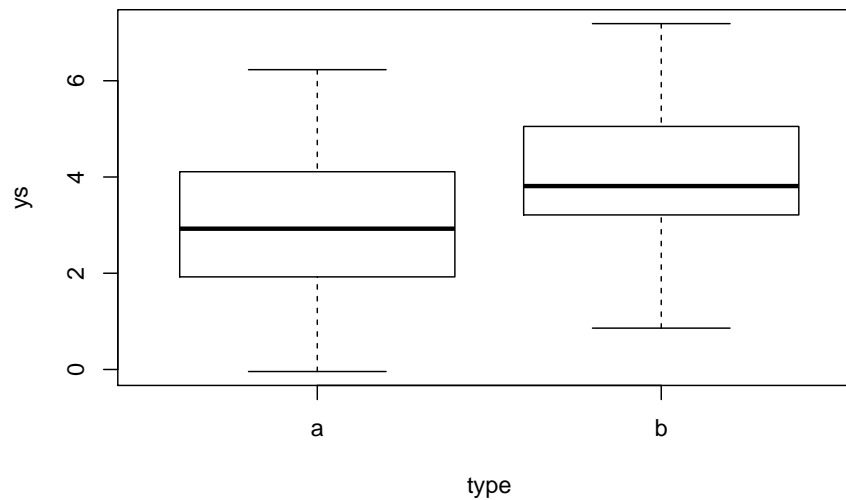
```
## [1] 55
```

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
```

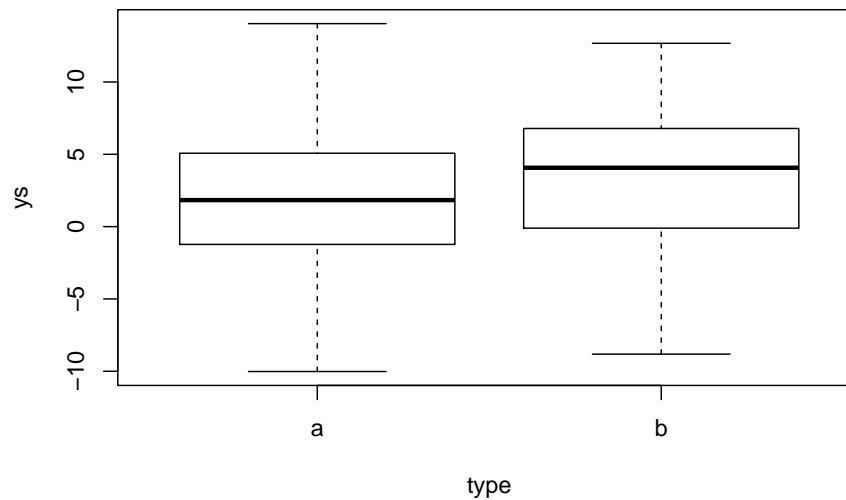
This is required because in a linear model, all the other parameters associated with levels of a factor will be compared to a reference value, that of the intercept, which happens to be the mean under treatment a. Below you will see more examples of this.

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
```

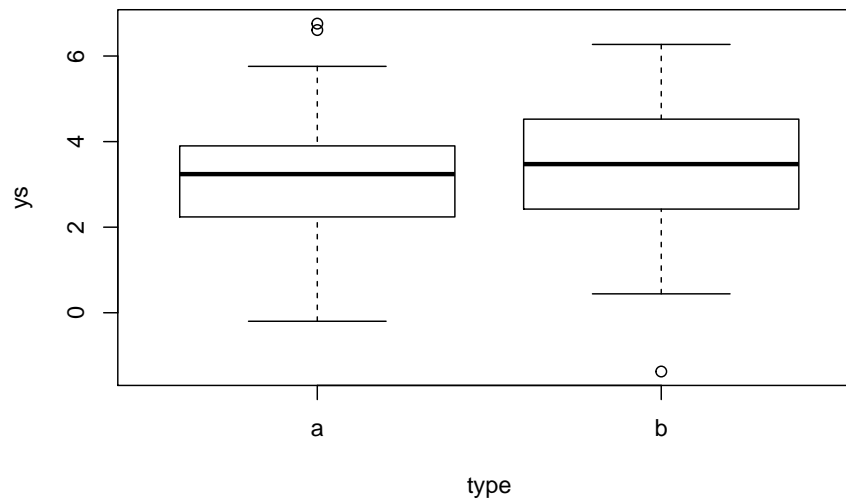
and as you can see, we get the same result for the test statistic, but now with a non significant test.

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
```

5.2 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))
```

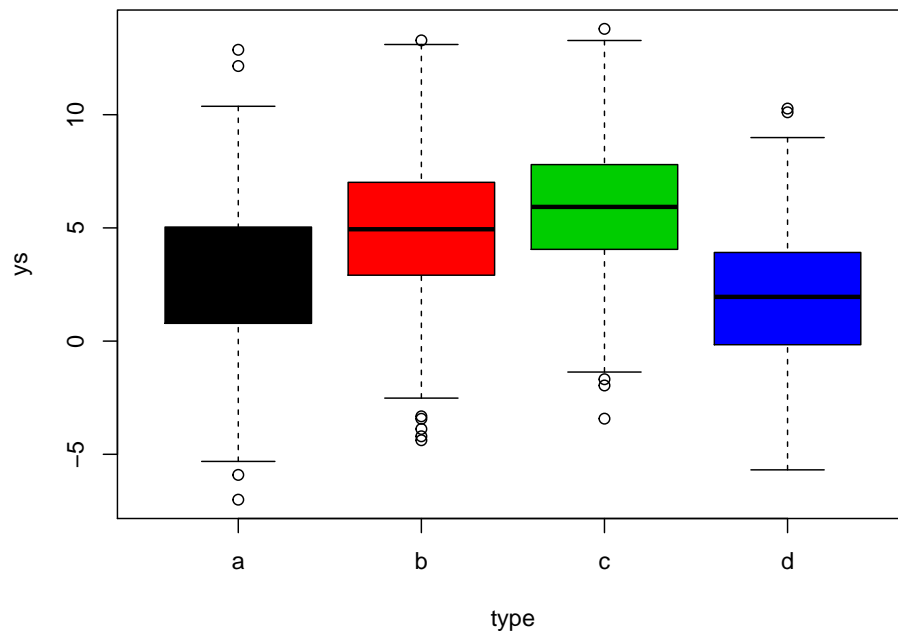
```
## [1] 68
```

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))
```

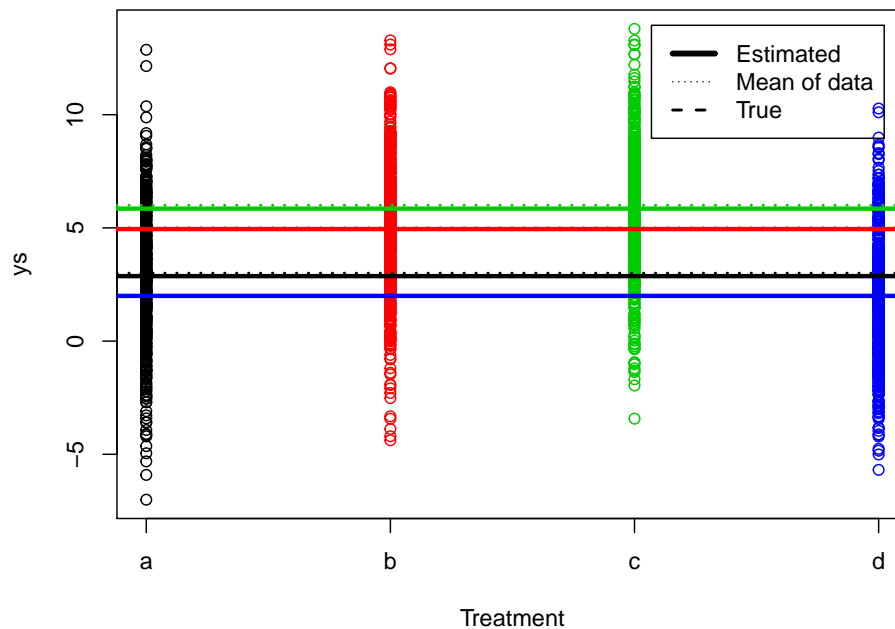
```
##              Df Sum Sq Mean Sq F value Pr(>F)
## type          3    4792   1597.5    183.6 <2e-16 ***
## Residuals    1996   17365     8.7
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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=
```



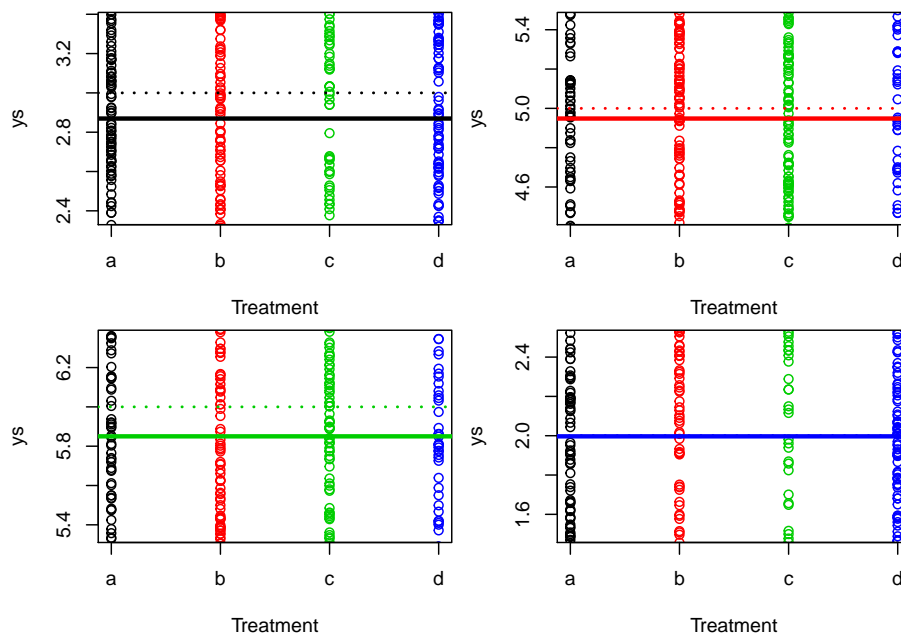
It's not easy to see because these overlap (large sample size, high precision) but the estimated means are really close to the real means. It's a bit easier to see if we separate in 4 plots and zoom in on the mean of each treatment, but still the blue lines are all on top of each other, since the mean 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[treatment]),
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 6

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 (loooooooooooooong!) 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

6.1 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 5.

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.

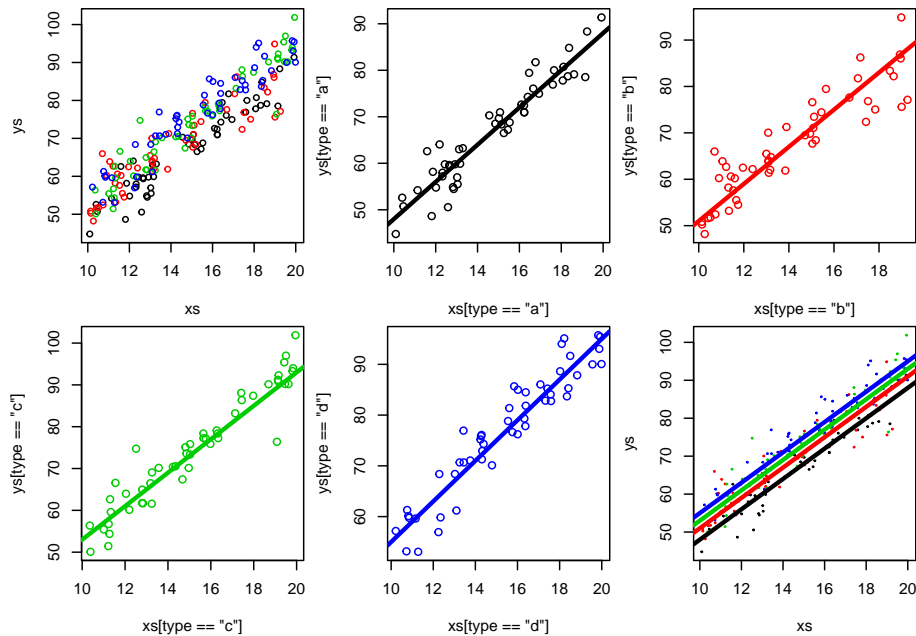
```
#all slopes the same, different intercepts - no interactions
set.seed(1234)
n<-200
nbygroup<-50
xs <- runif(n,10,20)
tr <- c("a","b","c","d")
type <- rep(tr,each=nbygroup)
cores <- rep(1:4,each=nbygroup)
a<-3
b<-4
error<-4
ys <- a+b*xs+
ifelse(type=="a",5,ifelse(type=="b",8,ifelse(type=="c",10,12)))+rnorm(n,0,4)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

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)
```

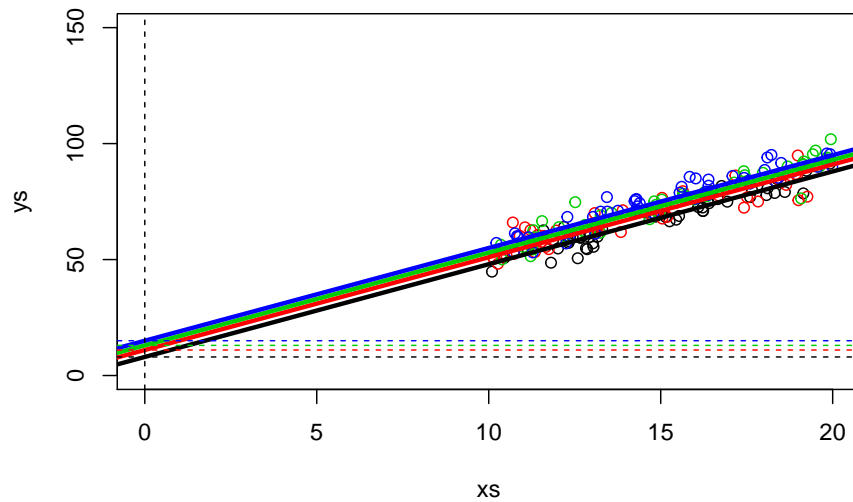

6.1. COMMON SLOPE, DIFFERENT INTERCEPTS PER TREATMENT73

```
#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
```

6.1. COMMON SLOPE, DIFFERENT INTERCEPTS PER TREATMENT⁷⁵

```
## 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

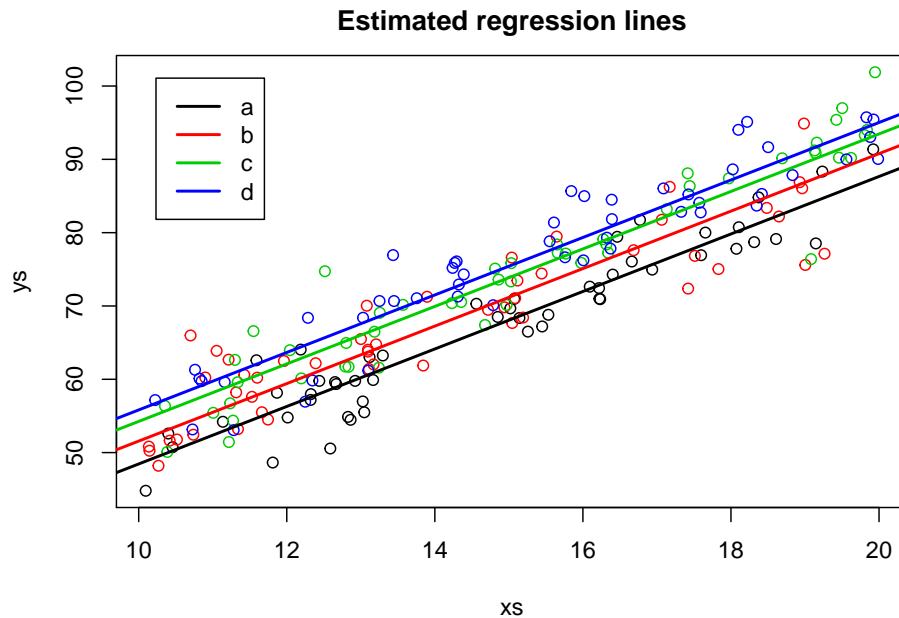
```
lm.ancova1$coefficients[2]
```

```
## [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$coefficients[2],col=2,lwd=2)  
abline(lm.ancova1$coefficients[1]+lm.ancova1$coefficients[4],lm.ancova1$coefficients[2],col=3,lwd=2)  
abline(lm.ancova1$coefficients[1]+lm.ancova1$coefficients[5],lm.ancova1$coefficients[2],col=4,lwd=2)  
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 against 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)
```

#grupo a

```
abline(lm.ancova1$coefficients[1],lm.ancova1$coefficients[2],lwd=1,col=1,lty=2)
```

#grupo b

*# intercept+slope*xs+intercept específico do grupo b*

(intercept+intercept específico do grupo b)+ slope

```
abline(lm.ancova1$coefficients[1]+lm.ancova1$coefficients[3],lm.ancova1$coefficients[2],lwd=1,col=2,lty=2)
```

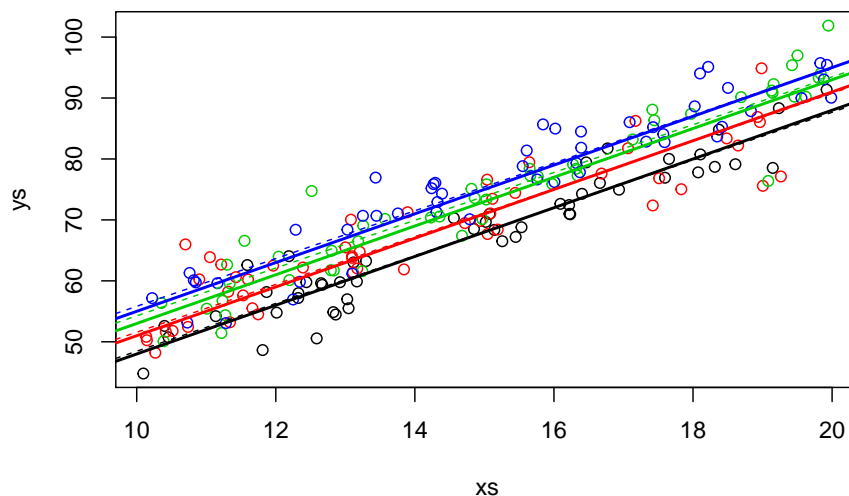
#grupo c

```
abline(lm.ancova1$coefficients[1]+lm.ancova1$coefficients[4],lm.ancova1$coefficients[2],lwd=1,col=3,lty=2)
```

#grupo d

```
abline(lm.ancova1$coefficients[1]+lm.ancova1$coefficients[5],lm.ancova1$coefficients[2],lwd=1,col=4,lty=2)
```

6.1. COMMON SLOPE, DIFFERENT INTERCEPTS PER TREATMENT77



As we can see, they are quite close. The error is small compared with the effect 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 different `types`

considered

```
#explaining it
```

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

```
##          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 $Y = \text{parameters} \times \text{design matrix}$, or $Y = \beta X$ (see e.g. https://en.wikipedia.org/wiki/Design_matrix)

```
#the design matrix
```

```
head(data.frame(xs=x,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 7

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 6, 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,

7.1 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 6 under said perspective. We will do so with the help of two unlikely characters. 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 friend, 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!

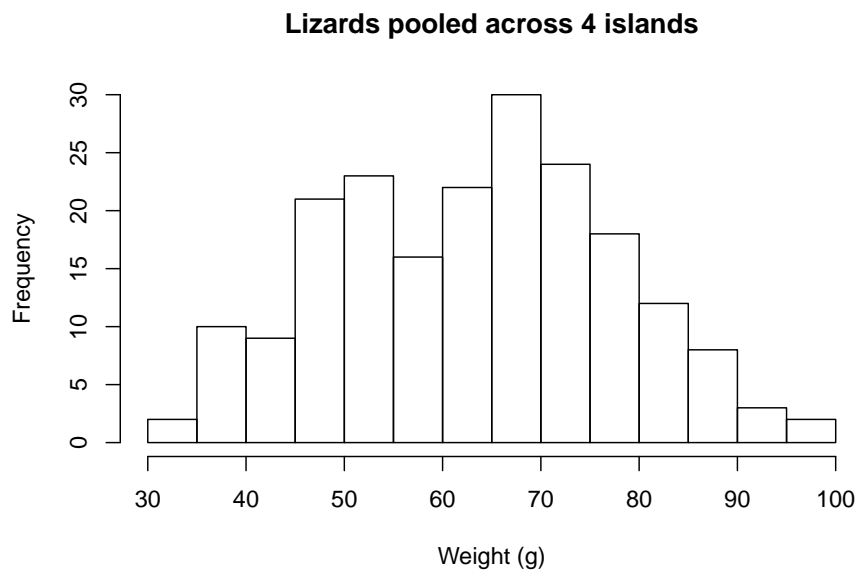
Great when on a journey to a distant archipelago where there are 4 islands, each potentially with a different species of lizard that Great is interested in. The folks providing names to islands where not has imaginative as Great's parents, so the islands are called just A, B, C and D. Imagine that Great did a great job and collected a great sample of lizards in each island. Great is also 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!)

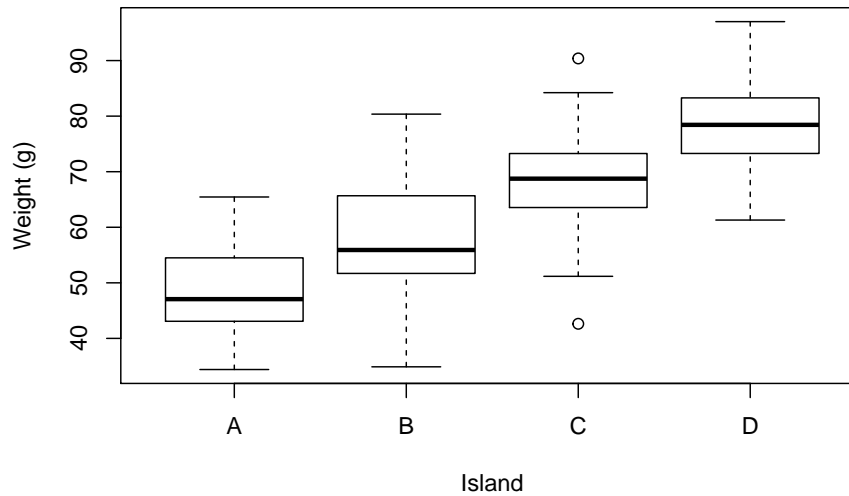
Great plotted the weights, in grams (g), of the captured lizards. These look like this:

```
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 5) will show:

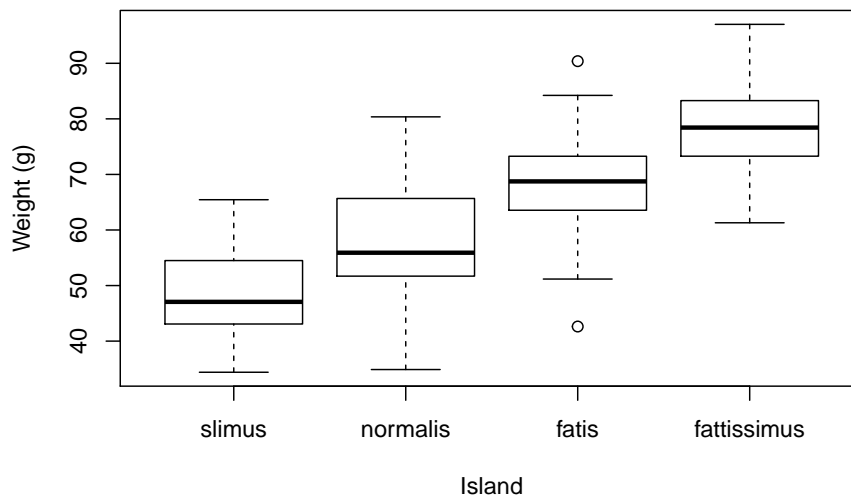
```
summary(lm(ys~type))
```

```
##
## 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 different 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.

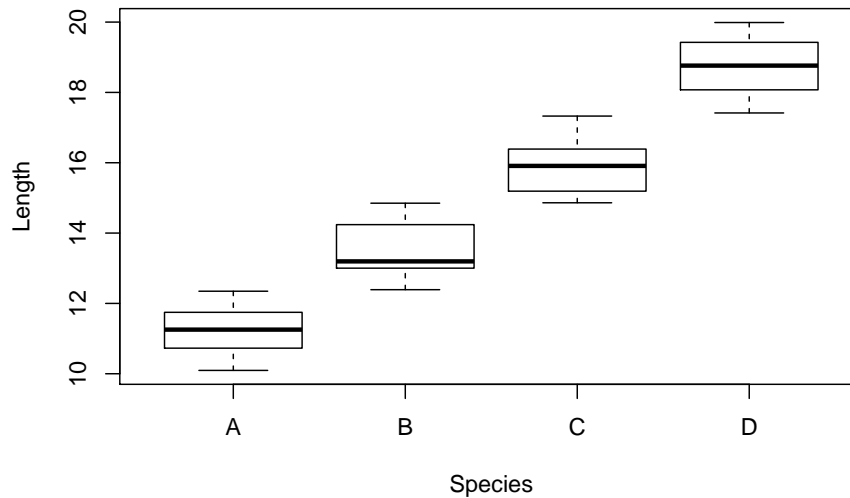
```
spnames <- c("slimus", "normalis", "fatis", "fattissimus")
boxplot(ys~type,ylab="Weight (g)",xlab="Island",
names=spnames)
```



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.

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



```
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 embarrassment 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 continuous variable (length).

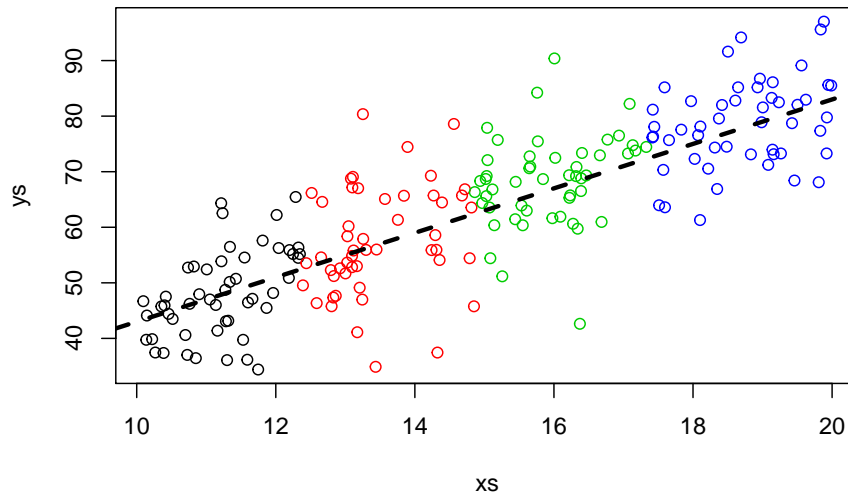
He implements the models and, much to his despair, realizes that, once he accounts for the length, the weights are not different per island. The damn lizards are exactly the same weight in the different islands once you account for their length... :(

```
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 ecological 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 opposite 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 lengths 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:

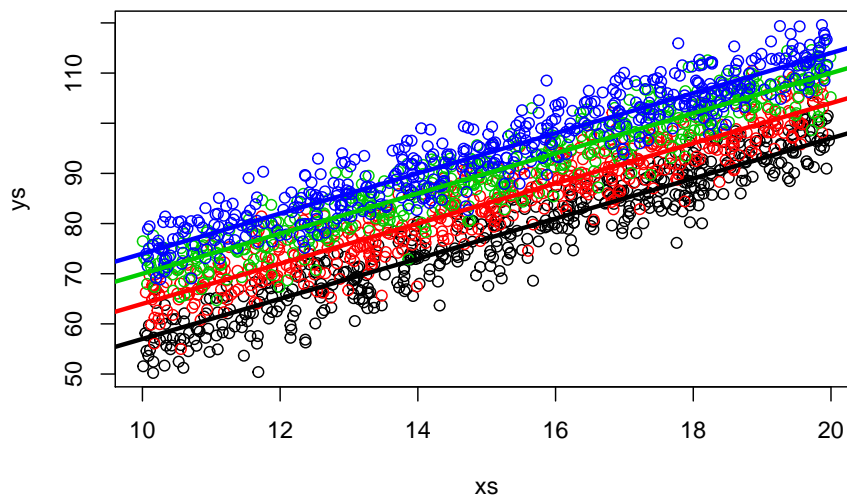
The lizards relationship between weight and length is different per island. Say, akin to what we saw before. Just by magin, we have access to all the lizards in the island.

```
#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 length.

```

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)

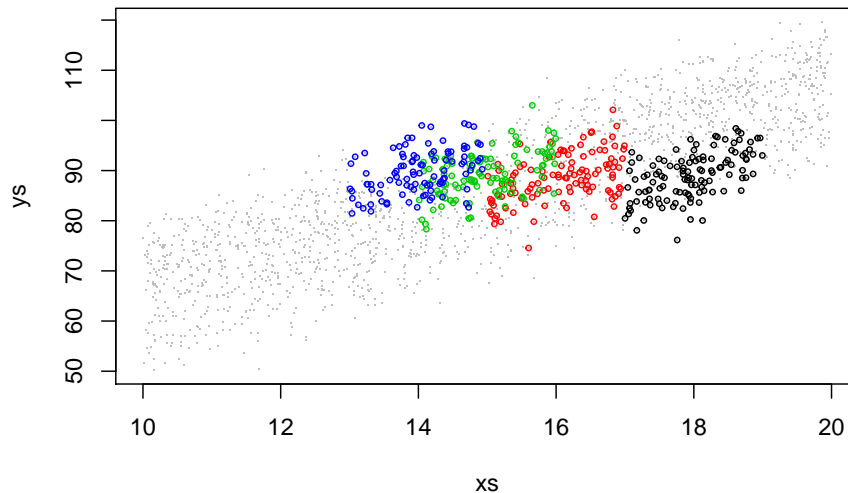
```

```
#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.

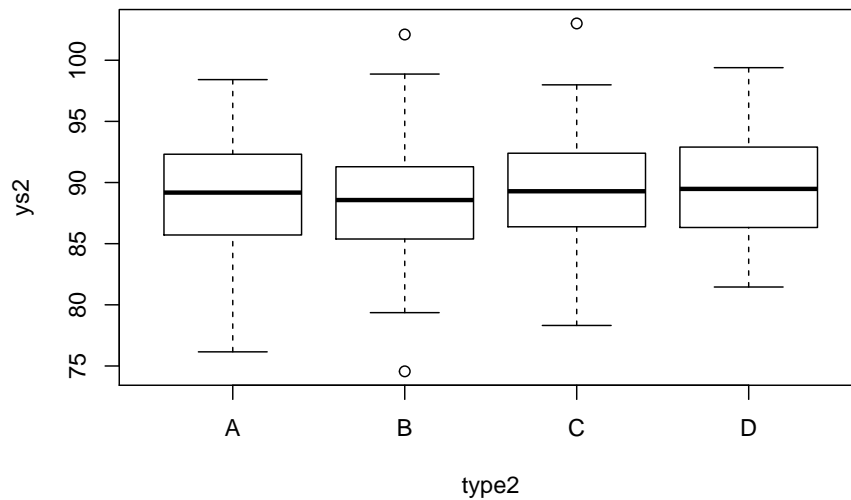
```
plot(xs,ys,pch=".",col="grey")
points(xs2,ys2,col=cores2,cex=0.5)
```



```
#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
```

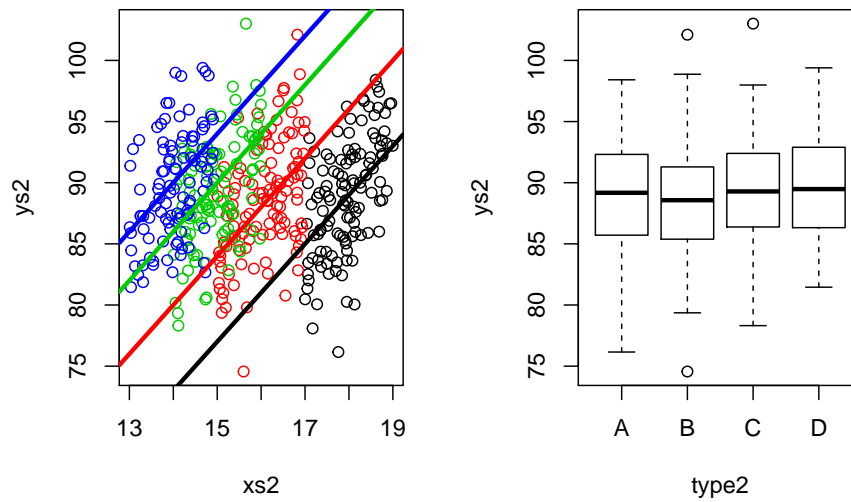
This is where we need to be smart. If we conduct the correct analysis, one that includes and adjusts for the effect of length, the differences in length to weight relationship are clear. The intercepts of the different lines are all different from each other.

```
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)
```

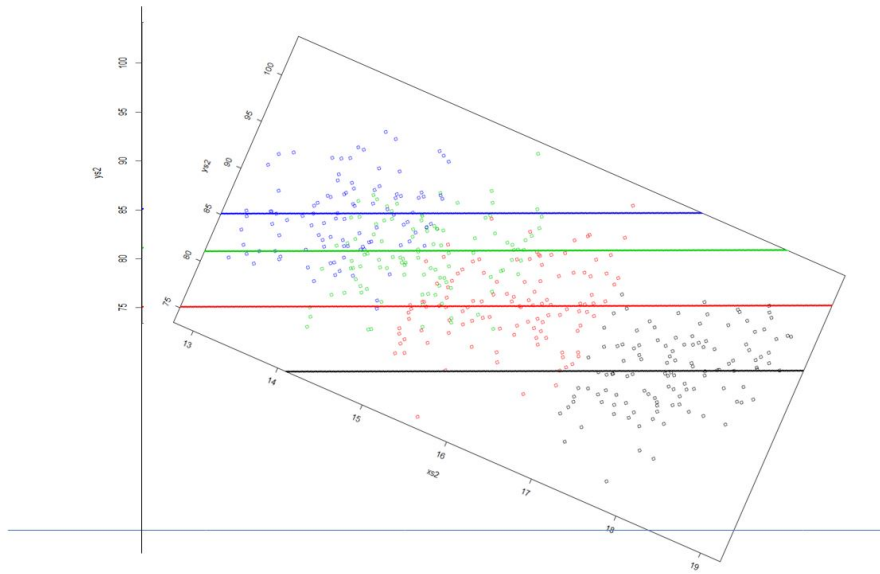


Actually, what the ANCOVA model does is equivalent to looking at the data by rotating the left plot above and see it in the “axis” we care about. That corresponds to the axis such that the slope of the regression lines are aligned with the x-axis of the Cartesian referential.

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, carefully 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 quantitative 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 8

Class 11: 03 10 2020

ANCOVA with different slopes: interactions

The previous model assumed that the slopes were the same across the different groups. What would change if they were different? We extend the previous case to where the slope of the relationship is also different per treatment.

Simulate treatments, same as before, but this gives us the option to change later 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

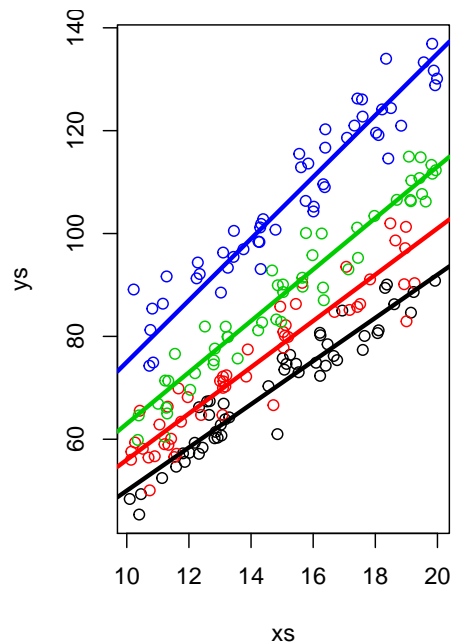
```
ys=3+  
ifelse(type=="a",5,ifelse(type=="b",8,ifelse(type=="c",10,12)))+  
4*xs+ifelse(type=="a",0.2,ifelse(type=="b",0.5,ifelse(type=="c",1,2)))*xs+  
rnorm(200,0,4)
```

note this is the same as what we have below, but below it might be simpler to understand that these do correspond to 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)
```

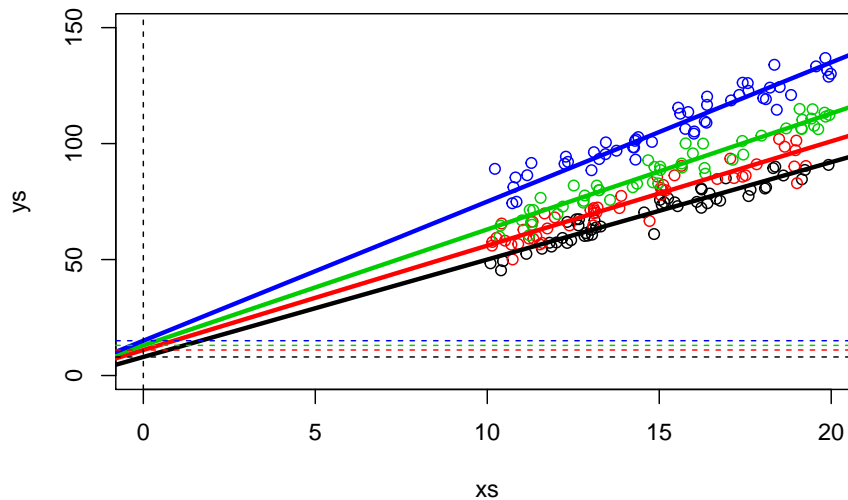
We can look at the data

```
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 and force that

```
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)
```



Now, we implement the ANCOVA linear model, but with an *interaction* term!

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

and we look at the output of the model

```
sum.lm.ancova2
```

```
##
## Call:
## lm(formula = ys ~ xs + type + xs * type)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.2514  -2.7831  -0.2006   3.0000  10.1051
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    7.0480     3.4994   2.014 0.045397 *
## xs             4.2667     0.2345  18.198 < 2e-16 ***
## typeb          4.5418     4.6361   0.980 0.328489
## typec          3.7997     4.7096   0.807 0.420787
## typed         15.9380     4.8844   3.263 0.001305 **
## xs:typeb       0.1848     0.3160   0.585 0.559468
## xs:typec       0.8442     0.3103   2.721 0.007106 **
## xs:typed       1.2325     0.3203   3.848 0.000162 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.331 on 192 degrees of freedom
## Multiple R-squared:  0.9578, Adjusted R-squared:  0.9562
## F-statistic: 622.3 on 7 and 192 DF,  p-value: < 2.2e-16
```

check how this is an output similar to the ANOVA (implemented via `aov`, the R function that produces ANOVA tables from expressions akin to linear models)

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

```
##              Df Sum Sq Mean Sq  F value    Pr(>F)
## xs              1  48776   48776  2600.680 < 2e-16 ***
## type            3  32550   10850   578.501 < 2e-16 ***
## xs:type         3    379     126    6.742 0.000239 ***
## Residuals     192   3601        19
## ---
## 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 622.3, 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.

The most interesting aspect it that, naturally, we can check the values of the estimated coefficients, and in particular how to estimate the corresponding regression lines per group

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

```
## (Intercept)
##      7.048022
```

```
lm.ancova2$coefficients[2]
```

```
##          xs
##  4.266671
```

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

```
## (Intercept)
##      11.5898
```

```
lm.ancova2$coefficients[2]+lm.ancova2$coefficients[6]
```

```
##          xs
```



```
## 4.451449
#type c
lm.ancova2$coefficients[1]+lm.ancova2$coefficients[4]
```

```
## (Intercept)
## 10.84769
lm.ancova2$coefficients[2]+lm.ancova2$coefficients[7]
```

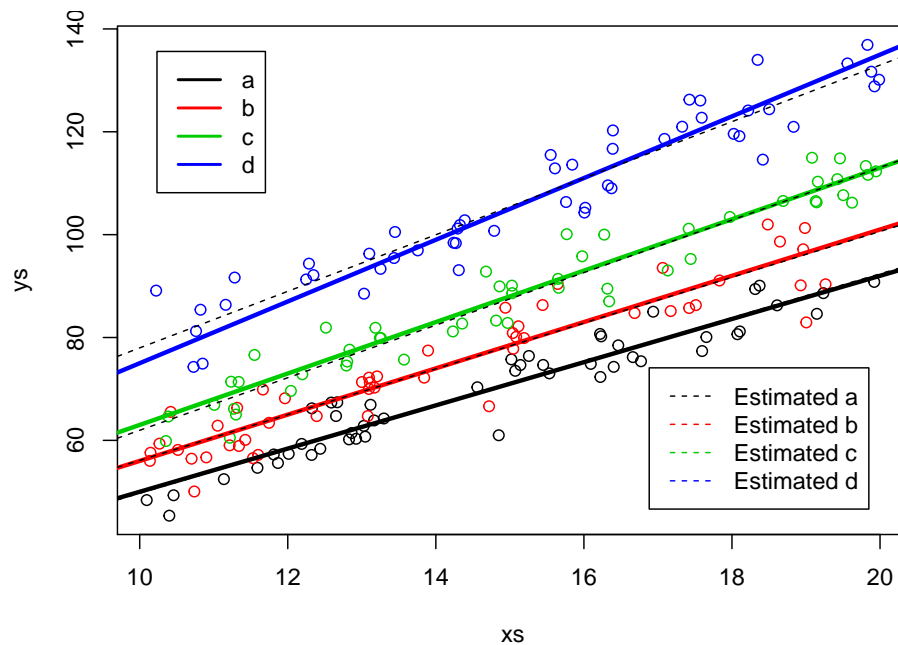
```
## xs
## 5.110884
#type b
lm.ancova2$coefficients[1]+lm.ancova2$coefficients[5]
```

```
## (Intercept)
## 22.98597
lm.ancova2$coefficients[2]+lm.ancova2$coefficients[8]
```

```
## xs
## 5.499171
```

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

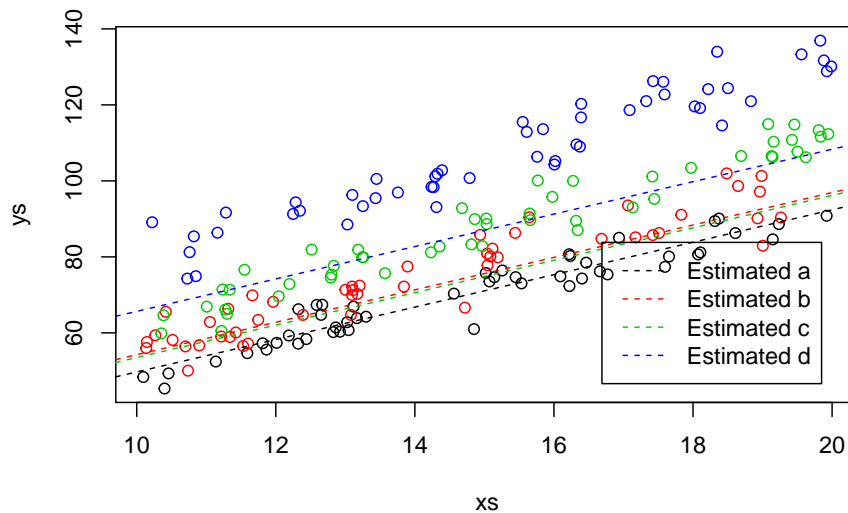
```
abline(lm.ancova2$coefficients[1],lm.ancova2$coefficients[2],lwd=1,col=1,lty=2)
```

```
abline(lm.ancova2$coefficients[1]+lm.ancova2$coefficients[3],lm.ancova2$coefficients[2],
```

```
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)
```



8.1 Modeling a data set

In a given dataset we might want to know if the interaction is needed or not, or in other words, if the different lines might have different slopes, or not.

We illustrate that with the data set `data4lines.csv` that we considered before (add link here!)

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

And now we fit a model without interaction, as we did before,

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

```
##
## 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
```

and after a model with the interaction term

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

```
##
## Call:
## lm(formula = y ~ x + z + x:z, data = data4lines)
##
## 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 second model is best! Which 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.

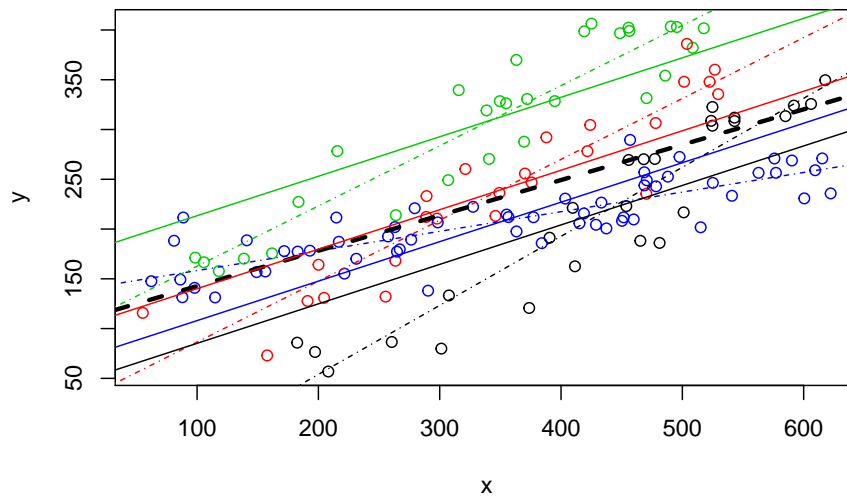
```
AIC(lmANC,lmIinesI)
```

```
##          df      AIC
## lmANC      6 1433.414
## lmIinesI    9 1321.927
```

Now lets go back to the data. Remember a plot we had on this dataset before?

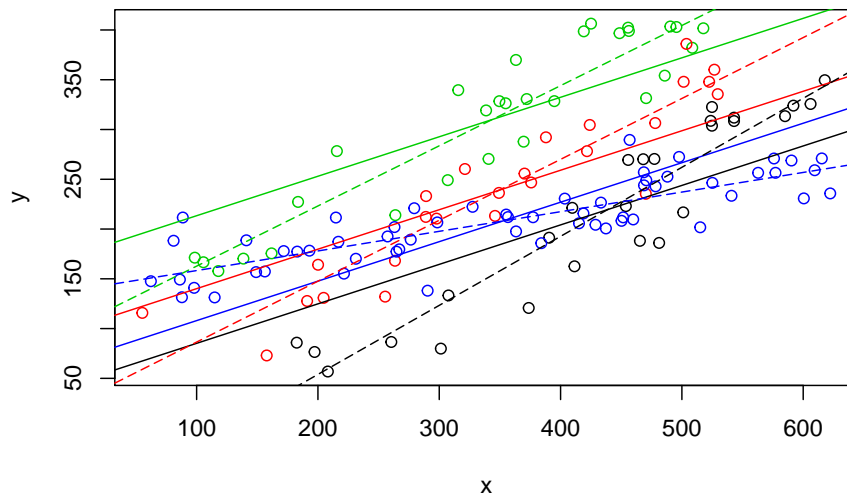
We noted the plot was messy, including the pooled regression (the thick black line), the regressions fitted to independent data sets, one for each species (museums before!) (the solid lines), and the regressions resulting from the model with species as a factor covariate (dotted-dashed lines).

```
#plot all the data
plot(y~x,col=as.numeric(as.factor(z)),data=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(lmIinesG,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)
```



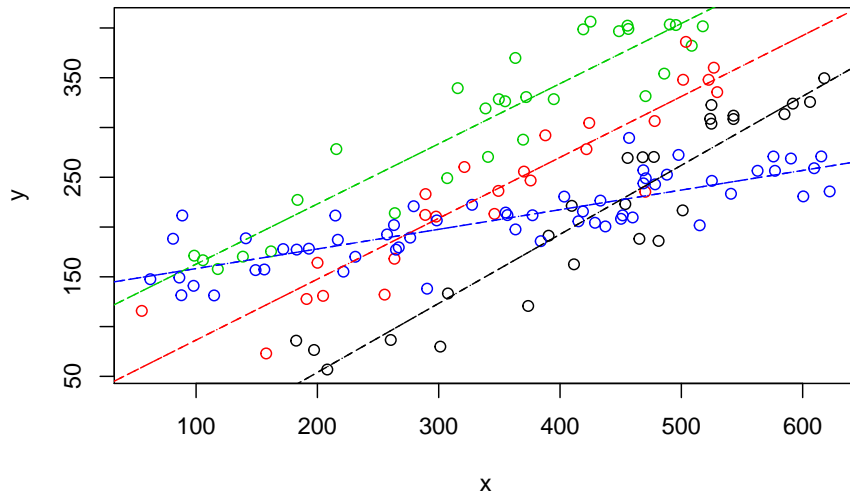
Now we can remove the independent lines (to clean it up a bit!) and just leave the no interaction model estimated values, then add the estimated lines from the interaction model below. Remember, AIC says it's the ones from the interaction model that are the best representation of the data.

```
#plot all the data
plot(y~x,col=as.numeric(as.factor(z)),data=data4lines,pch=1)
# #completely independent 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)
# 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]+lmlinesI$coefficients[4],col=2,lty=5)
abline(lmlinesI$coefficients[1]+lmlinesI$coefficients[4],lmlinesI$coefficients[2]+lmlinesI$coefficients[5],col=3,lty=5)
abline(lmlinesI$coefficients[1]+lmlinesI$coefficients[5],lmlinesI$coefficients[2]+lmlinesI$coefficients[6],col=4,lty=5)
```



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=data4lines,pch=1)
#completely independent 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)
# 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 exactly 8 parameters to describe the data... it's the same thing!!! Linear models are cool :)

8.2 Conclusion

The material in this and the last 3 lectures allows you to fully understand the outputs of simple regression models, and to see how some statistical models that you know from other names are just 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 9

Final Words

We have finished a nice book.

Chapter 10

Agradecimentos

A todos os meus alunos de Modelação Ecológica 2020/2021 que ajudaram com comentários a tornar este documento um pouco mais fácil de ler :)

Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2020). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.20.