# Lecture 6: solutions

Statistical Computing with R

## Exercise 1

### 1.

Finding the dimensions and variable names of the dataframe is quite straightforward. To create the dataframe with means for numeric variables, we first create an empty dataframe that we can fill with information later. After that, we loop through the columns of the dataframe. For each column we check if it is numeric, and then we add its name and the mean to the empty dataframe. You can do this as shown below, or you can use the `rbind` function that has not been covered in the lectures yet.

```
my_function <- function(data){
  dm <- dim(data) # Calculate the dimensions
  vars <- names(data) # Retrieve the column names

  # Create an empty dataframe to add information to in the for loop
  summary <- data.frame(matrix(nrow=0, ncol=2))
  colnames(summary) <- c("Variable", "Mean")

  # Loop through the columns to fill the dataframe
  for (i in 1:ncol(data)){
    if (is.numeric(data[,i])){
      summary[nrow(summary)+1,] <- c(vars[i], round(mean(data[,i]),3))
    }
  }

  output <- list("dimensions" = dm,
                 "names" = vars,
                 "summary" = summary)

  return(output)
}
```

### 2.

```
test <- my_function(iris)
test
```

```
## $dimensions
## [1] 150   5
##
## $names
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
##
## $summary
##       Variable  Mean
## 1 Sepal.Length 5.843
## 2  Sepal.Width 3.057
## 3 Petal.Length 3.758
## 4  Petal.Width 1.199
```

### 3.

```
test[[3]]
```

```
##       Variable  Mean
## 1 Sepal.Length 5.843
## 2  Sepal.Width 3.057
## 3 Petal.Length 3.758
## 4  Petal.Width 1.199
```

```
# Or

test$summary
```

```
##       Variable  Mean
## 1 Sepal.Length 5.843
## 2  Sepal.Width 3.057
## 3 Petal.Length 3.758
## 4  Petal.Width 1.199
```

### 6.

```
test[[2]][1]
```

```
## [1] "Sepal.Length"
```

```
# Or

test$names[1]
```

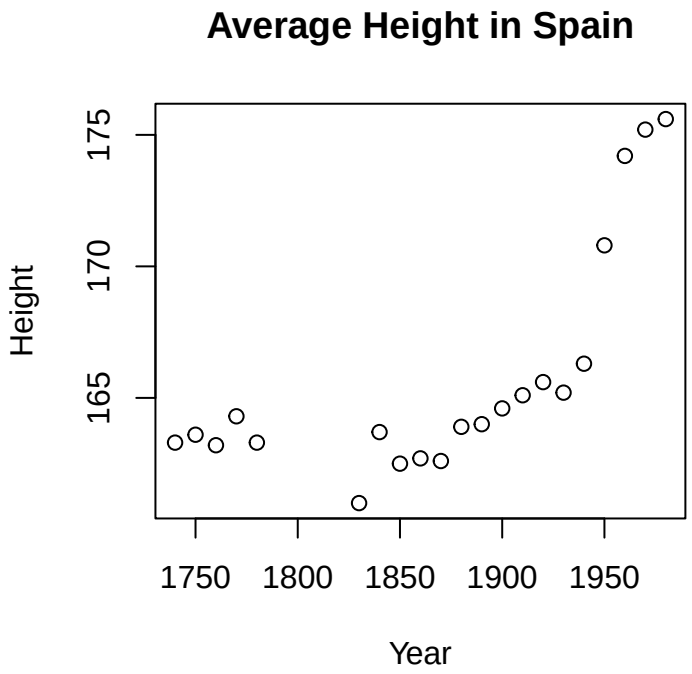
```
## [1] "Sepal.Length"
```

## Exercise 2

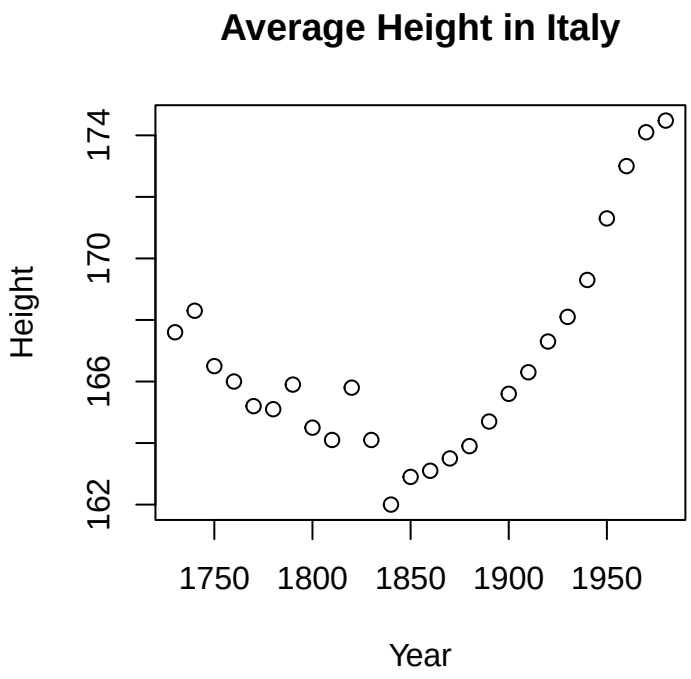
### 1.

First define the function as we did before. Within the function, you can plot the height for the country. `R` automatically returns the plot as an output of the function.

```
# Load brolgar library if not loaded before
library(brolgar)

# Define the function
my_scatter <- function(country){
  # Input:
  # country - a string with a country name

  # Get data specific to the country
  data <- heights[heights$country==country,]

  # plot the height_cm vs year, change the labels and add a title
  plot(x = data$year,
       y = data$height_cm,
       xlab = "Year",
       ylab = "Height",
       main = paste0("Average Height in ", country))
}

# Testing function
my_scatter("Spain")
```



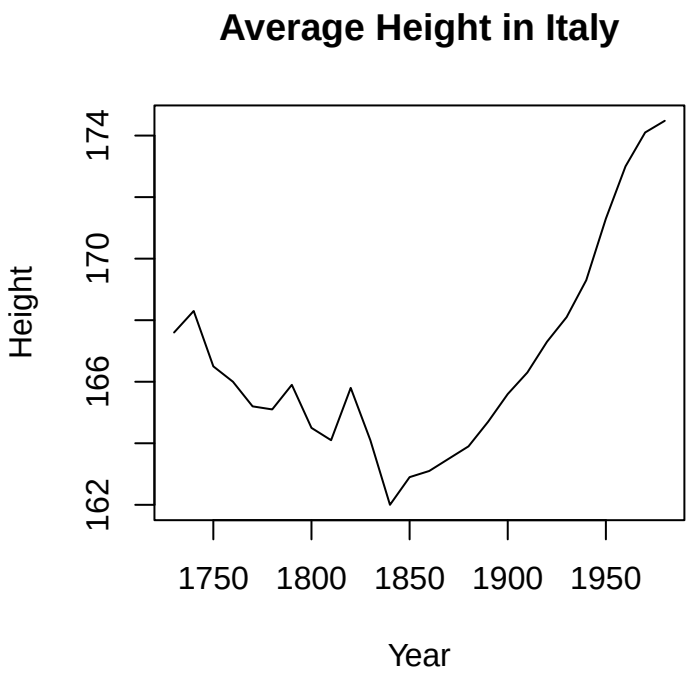
```
my_scatter("Italy")
```




**2.**

It is the same function as in part 1. Only now we change the line type. The lecture slides contain many modifications to the plots.

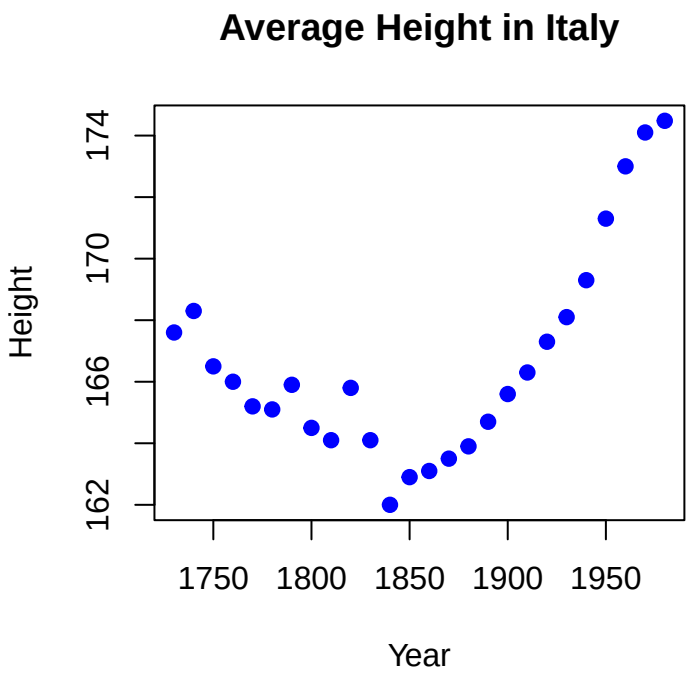
```
# Define the function
my_scatter <- function(country){
  # Input:
  # country - a string with a country name

  # Get data specific to the country
  data <- heights[heights$country==country,]

  # plot the height_cm vs year, change the labels, switch line type, and add a title
  plot(x = data$year,
       y = data$height_cm,
       xlab = "Year",
       ylab = "Height",
       type = "l",
       main = paste0("Average Height in ", country))
}

# Testing function
my_scatter("Italy")
```




### 3.

We now specify two more input parameters, that are used as inputs for the plot function to modify the shape and color of the points.

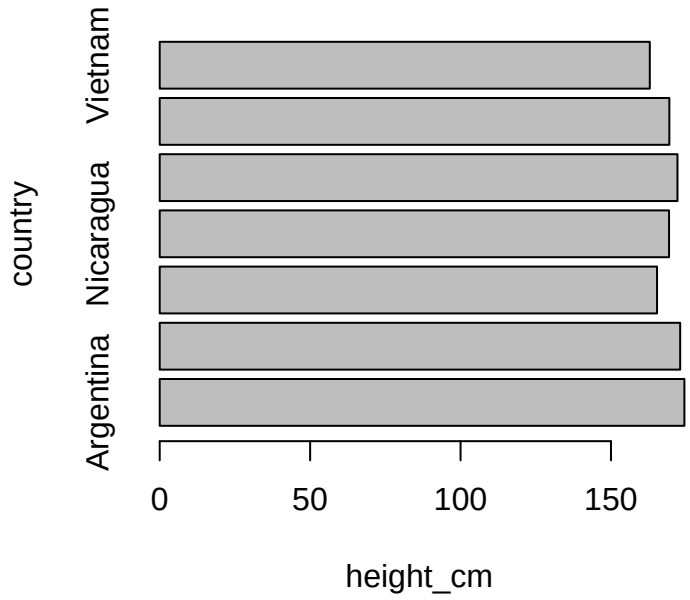
```
# Define the function
my_scatter <- function(country, shape = 19, pnt_col = "blue"){
  # Input:
  # country - a string with a country name
  # shape - the point shape, by default shape 19
  # pnt_col - the point or line color, by default blue

  # Get data specific to the country
  data <- heights[heights$country==country,]

  # plot the height_cm vs year, change the labels,
  # the modified point shape and color, and add a title
  plot(x = data$year,
       y = data$height_cm,
       xlab = "Year",
       ylab = "Height",
       col = pnt_col,
       pch = shape,
       main = paste0("Average Height in ", country))
}

# Testing function
my_scatter("Italy")
```



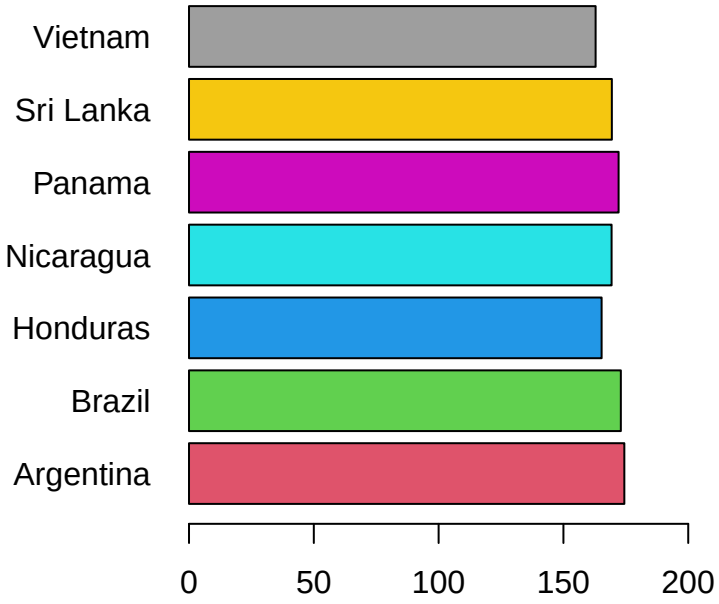

**4.**

Creating a barplot that shows the `height_cm` per country. First, we subset within the data argument. The command `%in%` specifies all countries within the vector countries. The `&` command stands for *AND*. Thus, if the country in the `country` column is in the vector `countries`, AND the year in the column year is equal to *1990*, the row will be included in the data for the plot. The argument `horiz = T` specifies that we want the bars to be horizontal rather than vertical.

```
# subsetting countries mentioned in exercise
countries <- c("Argentina", "Brazil", "Honduras", "Nicaragua",
               "Panama", "Sri Lanka", "Vietnam")

# Creating a barplot that shows the height_cm per country
barplot(height_cm ~ country,
        data = heights[heights$country %in% countries
                       & heights$year == "1990", ],
        horiz = T)
```



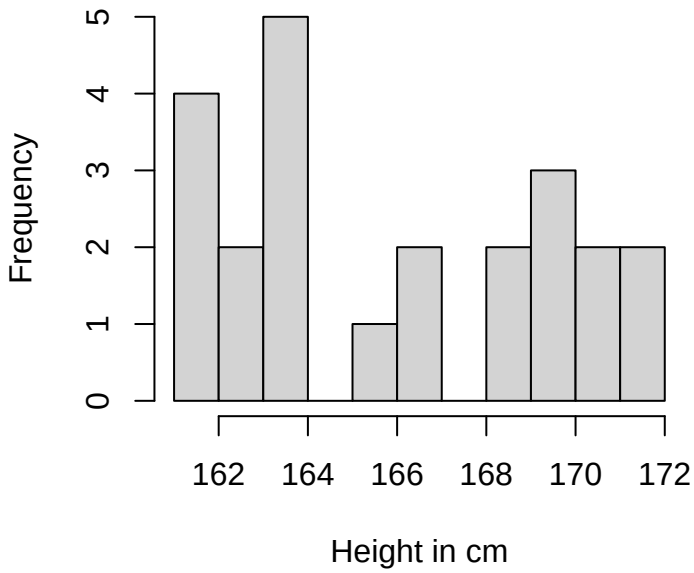

```
par(mar=c(3,5,3,2))
barplot(height_cm ~ country,
        data = heights[heights$country %in% countries
                       & heights$year == "1990", ],
        horiz = T,
        las = 1,
        mar = c(4, 7, 2, 4),
        ylab = "",
        col = 2:8,
        xlim = c(0, 200))
```



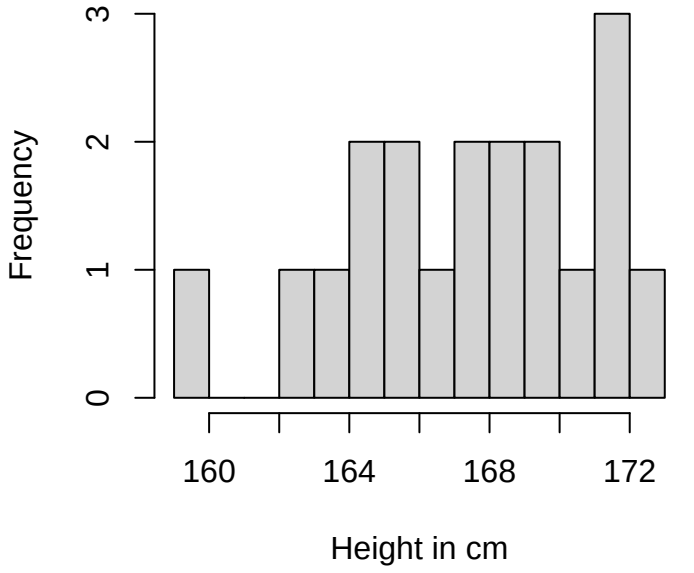

**5.**

Using the function `hist()` to plot the histogram, and using the functions `plot()`, `density()`, and `polygon()` to plot the density. It is difficult to see, but the mean height of Asian countries seems to have increased. Moreover, the variability between countries seems to have diminished.

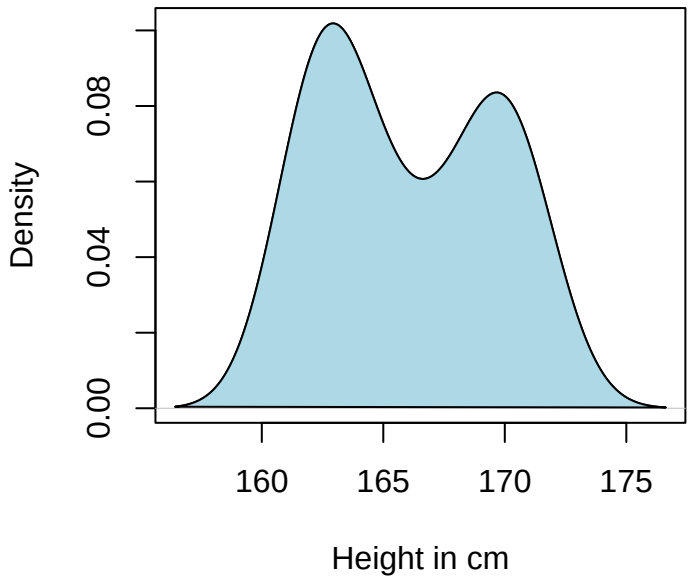
```
# subsetting asian countries
height_asia <- heights[heights$continent=="Asia",]

# making histogram
hist(height_asia$height_cm[height_asia$year=="1950"],
     main = "",
     xlab = "Height in cm",
     breaks = 10)
```



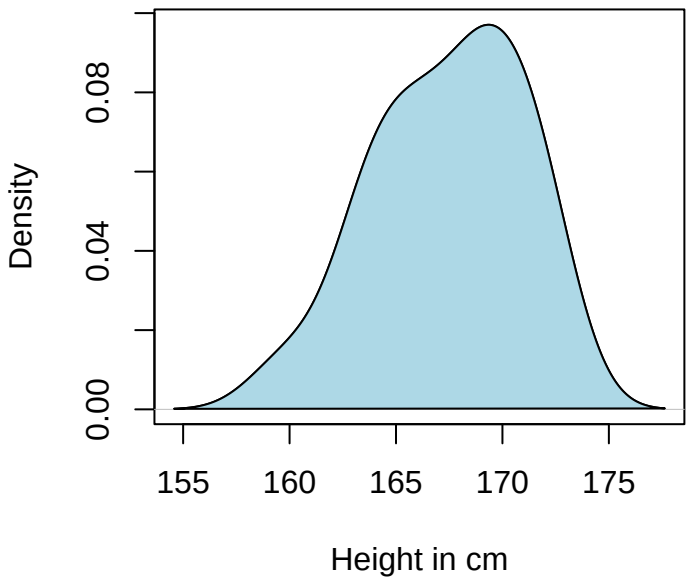

```
hist(height_asia$height_cm[height_asia$year=="1990"],
     main = "",
     xlab = "Height in cm",
     breaks = 10)
```



```r
# plotting the densities
plot(density(height_asia$height_cm[height_asia$year=="1950"]),
     main = "",
     xlab = "Height in cm")
polygon(density(height_asia$height_cm[height_asia$year=="1950"]),
        col = "lightblue")
```




```r
plot(density(height_asia$height_cm[height_asia$year=="1990"]),
     main = "",
     xlab = "Height in cm")
polygon(density(height_asia$height_cm[height_asia$year=="1990"]),
        col = "lightblue")
```


Density
0.00
0.04
0.08
155
160
165
170
175
Height in cm

# Exercise 3

## 1.

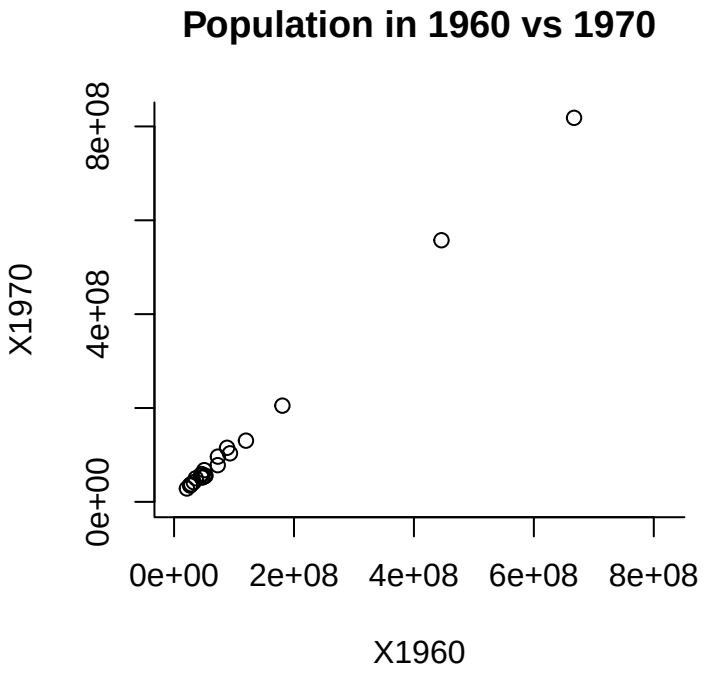
First, load the data. We first remove all rows that are not countries. You can copy this part of the code to save time or if you don't want to type it manually. Lastly, we save the countries with the largest population in 1990 in a separate data frame.

```
# setwd("D:/Leiden/Statistical_Computing_in_R/Lecture 6/0_Data")
population <- read.csv(
  "API_SP.POP.TOTL_DS2_en_csv_v2_5871594.csv",
  header = TRUE, skip = 4)

# deleting rows that are not countries
notCountry <- c("World", "IDA & IBRD total", "Low & middle income", "Middle income",
                "IBRD only", "Early-demographic dividend", "Lower middle income",
                "Upper middle income", "Late-demographic dividend",
                "East Asia & Pacific", "East Asia & Pacific (excluding high income)",
                "East Asia & Pacific (IDA & IBRD countries)", "South Asia",
                "South Asia (IDA & IBRD)", "OECD members", "High income",
                "Post-demographic dividend", "Europe & Central Asia", "IDA total",
                "IDA only", "Sub-Saharan Africa",
                "Sub-Saharan Africa (IDA & IBRD countries)",
                "Sub-Saharan Africa (excluding high income)",
                "Least developed countries: UN classification",
                "Latin America & Caribbean", "Fragile and conflict affected situations",
                "Europe & Central Asia (IDA & IBRD countries)",
                "Latin America & the Caribbean (IDA & IBRD countries)", "European Union",
                "Pre-demographic dividend",
                "Latin America & Caribbean (excluding high income)",
                "Europe & Central Asia (excluding high income)",
                "Heavily indebted poor countries (HIPC)", "Euro area",
                "Africa Eastern and Southern", "Low income", "IDA blend", "North America",
                "Middle East & North Africa",
                "Middle East & North Africa (excluding high income)",
                "Middle East & North Africa (IDA & IBRD countries)", "Arab World",
                "Africa Western and Central", "Central Europe and the Baltics")

popc <- population[-which(population$Country.Name %in% notCountry), ]

# subsetting countries
popc <- popc[,c("Country.Name","X1990","X1960","X1970","X1980","X2000","X2010","X2020")]

# Checking top20 countries by population size
poptop <- head(popc[order(popc$X1990, decreasing = T), ], 20)
```

## 2.

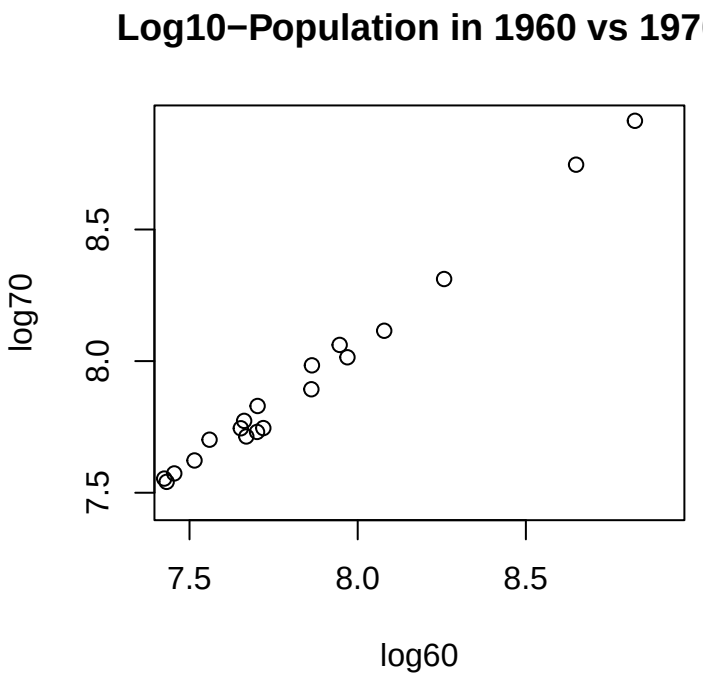
We plot the population sizes against each other. For the axis limits, I specified the limit to the largest population of 1970, because populations grow over time, and the largest population of a country in 1970 was likely larger than in 1960.

```r
# subsetting
pop6070c <- poptop[,c("Country.Name","X1960","X1970")]

par(bty = "l")

# plotting
plot(X1970 ~ X1960,
     data = pop6070c,
     xlim = c(0, max(pop6070c$X1970, na.rm=T)),
     ylim = c(0, max(pop6070c$X1970, na.rm=T)),
     main = "Population in 1960 vs 1970")
```




**3.**

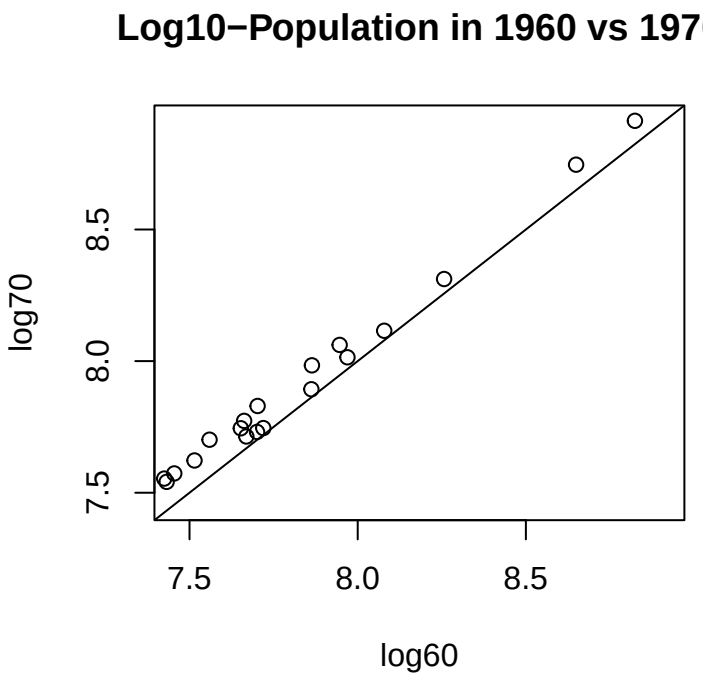
```r
# Taking the log with base 10 of the population numbers
pop6070c$log60 <- log(pop6070c$X1960, base=10)
pop6070c$log70 <- log(pop6070c$X1970, base=10)

plot(log70 ~ log60,
     data = pop6070c,
     xlim = c(min(pop6070c$log70, na.rm=T), max(pop6070c$log70, na.rm=T)),
     ylim = c(min(pop6070c$log70, na.rm=T), max(pop6070c$log70, na.rm=T)),
     main = "Log10-Population in 1960 vs 1970")
```




**4.**

Now we add a line by using `abline()`, with an intercept $a$ and slope $b$. In order to plot this line, you need to run both the `plot` and the `abline` function. Since all points are above the x = y line, it can be concluded that all countries experienced population growth.

```
# plotting
plot(log70 ~ log60,
     data = pop6070c,
     xlim = c(min(pop6070c$log70, na.rm=T), max(pop6070c$log70, na.rm=T)),
     ylim = c(min(pop6070c$log70, na.rm=T), max(pop6070c$log70, na.rm=T)),
     main = "Log10-Population in 1960 vs 1970")
# adding line with intercept a and slope b
abline(a = 0, b = 1)
```

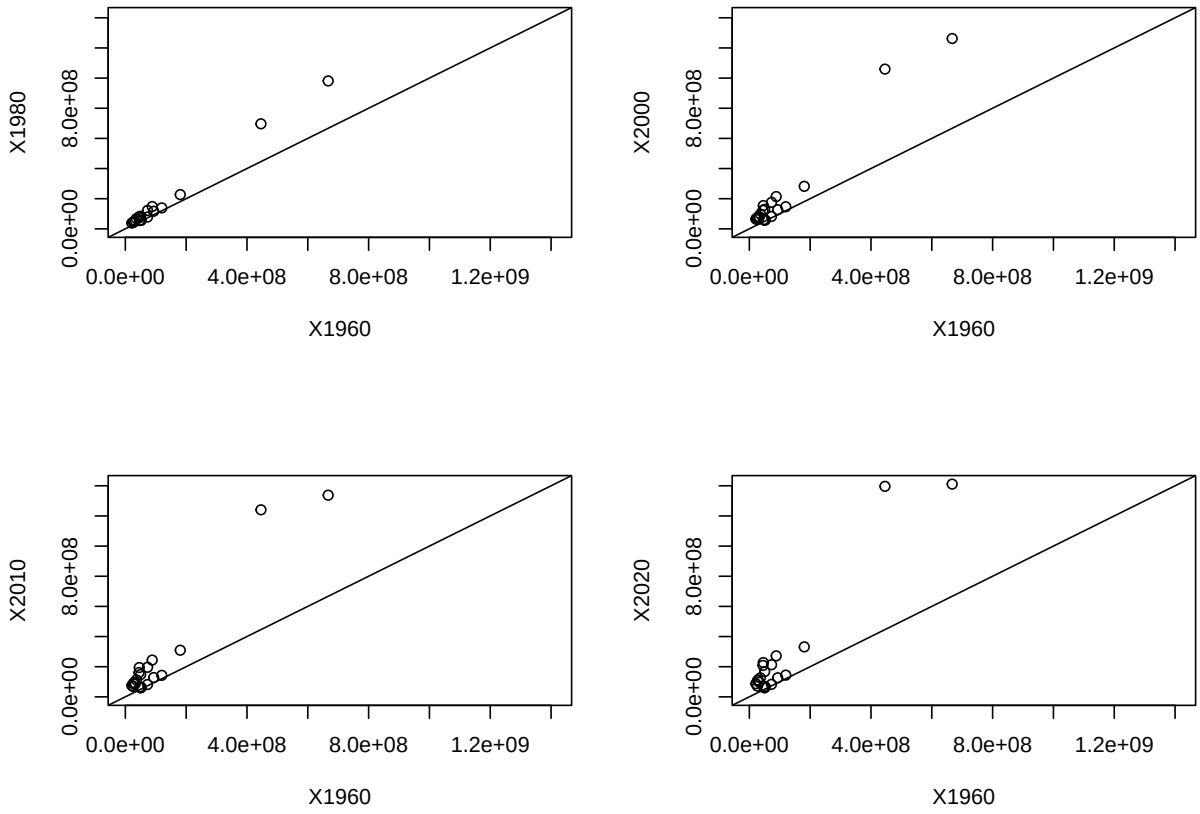



**5.**

The `par()` function allows to create panels.

```
# setting that we want four panels, each with a plot, combined in one figure
par(mfrow=c(2,2))

plot(X1980 ~ X1960,
     data = poptop,
     xlim = c(0, max(poptop$X2020, na.rm=T)),
     ylim = c(0, max(poptop$X2020, na.rm=T)))
abline(a = 0, b = 1)

plot(X2000 ~ X1960,
     data = poptop,
     xlim = c(0, max(poptop$X2020, na.rm=T)),
     ylim = c(0, max(poptop$X2020, na.rm=T)))
abline(a = 0, b = 1)

plot(X2010 ~ X1960,
     data = poptop,
     xlim = c(0, max(poptop$X2020, na.rm=T)),
     ylim = c(0, max(poptop$X2020, na.rm=T)))
abline(a = 0, b = 1)

plot(X2020 ~ X1960,
     data = poptop,
     xlim = c(0, max(poptop$X2020, na.rm=T)),
```

```
     ylim = c(0, max(poptop$X2020, na.rm=T)))
abline(a = 0, b = 1)
```



## Exercise 4

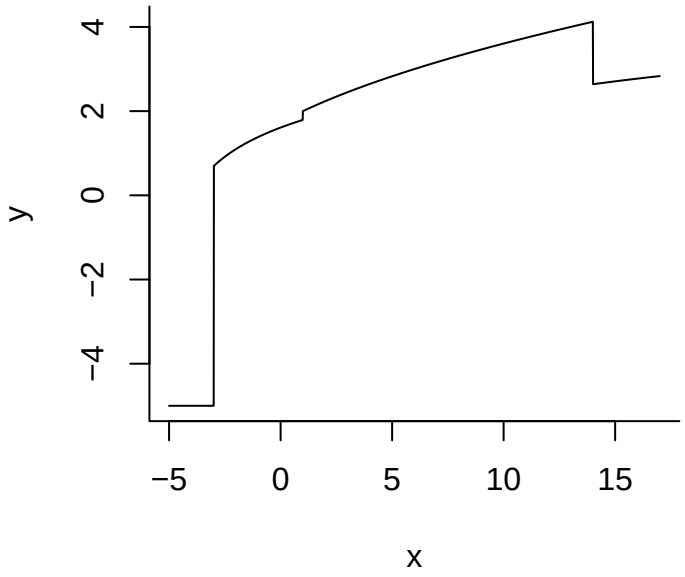
Make sure to copy your answer from week 4, or to redefine the function $f(x)$.

```
f.x <- function(x){
  if(x <= -3){
    return(-5)
  }else if(x > -3 & x < 1){
    return(log(x + 5))
  }else if(x == 1){
    return(2)
  }else if(x > 1 & x <= 14){
    return(sqrt(x + 3))
  }else{
    return(log(x))
  }
}
```

### 1.

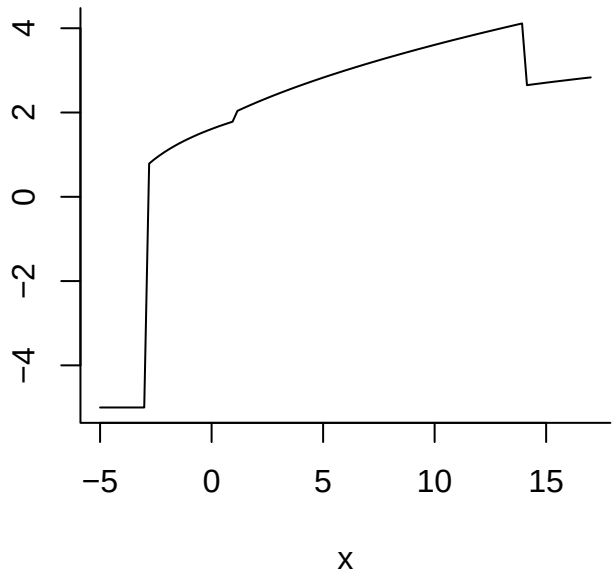
There are two solutions. First, a "poor man's" approach. In this solution you are plotting the function section for section to circumvent the non-continuous part of the function.

```
x = seq(-5, 17, by = 0.01)
y = rep(NA, length(x))
for (i in 1:length(x)) y[i] = f.x(x[i])
par(bty = 'l')
plot(x, y, type = 'l')
```



A nicer approach might be to vectorize the function $f(x)$. Since $f(x)$ is not continuous, the function needs to be enclosed with `Vectorize()` in order for the `curve()` function to treat it as continuous.

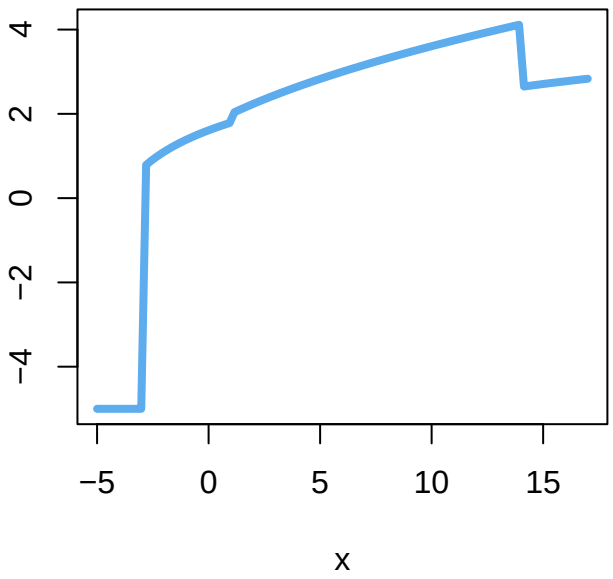
```
par(bty = "l")
curve(Vectorize(f.x)(x), xlim = c(-5, 17), ylab = "")
```




**2.**

The color and width of the line can be modified using the `col` and `lwd` (line width) parameters respectively.

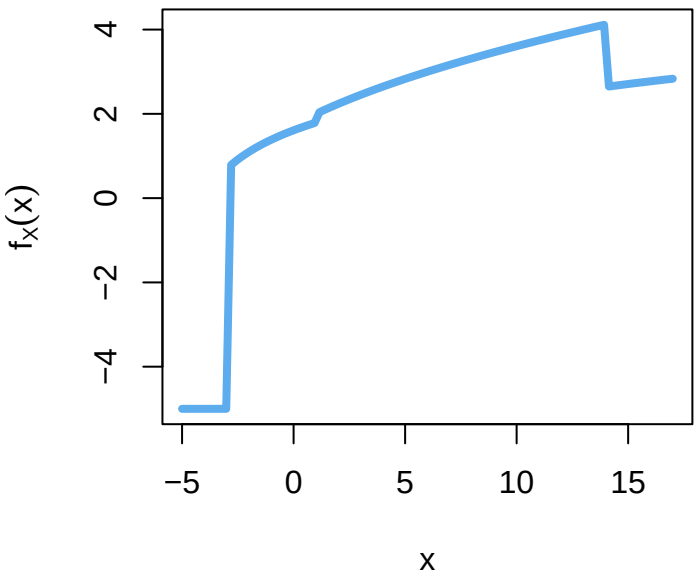
```
curve(Vectorize(f.x)(x), xlim = c(-5, 17), col = "steelblue2", lwd = 4, ylab = "")
```




### 3.

The `expression` function allows for mathematical notation in plots.

```
curve(Vectorize(f.x)(x),
      xlim = c(-5, 17),
      col = "steelblue2",
      lwd = 4,
      ylab = expression(f[X](x)))
```


f_X(x)
4
2
0
−2
−4
−5
0
5
10
15
x

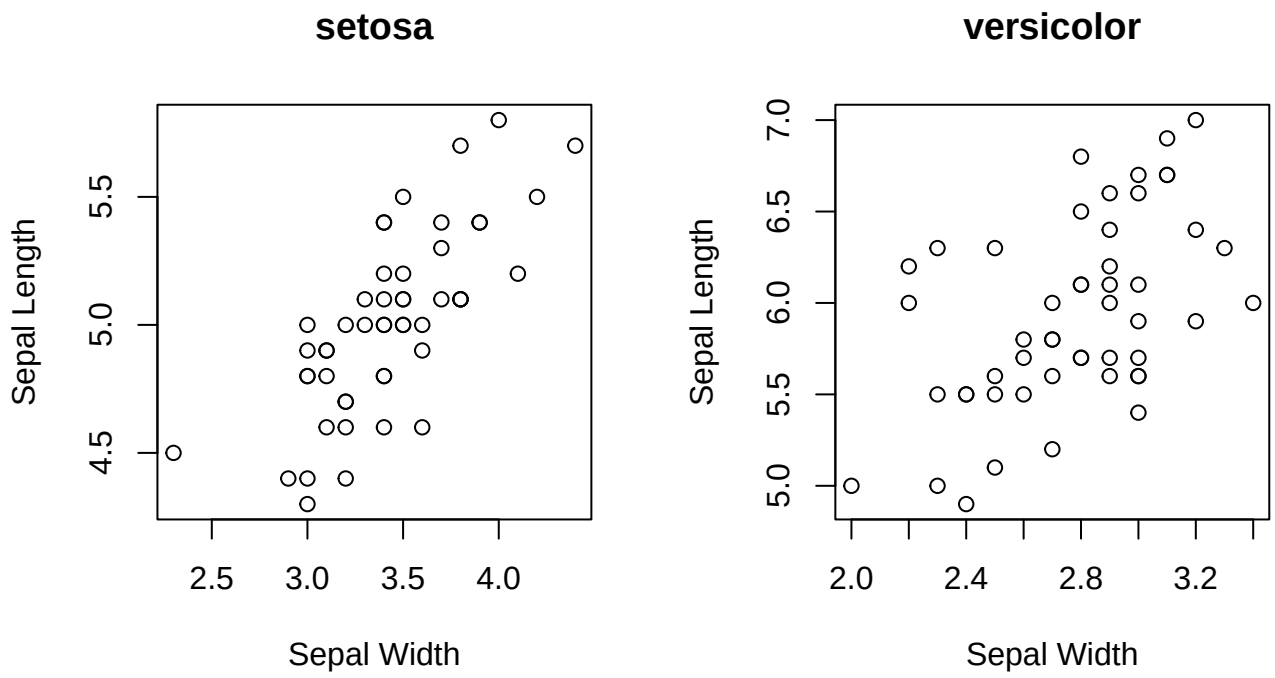
## Exercise 5

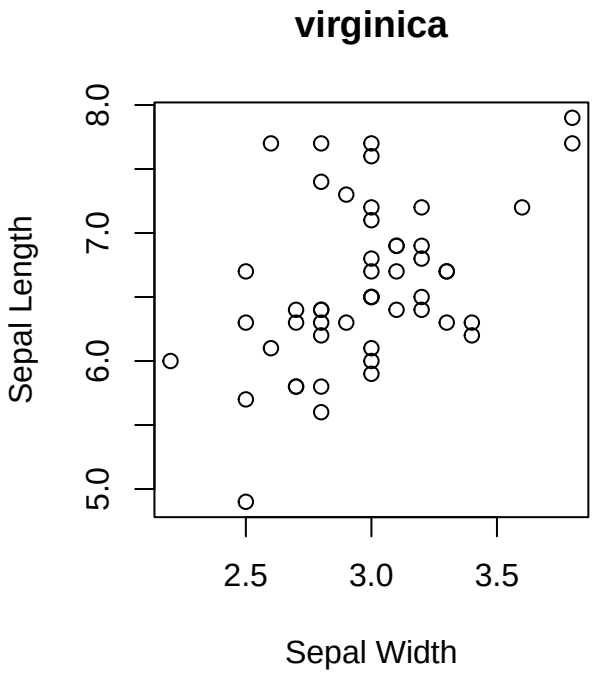
### 1.

```
splitiris <- split(iris, iris$Species)
```

### 2.

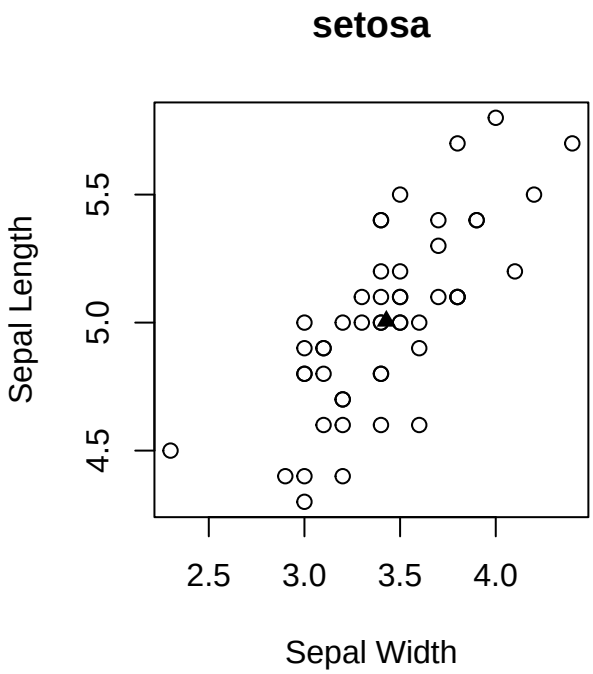
```
for (i in 1:length(splitiris)){
  plot(splitiris[[i]]$Sepal.Length ~ splitiris[[i]]$Sepal.Width,
       main = names(splitiris)[i], xlab = "Sepal Width", ylab = "Sepal Length")
}
```

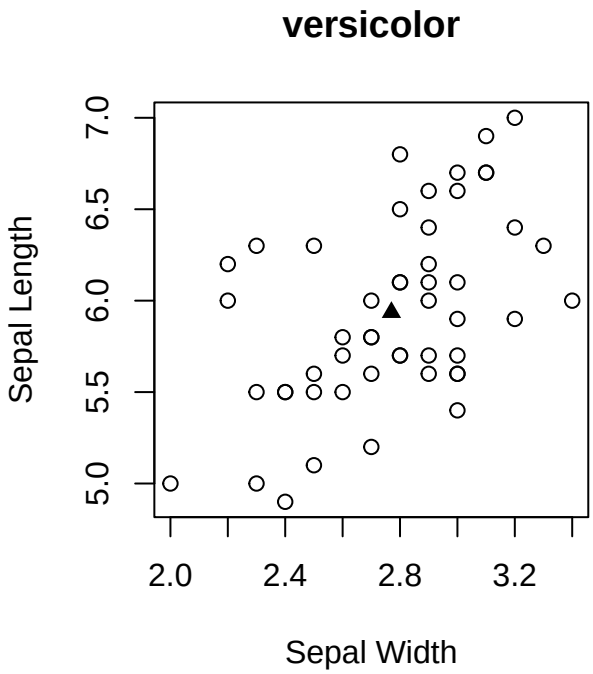


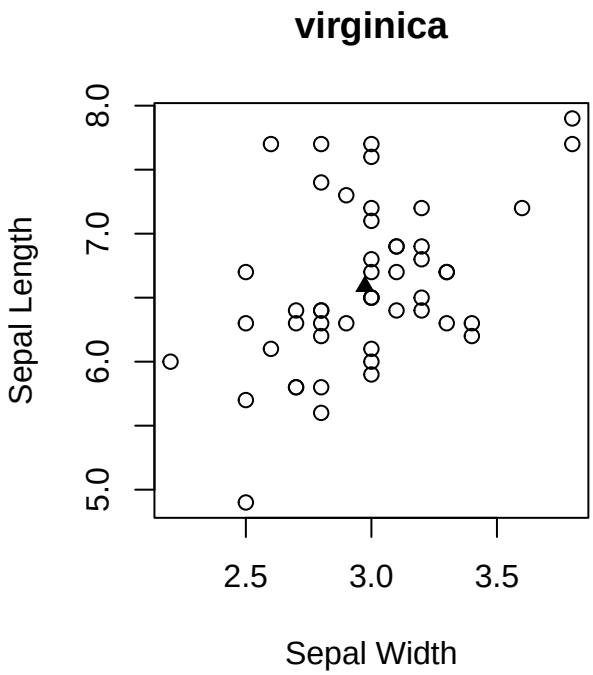



**3.**

```
for (i in 1:length(splitiris)){
  plot(splitiris[[i]]$Sepal.Length ~ splitiris[[i]]$Sepal.Width,
       main = names(splitiris)[i], xlab = "Sepal Width", ylab = "Sepal Length")
  points(mean(splitiris[[i]]$Sepal.Width), mean(splitiris[[i]]$Sepal.Length), pch=17)
}
```




**4.**

First check which of objects in the list represents the Setosa species. Then remove that object from the list.

```r
index <- which(names(splitiris) == "setosa")
splitiris[index] <- NULL
```