Empirical Project 1: Working in R code

Jean Gonzalez & Cristian Beltran

# Empirical Project 1 Working in R

These code downloads have been constructed as supplements to the full Doing Economics projects (<https://core-econ.org/doing-economics/>). You’ll need to download the data before running the code that follows.

## Part 1.1 The behaviour of average surface temperature over time

### R walk-through 1.1 Importing the datafile into R

We want to import the datafile called ‘NH.Ts+dSST.csv’ into R.

We start by setting our working directory using the setwd command. This command tells R where your datafiles are stored. In the code below, replace ‘YOURFILEPATH’ with the full filepath that indicates the folder in which you have saved the datafile. If you don’t know how to find the path to your working folder, see the ‘Technical Reference’ section (<https://tinyco.re/3407438>).

library(readxl)  
  
setwd("C:/Users/Marcos Gonzalez/OneDrive - Universidad del rosario/Escritorio/UR/Trabajos/Sexto Semestre/Haciendo Economia/Taller 3")

Since our data is in csv format, we use the read.csv function to import the data into R. We will call our file ‘tempdata’ (short for ‘temperature data’).

Here you can see commands to R which are spread across two lines. You can spread a command across multiple lines, but you must adhere to the following two rules for this to work. First, the line break should come inside a set of parenthesis (i.e. between ( and ) or straight after the assignment operator (<-). Second, the line break must not be inside a string (whatever is inside quotes) or in the middle of a word or number.

tempdata <- read.csv("NH.Ts+dSST.csv",  
 skip = 1, na.strings = "\*\*\*")

When using this function, we added two options. If you open the spreadsheet in Excel, you will see that the real data table only starts in Row 2, so we use the skip = 1 option to skip the first row when importing the data. When looking at the spreadsheet, you can see that missing temperature data is coded as "\*\*\*". In order for R to recognise the non-missing temperature data as numbers, we use the na.strings = "\*\*\*" option to indicate that missing observations in the spreadsheet are coded as "\*\*\*".

To check that the data has been imported correctly, you can use the head function to view the first six rows of the dataset, and confirm that they correspond to the columns in the csv file.

head(tempdata)

## Year Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec  
## 1 1880 -0.38 -0.52 -0.23 -0.30 -0.04 -0.18 -0.21 -0.25 -0.24 -0.29 -0.43 -0.41  
## 2 1881 -0.30 -0.24 -0.05 -0.02 0.05 -0.33 0.10 -0.05 -0.27 -0.44 -0.36 -0.23  
## 3 1882 0.26 0.22 0.03 -0.29 -0.22 -0.28 -0.28 -0.14 -0.24 -0.51 -0.33 -0.67  
## 4 1883 -0.57 -0.65 -0.14 -0.29 -0.25 -0.11 -0.05 -0.22 -0.33 -0.15 -0.43 -0.14  
## 5 1884 -0.15 -0.10 -0.63 -0.58 -0.35 -0.40 -0.40 -0.50 -0.44 -0.43 -0.57 -0.46  
## 6 1885 -1.00 -0.45 -0.23 -0.48 -0.57 -0.44 -0.33 -0.40 -0.38 -0.36 -0.37 -0.10  
## J.D D.N DJF MAM JJA SON  
## 1 -0.29 NA NA -0.19 -0.21 -0.32  
## 2 -0.18 -0.19 -0.32 -0.01 -0.09 -0.36  
## 3 -0.20 -0.17 0.08 -0.16 -0.23 -0.36  
## 4 -0.28 -0.32 -0.63 -0.23 -0.12 -0.31  
## 5 -0.42 -0.39 -0.13 -0.52 -0.44 -0.48  
## 6 -0.43 -0.46 -0.64 -0.42 -0.39 -0.37

Before working with the important data, we use the str function to check that the data is formatted correctly.

str(tempdata)

## 'data.frame': 145 obs. of 19 variables:  
## $ Year: int 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889 ...  
## $ Jan : num -0.38 -0.3 0.26 -0.57 -0.15 -1 -0.73 -1.08 -0.48 -0.27 ...  
## $ Feb : num -0.52 -0.24 0.22 -0.65 -0.1 -0.45 -0.83 -0.7 -0.61 0.3 ...  
## $ Mar : num -0.23 -0.05 0.03 -0.14 -0.63 -0.23 -0.7 -0.44 -0.63 -0.01 ...  
## $ Apr : num -0.3 -0.02 -0.29 -0.29 -0.58 -0.48 -0.36 -0.38 -0.21 0.17 ...  
## $ May : num -0.04 0.05 -0.22 -0.25 -0.35 -0.57 -0.33 -0.25 -0.14 -0.03 ...  
## $ Jun : num -0.18 -0.33 -0.28 -0.11 -0.4 -0.44 -0.37 -0.2 -0.02 -0.06 ...  
## $ Jul : num -0.21 0.1 -0.28 -0.05 -0.4 -0.33 -0.14 -0.24 0 -0.08 ...  
## $ Aug : num -0.25 -0.05 -0.14 -0.22 -0.5 -0.4 -0.42 -0.55 -0.21 -0.2 ...  
## $ Sep : num -0.24 -0.27 -0.24 -0.33 -0.44 -0.38 -0.32 -0.2 -0.19 -0.29 ...  
## $ Oct : num -0.29 -0.44 -0.51 -0.15 -0.43 -0.36 -0.31 -0.49 -0.03 -0.41 ...  
## $ Nov : num -0.43 -0.36 -0.33 -0.43 -0.57 -0.37 -0.4 -0.27 0 -0.61 ...  
## $ Dec : num -0.41 -0.23 -0.67 -0.14 -0.46 -0.1 -0.21 -0.42 -0.23 -0.54 ...  
## $ J.D : num -0.29 -0.18 -0.2 -0.28 -0.42 -0.43 -0.43 -0.43 -0.23 -0.17 ...  
## $ D.N : num NA -0.19 -0.17 -0.32 -0.39 -0.46 -0.42 -0.42 -0.25 -0.14 ...  
## $ DJF : num NA -0.32 0.08 -0.63 -0.13 -0.64 -0.55 -0.66 -0.5 -0.07 ...  
## $ MAM : num -0.19 -0.01 -0.16 -0.23 -0.52 -0.42 -0.46 -0.35 -0.33 0.04 ...  
## $ JJA : num -0.21 -0.09 -0.23 -0.12 -0.44 -0.39 -0.31 -0.33 -0.07 -0.11 ...  
## $ SON : num -0.32 -0.36 -0.36 -0.31 -0.48 -0.37 -0.34 -0.32 -0.07 -0.44 ...

You can see that all variables are formatted as numerical data (num), so R correctly recognises that the data are numbers.

[End of walk-through]

### R walk-through 1.2 Drawing a line chart of temperature and time

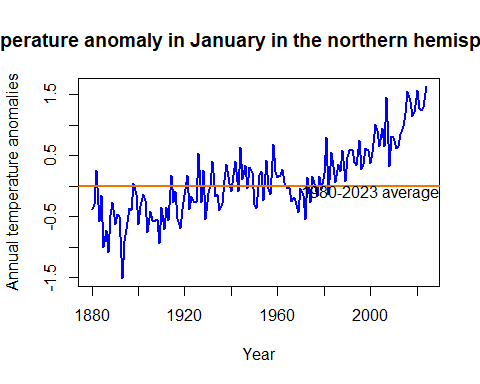
The data is formatted as numerical (num) data, so R recognises each variable as a series of numbers (instead of text), but does not recognise that these numbers correspond to the same variable for different time periods (known as ‘time series data’ in economics). Letting R know that we have time series data will make coding easier later (especially with making graphs). You can use the ts function to specify that a variable is a time series. Make sure to amend the code below so that the end year (end = c()) corresponds to the latest year in your dataset (our example uses 2017).

tempdata$Jan <- ts(tempdata$Jan,   
 start = c(1880), end = c(2024), frequency = 1)   
tempdata$DJF <- ts(tempdata$DJF,   
 start = c(1880), end = c(2024), frequency = 1)   
tempdata$MAM <- ts(tempdata$MAM,   
 start = c(1880), end = c(2024), frequency = 1)   
tempdata$JJA <- ts(tempdata$JJA,   
 start = c(1880), end = c(2024), frequency = 1)   
tempdata$SON <- ts(tempdata$SON,   
 start = c(1880), end = c(2024), frequency = 1)   
tempdata$J.D <- ts(tempdata$J.D,   
 start = c(1880), end = c(2024), frequency = 1)

Note that we placed each of these quarterly series in the relevant middle month. You could do the same for the remaining series, but we will only use the series above in this R walk-through.

We can now use these variables to draw line charts using the plot function. As an example, we will draw a line chart using data for January (tempdata$Jan) for the years 1880–2023. The title option on the next line adds a chart title, and the abline option draws a horizontal line according to our specifications. Make sure to amend the code below so that your chart title corresponds to the latest year in your dataset (our example uses 2016).

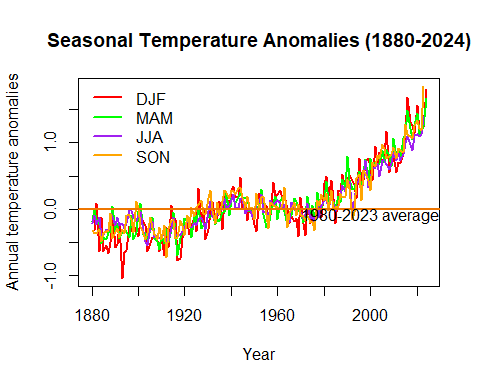
# Set line width and colour  
plot(tempdata$Jan, type = "l", col = "blue", lwd = 2,  
 ylab = "Annual temperature anomalies", xlab = "Year")  
  
# Add a title  
title("Average temperature anomaly in January in the northern hemisphere (1880-2016)")  
  
# Add a horizontal line (at y = 0)  
abline(h = 0, col = "darkorange2", lwd = 2)  
  
# Add a label to the horizontal line  
text(2000, -0.1, "1980-2023 average")



Try different values for type and col in the plot function to figure out what these options do (some online research could help). xlab and ylab define the respective axis titles.

It is important to remember that all axis and chart titles should be enclosed in quotation marks (""), as well as any words that are not options (for example, colour names or filenames).

# Crea un vector con los años correspondientes  
years <- 1880:2024  
  
# Inicia el gráfico con DJF usando los años como eje X  
plot(years, tempdata$DJF, type = "l", col = "red", lwd = 2,  
 ylab = "Annual temperature anomalies", xlab = "Year",  
 main = "Seasonal Temperature Anomalies (1880-2024)",  
 ylim = range(c(tempdata$DJF, tempdata$MAM, tempdata$JJA, tempdata$SON), finite = TRUE))  
  
# Agrega las otras líneas al mismo gráfico  
lines(years, tempdata$MAM, col = "green", lwd = 2)  
lines(years, tempdata$JJA, col = "purple", lwd = 2)  
lines(years, tempdata$SON, col = "orange", lwd = 2)  
  
# Añade una línea horizontal en y = 0  
abline(h = 0, col = "darkorange2", lwd = 2)  
  
# Añade una etiqueta a la línea horizontal  
text(2000, -0.1, "1980-2023 average")  
  
# Añade una leyenda para diferenciar las líneas  
legend("topleft", legend = c("DJF", "MAM", "JJA", "SON"),  
 col = c("red", "green", "purple", "orange"),  
 lwd = 2, bty = "n")

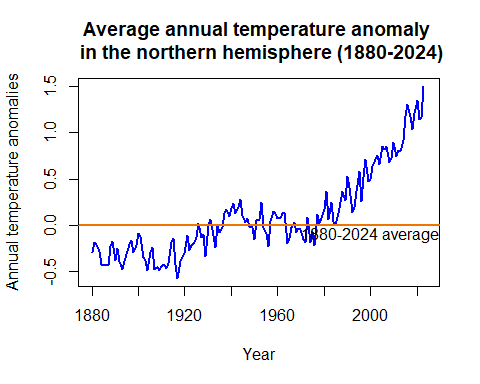


[End of walk-through]

### R walk-through 1.3 Producing a line chart for the annual temperature anomalies

This is where the power of programming languages becomes evident: to produce the same line chart for a different variable, we simply take the code used in R walk-through 1.2 and replace the variable name Jan with the name for the annual variable (J.D). Again, make sure to amend the code so that your chart title corresponds to the latest year in your data (our example uses 2016).

# Set line width and colour  
plot(tempdata$J.D, type = "l", col = "blue", lwd = 2,  
 ylab = "Annual temperature anomalies", xlab = "Year")  
  
# \n creates a line break  
title("Average annual temperature anomaly \n in the northern hemisphere (1880-2024)")  
  
# Add a horizontal line (at y = 0)  
abline(h = 0, col = "darkorange2", lwd = 2)  
  
# Add a label to the horizontal line  
text(2000, -0.1, "1880-2024 average")



[End of walk-through]

## Part 1.2 Variation in temperature over time

### R walk-through 1.4 Creating frequency tables and histograms

Since we will be looking at data from different subperiods (year intervals) separately, we will create a categorical variable (a variable that has two or more categories) that indicates the subperiod for each observation (row). In R this type of variable is called a ‘factor variable’. When we create a factor variable, we need to define the categories that this variable can take.

tempdata$Period <-   
 factor(NA, levels =   
 c("1921-1950", "1951-1980", "1981-2010"),   
 ordered = TRUE)

We created a new variable called Period and defined the possible categories (which R refers to as ‘levels’). Since we will not be using data for some years (before 1921 and after 2010), we want Period to take the value ‘NA’ (not available) for these observations (rows), and the appropriate category for all the other observations (between 1921–2010). One way to do this is by defining Period as ‘NA’ for all observations, then change the values of Period for the observations in 1921–2010.

tempdata$Period[(tempdata$Year > 1920) & (tempdata$Year < 1951)] <- "1921-1950"  
tempdata$Period[(tempdata$Year > 1950) &  
 (tempdata$Year < 1981)] <- "1951-1980"  
  
tempdata$Period[(tempdata$Year > 1980) &  
 (tempdata$Year < 2011)] <- "1981-2010"

We need to use all monthly anomalies from June, July, and August, but they are currently in three separate columns. We will use the c (combine) function to create one new variable (called temp\_summer) that contains all these values.

# Combine the temperature data for June, July, and August  
temp\_summer <- c(tempdata$Jun, tempdata$Jul, tempdata$Aug)

Now we have one long variable (temp\_summer), with the monthly temperature anomalies for the three months (from 1880 to the latest year) attached to each other. But remember that we want to make separate calculations for each category in Period (1921–1950, 1951–1980, 1981–2010). To make a variable showing the categories for the temp\_summer variable, we use the c function again.

temp\_summer <- unlist(tempdata[,7:9],use.names = FALSE)  
# Mirror the Period information for temp\_sum  
temp\_Period <-   
c(tempdata$Period, tempdata$Period, tempdata$Period)

After using the c function, we had to use the factor function again to tell R that our new variable temp\_Period is a factor variable.

We have now created the variables needed to make frequency tables and histograms (temp\_summer and temp\_Period). To obtain the frequency table for 1951–1980, we use the hist function on the monthly temperature anomalies from the period ‘1951–1980’: temp\_summer[(temp\_Period == "1951-1980")]. The option plot = FALSE tells R not to make a plot of this information. (See what happens if you set it to TRUE.)

hist(temp\_summer[(temp\_Period == "1951-1980")],   
 plot = FALSE)

## $breaks  
## [1] -0.30 -0.25 -0.20 -0.15 -0.10 -0.05 0.00 0.05 0.10 0.15 0.20 0.25  
##   
## $counts  
## [1] 2 7 3 12 11 7 13 14 10 7 4  
##   
## $density  
## [1] 0.4444444 1.5555556 0.6666667 2.6666667 2.4444444 1.5555556 2.8888889  
## [8] 3.1111111 2.2222222 1.5555556 0.8888889  
##   
## $mids  
## [1] -0.275 -0.225 -0.175 -0.125 -0.075 -0.025 0.025 0.075 0.125 0.175  
## [11] 0.225  
##   
## $xname  
## [1] "temp\_summer[(temp\_Period == \"1951-1980\")]"  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"

From the output you can see that we can get the temperature ranges (the values in $breaks correspond to Column 1 of Figure 1.5) and the frequencies ($counts), which is all we need to create a frequency table. However, in our case the frequency table is merely a temporary input required to produce a histogram.

We can make the three histograms we need all at once, using the histogram function from the mosaic package.

The function below includes multiple commands:

* | temp\_Period splits the data according to its category, given by temp\_Period.
* type = "count" indicates that we want to display the counts (frequencies) in each category.
* breaks = seq(-0.5, 1.3, 0.1) gives a sequence of numbers −0.5, −0.4, …, 1.3, which are boundaries for the categories.
* main = "Histogram of temperature anomalies" gives Figure 1.6 its title.

# Load the library we use for the following command.  
library(mosaic)

## Registered S3 method overwritten by 'mosaic':  
## method from   
## fortify.SpatialPolygonsDataFrame ggplot2

##   
## The 'mosaic' package masks several functions from core packages in order to add   
## additional features. The original behavior of these functions should not be affected by this.

##   
## Attaching package: 'mosaic'

## The following objects are masked from 'package:dplyr':  
##   
## count, do, tally

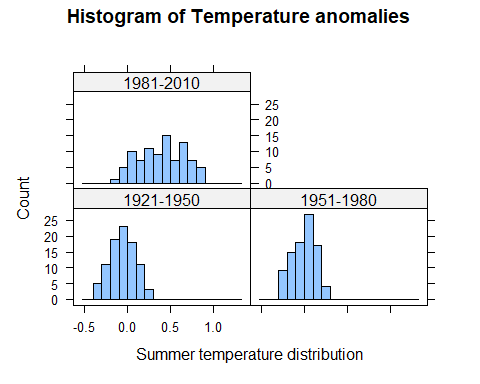
## The following object is masked from 'package:Matrix':  
##   
## mean

## The following object is masked from 'package:ggplot2':  
##   
## stat

## The following objects are masked from 'package:stats':  
##   
## binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,  
## quantile, sd, t.test, var

## The following objects are masked from 'package:base':  
##   
## max, mean, min, prod, range, sample, sum

histogram(~ temp\_summer | temp\_Period, type = "count",   
 breaks = seq(-0.5, 1.3, 0.10),   
 main = "Histogram of Temperature anomalies",   
 xlab = "Summer temperature distribution")



### R walk-through 1.5 Using the quantile function

First, we need to create a variable that contains all monthly anomalies in the years 1951–1980. Then, we use R’s quantile function to find the required percentiles (0.3 and 0.7 refer to the 3rd and 7th deciles, respectively).

*Note*: You may get slightly different values to those shown here if you are using the latest data.

# Select years 1951 to 1980  
temp\_all\_months <- subset(tempdata,   
 (Year >= 1951 & Year <= 1980))  
   
# Columns 2 to 13 contain months Jan to Dec.  
temp\_51to80 <- unlist(temp\_all\_months[, 2:13])  
   
# c(0.3, 0.7) indicates the chosen percentiles.  
perc <- quantile(temp\_51to80, c(0.3, 0.7))   
  
# The cold threshold  
p30 <- perc[1]  
p30

## 30%   
## -0.1

# The hot threshold  
p70 <- perc[2]  
p70

## 70%   
## 0.1

Taking as a reference the temperatures from 1951 to 1980 and considering that normal temperatures range between the third and seventh deciles (-0.1 and 0.1 standard deviations from the mean, respectively), the following statements can be made:  
- From 1921 to 1950, the deviations from the mean correspond to cold temperatures.  
- From 1951 to 1980, the deviations are mostly close to the mean.  
- In more recent years, from 1981 to 2010, the majority of temperatures are in the high range, reflecting the phenomenon of global warming.  
- The range of temperatures increases over time.  
[End of walk-through]

### R walk-through 1.6 Using the mean function

*Note*: You may get slightly different values to those shown here if you are using the latest data.

We repeat the steps used in R walk-through 1.5, now looking at monthly anomalies in the years 1981–2010. We can simply change the year values in the code from R walk-through 1.5.

# Select years 1951 to 1980  
temp\_all\_months <- subset(tempdata,   
 (Year >= 1981 & Year <= 2010))  
   
# Columns 2 to 13 contain months Jan to Dec.  
temp\_81to10 <- unlist(temp\_all\_months[, 2:13])

Now that we have all the monthly data for 1981–2010, we want to count the proportion of observations that are smaller than –0.1. This is easily achieved with the following lines of code:

paste("Proportion smaller than p30")

## [1] "Proportion smaller than p30"

temp <- temp\_81to10 < p30  
mean(temp)

## [1] 0.01944444

Let’s check whether we get a similar result for the number of observations that are larger than 0.11.

paste("Proportion larger than p70")

## [1] "Proportion larger than p70"

mean(temp\_81to10 > p70)

## [1] 0.8472222

Of the total temperatures recorded between 1981 and 2010, 1.9% correspond to low temperatures (below the third decile), while 84.7% are classified as high temperatures (above the seventh decile).

[End of walk-through]

### R walk-through 1.7 Calculating and understanding mean and variance

Calculate mean and variance. One option is to use the mosaic package

paste("Mean of DJF temperature anomalies across periods")

## [1] "Mean of DJF temperature anomalies across periods"

mean(~DJF|Period,data = tempdata)

## 1921-1950 1951-1980 1981-2010   
## -0.030333333 -0.002666667 0.523333333

paste("Variance of DJF anomalies across periods")

## [1] "Variance of DJF anomalies across periods"

var(~DJF|Period,data = tempdata)

## 1921-1950 1951-1980 1981-2010   
## 0.05672057 0.05038575 0.07871264

Using the data in tempdata (data = tempdata), we calculated the mean (mean) and variance (var) of variable ~DJF separately for (|) each value of Period. The mosaic package allows us to calculate the means/variances for each period all at once. If mosaic is not loaded, you will get the error message: Error in mean(~DJF \| Period, data = tempdata) : unused argument (data = tempdata).

Observing the behavior of the mean over the years, we can see that its value increases as time passes. Meanwhile, the variance remains constant between the 1921-1950 and 1951-1980 blocks of years, but it increases between 1981 and 2010.

Let’s calculate the variances through the periods for the other seasons.

paste("Variance of MAM anomalies across periods")

## [1] "Variance of MAM anomalies across periods"

var(~MAM|Period,data = tempdata)

## 1921-1950 1951-1980 1981-2010   
## 0.03099782 0.02540000 0.07573345

paste("Variance of JJA anomalies across periods")

## [1] "Variance of JJA anomalies across periods"

var(~JJA|Period,data = tempdata)

## 1921-1950 1951-1980 1981-2010   
## 0.02128920 0.01460644 0.06749609

paste("Variance of SON anomalies across periods")

## [1] "Variance of SON anomalies across periods"

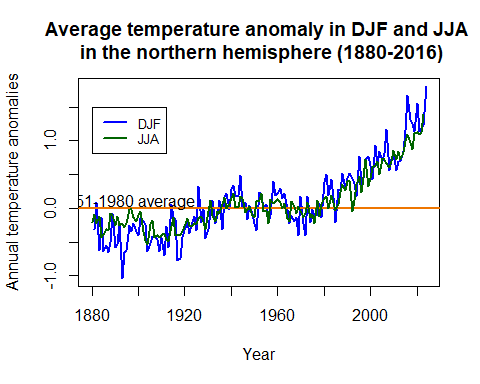
var(~SON|Period,data = tempdata)

## 1921-1950 1951-1980 1981-2010   
## 0.02819264 0.02635126 0.11104644

Observing the other seasons, we can see that the same trend in variance persists. From the first to the second block of years, the variance slightly decreases; however, in more recent years, the variance increases significantly.

We can plot a line chart to see these changes graphically. (This type of chart is formally known as a ‘time-series plot’). Make sure to change the chart title according to the latest year in your data (here we used 2016).

plot(tempdata$Year,tempdata$DJF, type = "l", col = "blue", lwd = 2,  
 ylab = "Annual temperature anomalies", xlab = "Year")  
  
# \n creates a line break  
title("Average temperature anomaly in DJF and JJA \n in the northern hemisphere (1880-2016)")  
  
# Add a horizontal line (at y = 0)  
abline(h = 0, col = "darkorange2", lwd = 2)  
lines(tempdata$Year, tempdata$JJA, col = "darkgreen", lwd = 2)   
  
# Add a label to the horizontal line  
text(1895, 0.1, "1951-1980 average")  
legend(1880, 1.5, legend = c("DJF", "JJA"),  
 col = c("blue", "darkgreen"),   
 lty = 1, cex = 0.8, lwd = 2)



[End of walk-through]

## Part 1.3 Carbon emissions and the environment

### R walk-through 1.8 Scatterplots and the correlation coefficient

First we will use the read.csv function to import the CO2 datafile into R, and call it CO2data.

CO2data <- read\_excel("CO2data.xlsx")

This file has monthly data, but in contrast to the data in tempdata, the data is all in one column (this is more conventional than the column per month format). To make this task easier, we will pick the June data from the CO2 emissions and add them as an additional variable to the tempdata dataset.

R has a convenient function called merge to do this. First we create a new dataset that contains only the June emissions data (‘CO2data\_june’).

CO2data\_june <- CO2data[CO2data$Month == 6,]

Then we use this data in the merge function. The merge function takes the original ‘tempdata’ and the ‘CO2data’ and merges (combines) them together. As the two dataframes have a common variable, Year, R automatically matches the data by year.

(*Extension:* Look up ?merge or Google ‘How to use the R merge function’ to figure out what all.x does, and to see other options that this function allows.)

names(CO2data)[1] <- "Year"  
tempCO2data <- merge(tempdata, CO2data\_june)

Let us have a look at the data and check that it was combined correctly:

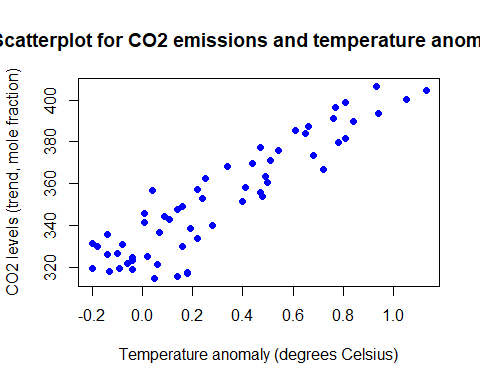
head(tempCO2data[, c("Year", "Jun", "Trend")])

## Year Jun Trend  
## 1 1958 0.05 314.85  
## 2 1959 0.14 315.92  
## 3 1960 0.18 317.36  
## 4 1961 0.18 317.48  
## 5 1962 -0.13 318.27  
## 6 1963 -0.04 319.16

## Year Jun Trend  
## 1 1958 0.04 314.85  
## 2 1959 0.14 315.92  
## 3 1960 0.18 317.36  
## 4 1961 0.19 317.48  
## 5 1962 -0.10 318.27  
## 6 1963 -0.02 319.16

To make a scatterplot, we use the plot function. R’s default chart for plot is a scatterplot, so we do not need to specify the chart type. One new option that applies to scatterplots is pch =, which determines the appearance of the data points. The number 16 corresponds to filled-in circles, but you can experiment with other numbers (from 0 to 25) to see what the data points look like.

plot(tempCO2data$Jun, tempCO2data$Trend,   
 xlab = "Temperature anomaly (degrees Celsius)",   
 ylab = "CO2 levels (trend, mole fraction)",   
 pch = 16, col = "blue")  
  
title("Scatterplot for CO2 emissions and temperature anomalies")



The cor function calculates the correlation coefficient. *Note*: You may get slightly different results if you are using the latest data.

cor(tempCO2data$Jun, tempCO2data$Trend)

## [1] 0.9149093

## [1] 0.9157744

As shown in both the graph and the correlation coefficient, the association between temperature and carbon dioxide emissions is clearly strongly positive.

One limitation of this correlation measure is that it only tells us about the strength of the upward- or downward-sloping linear relationship between two variables, in other words how closely the scatterplot aligns along an upward- or downward-sloping straight line. The correlation coefficient cannot tell us if the two variables have a different kind of relationship (such as that represented by a wavy line).

*Note:* The word ‘strong’ is used for coefficients that are close to 1 or −1, and ‘weak’ is used for coefficients that are close to 0, though there is no precise range of values that are considered ‘strong’ or ‘weak’.

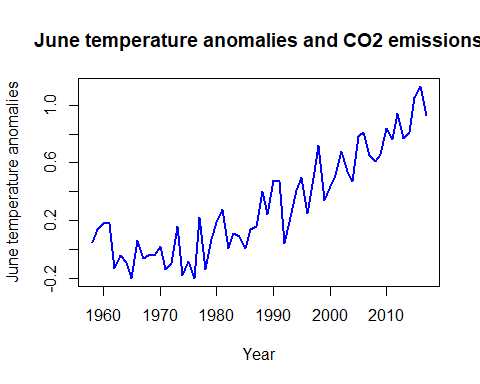
If you need more insight into correlation coefficients, you may find it helpful to watch online tutorials such as ‘Correlation coefficient intuition’ (<https://tinyco.re/4363520>) from the Khan Academy.

As we are dealing with time-series data, it is often more instructive to look at a line plot, as a scatterplot cannot convey how the observations relate to each other in the time dimension. If you were to check the variable types (using str(tempCO2data)), you would see that the data is not yet in time-series format. We could continue with the format as it is, but for plotting purposes it is useful to let R know that we are dealing with time-series data. We therefore apply the ts function as we did in Part 1.1.

tempCO2data$Jun <- ts(tempCO2data$Jun,   
 start = c(1958), end = c(2017), frequency = 1)   
tempCO2data$Trend <- ts(tempCO2data$Trend,   
 start = c(1958), end = c(2017), frequency = 1)

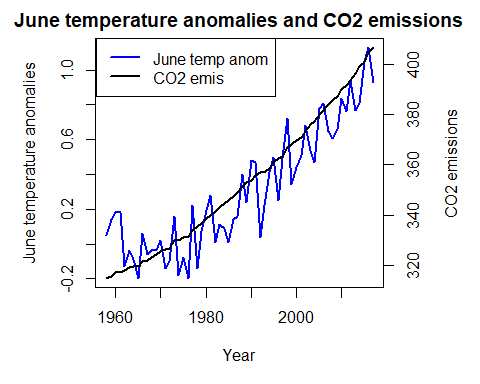
Let’s start by plotting the June temperature anomalies.

plot(tempCO2data$Jun, type = "l", col = "blue", lwd = 2,  
 ylab = "June temperature anomalies", xlab = "Year")  
  
title("June temperature anomalies and CO2 emissions")



Typically, when using the plot function we would now only need to add the line for the second variable using the lines command. The issue, however, is that the CO2 emissions variable (Trend) is on a different scale, and the automatic vertical axis scale (from –0.2 to about 1.2) would not allow for the display of Trend. To resolve this issue you can introduce a second vertical axis using the commands below. (*Tip:* You are unlikely to remember the exact commands required, however you can Google ‘R plot 2 vertical axes’ or a similar search term, and then adjust the code you find so it will work on your dataset.)

# Create extra margins used for the second axis  
par(mar = c(5, 5, 2, 5))  
  
plot(tempCO2data$Jun, type = "l", col = "blue", lwd = 2,  
 ylab = "June temperature anomalies", xlab = "Year")  
  
title("June temperature anomalies and CO2 emissions")   
  
# This puts the next plot into the same picture.  
par(new = T)  
  
# No axis, no labels  
plot(tempCO2data$Trend, pch = 16, lwd = 2,   
 axes = FALSE, xlab = NA, ylab = NA, cex = 1.2)   
axis(side = 4)  
mtext(side = 4, line = 3, 'CO2 emissions')  
  
legend("topleft", legend = c("June temp anom", "CO2 emis"),  
 lty = c(1, 1), col = c("blue", "black"), lwd = 2)

 The temperature anomalies for June, as well as CO2 emission levels, increase over time. Regardless of the variance in the anomalies, the rise in CO2 shows a linear trend.

[End of walk-through]