WORKING ON CRUTEM LESSON 3-4

Victor Nyabuti Ong'era

2022-10-18

Lesson 1 & 2 code

```
path1 = "C:/Users/VICTOR_NYABUTI/Climate/data/"
values = as.matrix(read.table(paste(path1,"GL.csv", sep=""), sep = ",",dec =
                                ".")) 
missing_values = which(values==-9.999)
values[missing_values] =  NA
n_column = length(values[1,])
n_rows   = length(values[,1])
values = values[, -n_column]
even_row = seq(2,n_rows, 2)
odd_row = seq(1,n_rows, 2)
temp = values[even_row,]
perc = values[odd_row,]
temp  = temp[-(1:7),]
temp  = temp[-length(temp[,1]),]
perc  = perc[-(1:7),]
perc  = perc[-length(perc[,1]),]
colnames(temp) = c("Year", "January" , "February", "March", "April", "May",
                   "June", "July","Aug", "Sep","Oct", "Nov", "December")
colnames(temp)  =   c("Year", "January" , "February", "March", "April", "May",
                      "June", "July","Aug", "Sep","Oct", "Nov", "December")
path2 = "C:/Users/VICTOR_NYABUTI/Climate/output/"
write.table(temp,paste(path2,"Temparature_anomally.csv", sep = ""), sep = ",",
            col.names = TRUE, row.names = FALSE, quote =FALSE )
start_year = temp[1,1]
length(temp[,1])
```

```
## [1] 165
```

```
end_year = temp[length(temp[,1]),1]
n_year <- end_year-start_year+1
annual_mean_1 <- matrix(ncol=2,nrow=n_year)
colnames(annual_mean_1) <- c("Year","Annual_mean")
annual_mean_1[,1] <- start_year:end_year
for(i in 1:n_year){
  annual_mean_1[i,2] <- mean(temp[i,2:13],na.rm=FALSE)
}
write.table(annual_mean_1,paste(path2,"Yearly_average_temperature_anomaly.csv",
                                sep=""),sep=";",col.names=TRUE,row.names=FALSE,
```

```
             quote=FALSE)
annual_mean <- as.data.frame(annual_mean_1)
```

Lesson 3

In this lesson we covered a couple of intresting stuff. First we met the concept of filtering. Its part of the big topic digital signal processing or DSP. Forget what zou know about signal. Signal here means anything that carries data.Two main types of signals are contionous or discrete. Filters: allow you to remove out specific portions of a signal at once, or allow you to remove noise from certain signals.I can not inser images here sadly but when zou think of a noisz graph think of a rugges graph and a filtered one think of a smooth graph check this link @DAmato2007

```
#https://www.researchgate.net/figure/Gaussian-Noise-and-Filtered-Noise_fig2_363888849
```

We used *Moving averages* to remove the noise

Moving average (rolling average or running average) is a calculation to analyze data points by creating a series of averages of different subsets of the full data set. It is also called a moving mean (MM)[1] or rolling mean and is a type of finite impulse response filter. Variations include: simple, cumulative, or weighted form.

```
#https://en.wikipedia.org/wiki/Moving_average
```

Now lets calculate moving avaerages

first create an emptz matrix that will store the moving average

```
moving_average <- matrix(ncol=4,nrow=n_year)
```

lets check it

```
head(moving_average)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   NA   NA   NA   NA
## [2,]   NA   NA   NA   NA
## [3,]   NA   NA   NA   NA
## [4,]   NA   NA   NA   NA
## [5,]   NA   NA   NA   NA
## [6,]   NA   NA   NA   NA
```

We are going to calculate this averages in what we call windows the first window will be 5 years, next 20 lastly 50

Name the columns

```
colnames(moving_average) = c("Year","Window_1","Window_2","Window_3")
```

check it

```
head(moving_average)
```

```
##      Year Window_1 Window_2 Window_3
## [1,]   NA       NA       NA       NA
## [2,]   NA       NA       NA       NA
## [3,]   NA       NA       NA       NA
## [4,]   NA       NA       NA       NA
## [5,]   NA       NA       NA       NA
## [6,]   NA       NA       NA       NA
```

we know the years, so letes insert them in the first column

```
moving_average[,1] <- start_year:end_year
```

check it

```
tail.matrix(moving_average)
```

```
##        Year Window_1 Window_2 Window_3
## [160,] 2016       NA       NA       NA
## [161,] 2017       NA       NA       NA
## [162,] 2018       NA       NA       NA
## [163,] 2019       NA       NA       NA
## [164,] 2020       NA       NA       NA
## [165,] 2021       NA       NA       NA
```

Now lets calculate the windows. what are this windows exactly. So imagine you have data and you have put data scattered all over the paper you can connect this points with a curve

Remember we are working with means. So we want a mean in the first period of 5 years. Another mean at a period of 11 years and so on. This means will be like our reference points to get a curve that sort of takes care of the means in this periods/windows.

Now how would you make sure that the mean you choose will be representertive oy the window? The best way is to selct the median year. The middle year in a period or a window is right in the centre and dicided the years before and those after equally.

If this analogy is clear then let me introduce how to get a middle year. So if you remember how to get median which was the (total number of values + 1)/2.

Example if we have data set

```
x = c(1,2,3,4,5)
median(x)
```

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

which is the number of the values i.e (5 +1) /2

Okay now lets go back to our lesson.

```
window_1 = 5
```

New variable *window_1* with a value of 1

The fist year where we can calculate the moving average is the one in the middle.

```
start_1 = ((window_1-1)/2)+1
```

A new variable *start_1* that takes the value of variable *window_1* which is 5. Then subract one divided with 2 and added 1 at the end. Please note its just a different way to calculate mode. our previous method could apply too. check it

```
start_1
```

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

```
start1 = ((window_1+1)/2)
start1
```

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

nOW lest define end value

```
end_1 = n_year-((window_1-1)/2)
```

So we are telling R for the variable *end_1* the value is the value of our median subtrated from the numbe rof years

Now we are gonna use if. remeber about running repetitve tasks . we use loops

```
for(i in start_1:end_1){
  moving_average[i,"Window_1"] <- mean(annual_mean[(i-((window_1-1)/2)):(i+((window_1-1)/2)),"Annual_mea
```

we are telling R fo every value in the range *start_1* to *end_1* assighn values to our matrix *moving_average* which has i rows(all values in the range *start_1* to *end_1*) and columns *window_1*

The row values will come from the mean of *annual_mean* but just values between *i-our-median* to the values of *i+our+median*

The column v alues will be the *Annual_means*

lets check what happened to our matrix

```
head(moving_average)
```

```
##      Year Window_1 Window_2 Window_3
## [1,] 1857       NA       NA       NA
## [2,] 1858       NA       NA       NA
## [3,] 1859 27.01667       NA       NA
## [4,] 1860 23.98333       NA       NA
## [5,] 1861 21.20000       NA       NA
## [6,] 1862 18.56667       NA       NA
```

you can see that our values start to be filled from row 3

```
tail(moving_average)
```

```
##        Year Window_1 Window_2 Window_3
## [160,] 2016 84.61667       NA       NA
## [161,] 2017 84.63333       NA       NA
## [162,] 2018 84.38333       NA       NA
## [163,] 2019 83.63333       NA       NA
## [164,] 2020       NA       NA       NA
## [165,] 2021       NA       NA       NA
```

and that stops two rows before

lets do the same for window 2 i.e 11 years and window 3 20 years

```
window_2 <- 11
start_2 <- ((window_2-1)/2)+1 #First row where I can calculate the moving average
end_2 <- n_year-((window_2-1)/2) #Last row where I can calculate the moving average
for(i in start_2:end_2){
  moving_average[i,"Window_2"] <- mean(annual_mean[(i-((window_2-1)/2)):(i+((window_2-1)/2)),"Annual_mea
}
```

```
head(moving_average)
```

```
##      Year Window_1 Window_2 Window_3
## [1,] 1857       NA       NA       NA
## [2,] 1858       NA       NA       NA
## [3,] 1859 27.01667       NA       NA
## [4,] 1860 23.98333       NA       NA
## [5,] 1861 21.20000       NA       NA
## [6,] 1862 18.56667 21.07576       NA
```

Now that starts at row 6.I think we get the hang of it

```
window_3 <- 21
start_3 <- ((window_3-1)/2)+1 #First row where I can calculate the moving average
end_3 <- n_year-((window_3-1)/2) #Last row where I can calculate the moving average

for(i in start_3:end_3){
  moving_average[i,"Window_3"] <- mean(annual_mean[(i-((window_3-1)/2)):(i+((window_3-1)/2)),"Annual_mea
}
```

now if you want we can export this amazing data into a table

```
write.table(moving_average,paste(path2,"Moving_averages.csv",sep=""),sep=";",col.names=TRUE,row.names=F
```

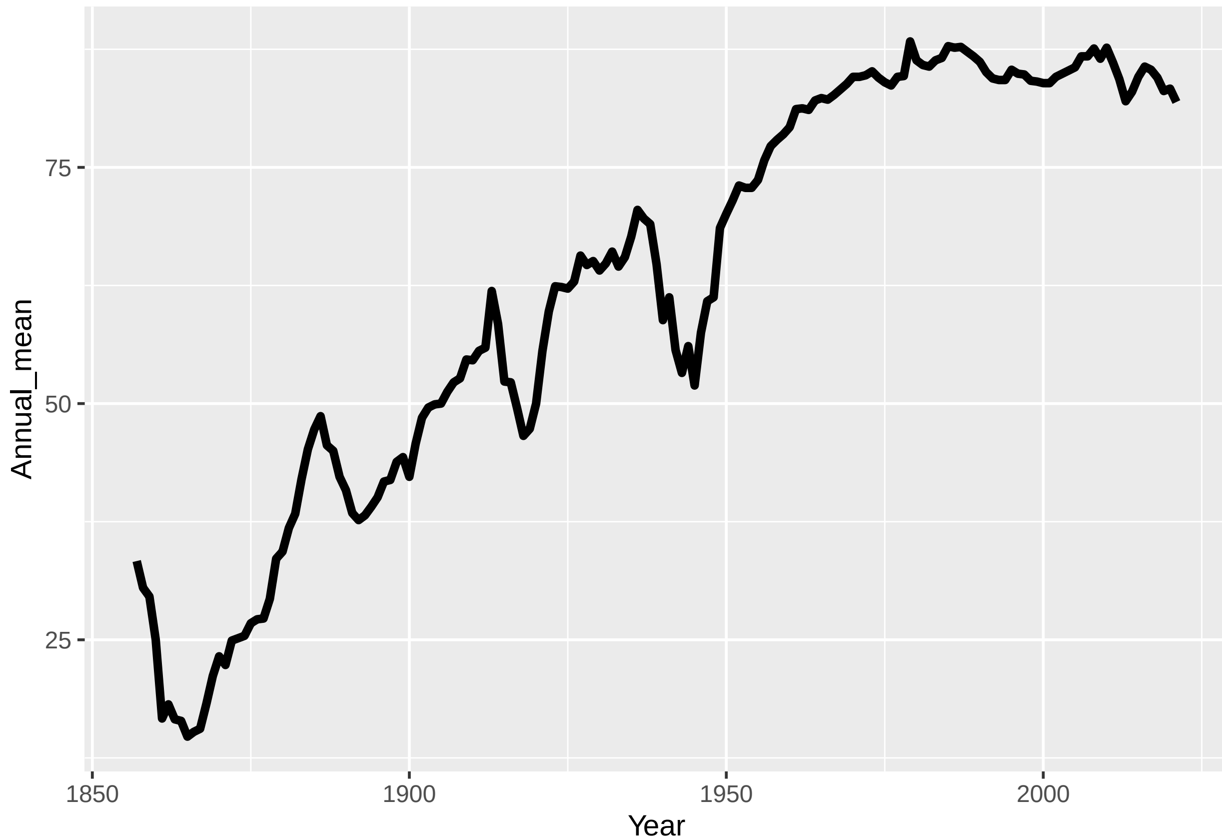
lets plot what we just did we will use ggplot that means our data has toi be in a data frame

```
moving_average <- as.data.frame(moving_average)
```

we save the same data with same name but as a data frame

To really see what this moving averages do to the graph we need to plot first just the graph of means we had the first lesson

```
library(ggplot2)
ggplot()+ #I create a grey paper
geom_line(data=annual_mean,aes(x=Year,y=Annual_mean),size=1.5,col="black")
```



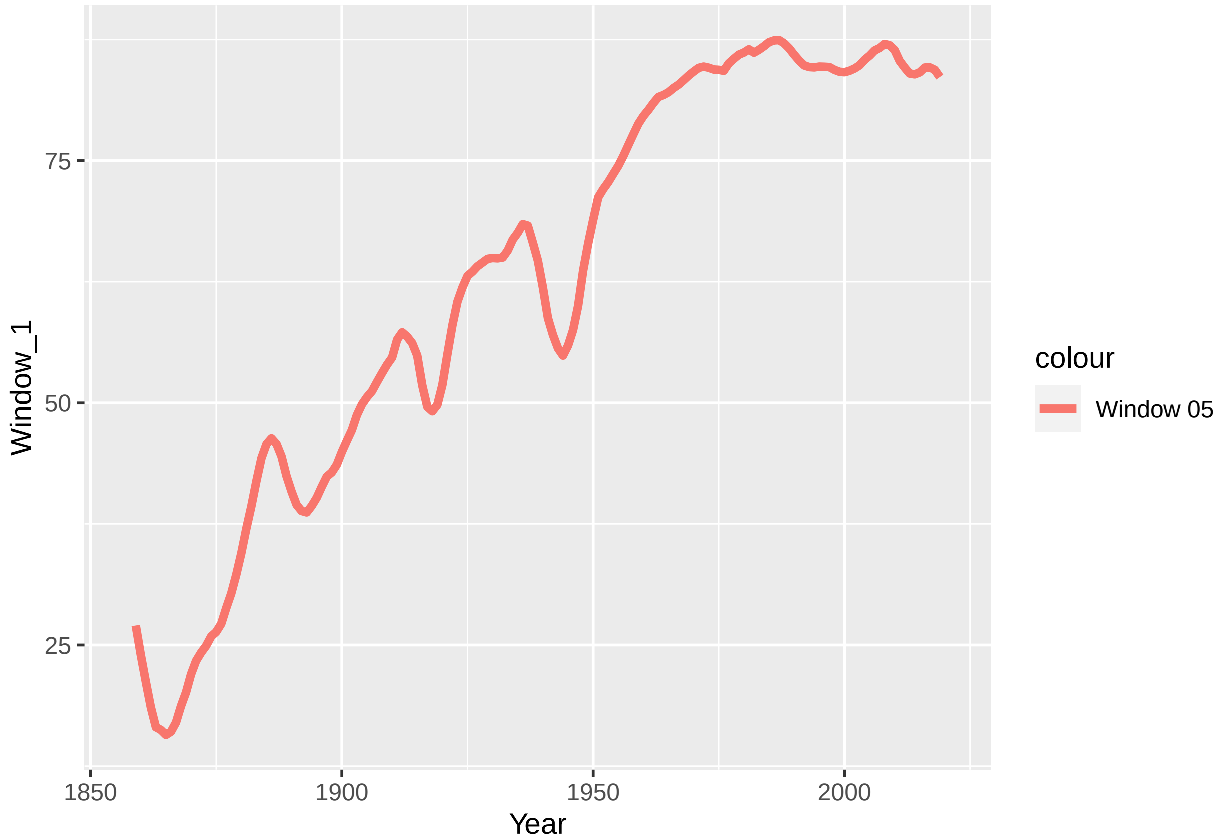
we use *geom_line* from ggplot2 to plot. remeber that we need to start with ggplot() and the + is an intergral part. its position must where its now.

In this past code its quite self explanatory. we tell R to plot using *geom_line* from our data *moving_average* and for aesthetics i.e *aes* let the x label be year and y label mean the size of out plot is 1,5 but that can change really as u wish and the col black

when writing strings or text in R you must wrap them in quaotation marks" "

```
library(ggplot2)
ggplot()+
geom_line(data=moving_average,aes(x=Year,y=Window_1,colour="Window 05"),size=1.5)
```

```
## Warning: Removed 4 row(s) containing missing values (geom_path).
```



i ploted this diffeently separately so that you can see the difference. The moving average makes the graphing look smoother compared to just the real values

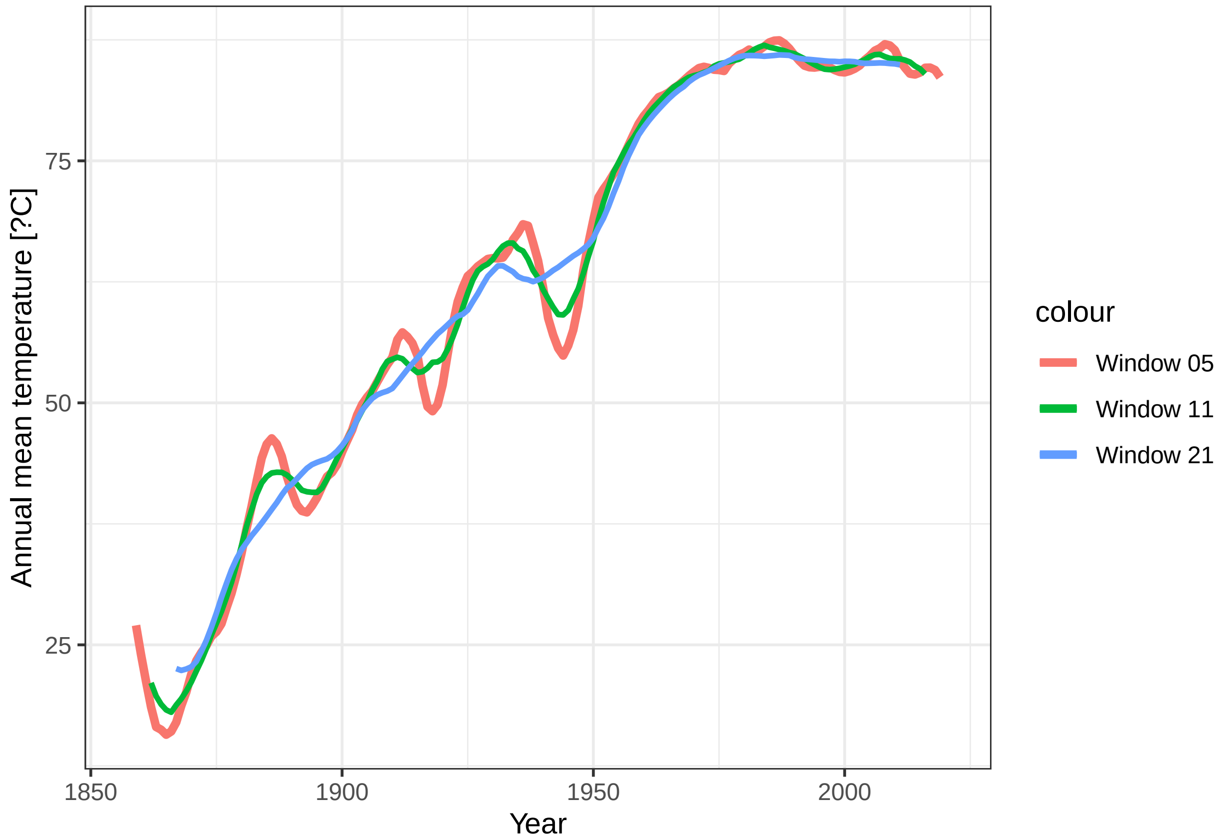
Now lets plot all graphs together

```
library(ggplot2)
ggplot()+
geom_line(data=moving_average,aes(x=Year,y=Window_1,colour="Window 05"),size=1.5)+
geom_line(data=moving_average,aes(x=Year,y=Window_2,colour="Window 11"),size=1)+
geom_line(data=moving_average,aes(x=Year,y=Window_3,colour="Window 21"),size=1)+
ylab("Annual mean temperature [?C]")+ #relabels y axis
theme_bw()#changes the background from gray to white
```

```
## Warning: Removed 4 row(s) containing missing values (geom_path).

## Warning: Removed 10 row(s) containing missing values (geom_path).

## Warning: Removed 20 row(s) containing missing values (geom_path).
```



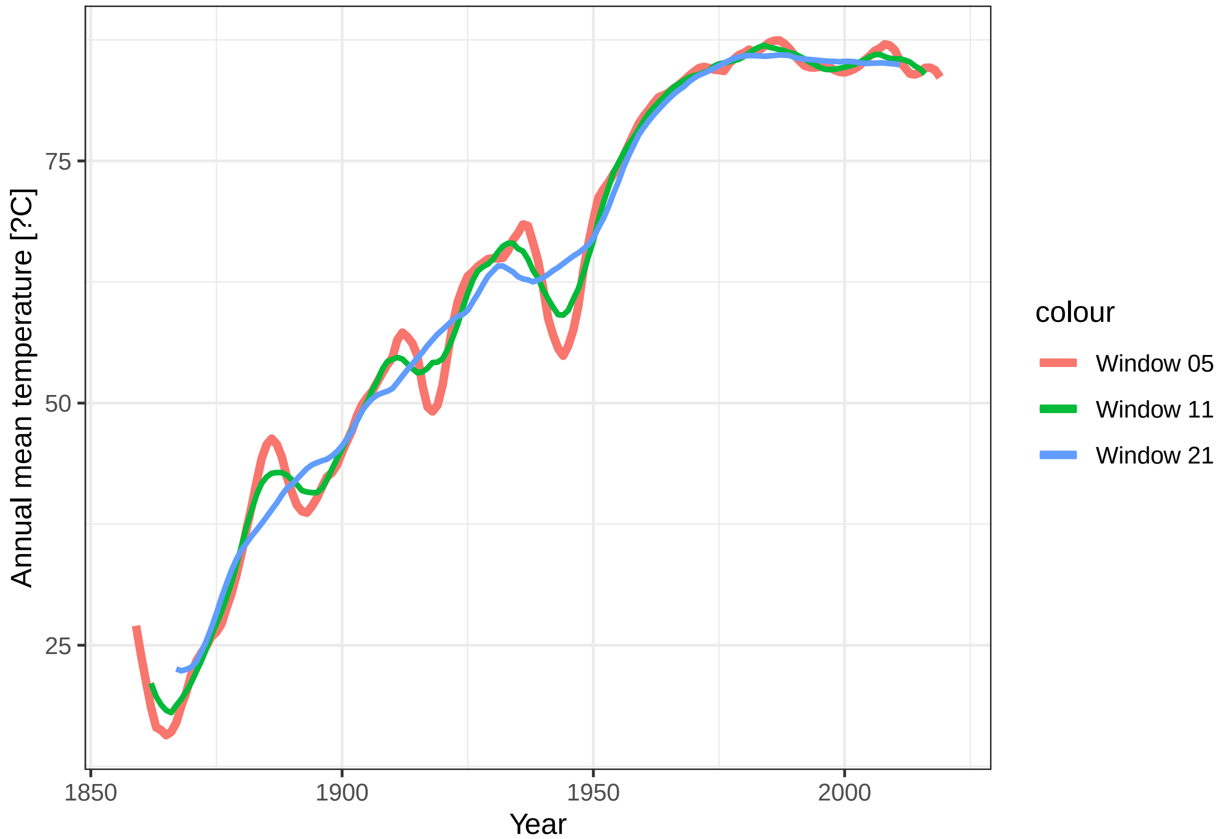
we can increase the label axes size

```
library(ggplot2)
ggplot()+
geom_line(data=moving_average,aes(x=Year,y=Window_1,colour="Window 05"),size=1.5)+
geom_line(data=moving_average,aes(x=Year,y=Window_2,colour="Window 11"),size=1)+
geom_line(data=moving_average,aes(x=Year,y=Window_3,colour="Window 21"),size=1)+
ylab("Annual mean temperature [?C]")+ #relabels y axis
theme_bw()#changes the background gray to white+
```

```
## Warning: Removed 4 row(s) containing missing values (geom_path).

## Warning: Removed 10 row(s) containing missing values (geom_path).

## Warning: Removed 20 row(s) containing missing values (geom_path).
```



You might need to read more on this package my knowledge is limited

Now lets do linear regression model. remember the equation of a straight line?

$$y = mx + c$$

where m is the slope and c the constant

```
y <- annual_mean[,"Annual_mean"]
x <- annual_mean[,"Year"]
```

we asign values for x and values for y

```
trend <- lm(y ~ x)
```

the trend

Now lets view the results

```
summary(trend)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -16.381  -5.447   1.533   5.978  11.222
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -792.48933   22.47915  -35.25   <2e-16 ***
## x              0.44077    0.01159   38.03   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.091 on 163 degrees of freedom
## Multiple R-squared:  0.8987, Adjusted R-squared:  0.8981
## F-statistic:  1446 on 1 and 163 DF,  p-value: < 2.2e-16
```

How do we read this

the **call**

this shows how R runs this model

the our *annual_mean* or **y** is the dependent and *year* or x is our indepedent

the **Residuals** is a difference between our actual values and dependent values

the **coefficients** so this is the main part actually. Lets undertand what we are actually trying to do. remeber that equation of a straight line. what the model does is to find a straight line that fits our mean temperatures in such a a way that it minimizes the distance between the mean temperatures and the line. Its from this that we get the coefficients

the **Estimate in COEFFICIENTS CAN US** in literally esimating the values. from our equation

the **std.Error** average amount that the coefficient estimates vary from the actual average value of our response variable

the **t value** how many standard deviations our coefficient estimate is far away from 0.

Now lets extract the values as numbers and save them.

```
slope <- as.numeric(coef(summary(trend))[, "Estimate"][2])
intercept <- as.numeric(coef(summary(trend))[, "Estimate"][1])
slope_error <- as.numeric(coef(summary(trend))[, "Std. Error"][2])
intercept_error <- as.numeric(coef(summary(trend))[, "Std. Error"][1])
significance <- as.numeric(coef(summary(trend))[, "Pr(>|t|)"][1])
```

so we save the summary as individual variables i.e as *slope, intercept, slope_error and significance* and for each we justuse the function **as.numeric** to convert them to number

to extract we tell r extract the coeff from summary of trend and then block blackets for coresponding column names. the location in the columns could be 2 or 1

Now we can save this results in a matrix ofcourse to export it as table later

```
trend_results <- matrix(ncol=2,nrow=5)
```

Empty matrix

```
head(trend_results)
```

```
##      [,1] [,2]
## [1,]   NA   NA
## [2,]   NA   NA
## [3,]   NA   NA
## [4,]   NA   NA
## [5,]   NA   NA
```

lets populate it

```
trend_results[1,1] <- "Slope"
trend_results[1,2] <- slope
trend_results[2,1] <- "Intercept"
trend_results[2,2] <- intercept
trend_results[3,1] <- "Slope_error"
trend_results[3,2] <- slope_error
trend_results[4,1] <- "Intercept_error"
trend_results[4,2] <- intercept_error
trend_results[5,1] <- "Significance"
trend_results[5,2] <- significance
```

now lets check our matrix

```
trend_results
```

```
##      [,1]              [,2]
## [1,] "Slope"           "0.440765278408535"
## [2,] "Intercept"       "-792.489329379603"
## [3,] "Slope_error"     "0.0115896707902544"
## [4,] "Intercept_error" "22.4791507068088"
## [5,] "Significance"    "3.60792446877293e-78"
```

Now export it to a table. I think we know how

Lesson 4

We did almost the same thing

```
window <- 51
start <- ((window-1)/2)+1 #First row where I can calculate the moving average
end <- n_year-((window-1)/2) #Last row where I can calculate the moving average
n_window <- end-start+1
```

The difference now is we are calculating for a **n_window** this means that instead of stopping at the end year, we add 1 year, running the circle again

```
trend_window <- matrix(ncol=7,nrow=n_window)
colnames(trend_window) <- c("Start_year","End_year", "Slope","Intercept","Slope_error","Intercept_error
```

Here we are creating a variable *trend window* itd gonna have a matrix with 7 columns and rows same as n window then we named the columns so that we can store all the informations we saw in the lesson 3 that we get from a regression model

after that we now need to run a for loop

```
ii <- 0
for(i in start:end){
  ii <- ii+1
  trend_window[ii,"Start_year"] <- annual_mean[(i-((window-1)/2)),"Year"]
  trend_window[ii,"End_year"] <- annual_mean[(i+((window-1)/2)),"Year"]
  y <- annual_mean[(i-((window-1)/2)):(i+((window-1)/2)),"Annual_mean"]
  x <- annual_mean[(i-((window-1)/2)):(i+((window-1)/2)),"Year"]
  trend <- lm(y ~ x)
  trend_window[ii,"Slope"] <- as.numeric(coef(summary(trend))[, "Estimate"][2])
  trend_window[ii,"Intercept"] <- as.numeric(coef(summary(trend))[, "Estimate"][1])
  trend_window[ii,"Slope_error"] <- as.numeric(coef(summary(trend))[, "Std. Error"][2])
  trend_window[ii,"Intercept_error"] <- as.numeric(coef(summary(trend))[, "Std. Error"][1])
  trend_window[ii,"Significance"] <- as.numeric(coef(summary(trend))[, "Pr(>|t|)"][1])
}
```

```
standard_deviation_anomaly <- sd(annual_mean[,2])

#I calculate the residuals of the anomaly from the low-pass-filter
residuals <- matrix(ncol=2,nrow=n_year)
colnames(residuals) <- c("Year","Residuals")
residuals[,"Year"] <- start_year:end_year
residuals[,"Residuals"] <- annual_mean[,"Annual_mean"]-moving_average[,"Window_2"]
standard_deviation_residuals <- sd(residuals[,"Residuals"],na.rm=TRUE)
```

```
ii <- 0
for(i in start:end){
  ii <- ii+1
  trend_window[ii,"Start_year"] <- annual_mean[(i-((window-1)/2)),"Year"]
  trend_window[ii,"End_year"] <- annual_mean[(i+((window-1)/2)),"Year"]
  y <- annual_mean[(i-((window-1)/2)):(i+((window-1)/2)),"Annual_mean"]
  x <- annual_mean[(i-((window-1)/2)):(i+((window-1)/2)),"Year"]
  trend <- lm(y ~ x)
  trend_window[ii,"Slope"] <- as.numeric(coef(summary(trend))[, "Estimate"][2])
  trend_window[ii,"Intercept"] <- as.numeric(coef(summary(trend))[, "Estimate"][1])
  trend_window[ii,"Slope_error"] <- as.numeric(coef(summary(trend))[, "Std. Error"][2])
  trend_window[ii,"Intercept_error"] <- as.numeric(coef(summary(trend))[, "Std. Error"][1])
  trend_window[ii,"Significance"] <- as.numeric(coef(summary(trend))[, "Pr(>|t|)"][1])
}
```