Statistics 1

Contents

About this course

This course is an introduction to data science. We have three primary aims. First, to introduce you to the logic of quantitative research design. Second, to familiarise you with statistical models that scientists and policy-makers use to answer social science questions. Third, to help you acquire the necessary skills to conduct your own quantitative research projects. No prior statistical knowledge is assumed. We will use the statistical software R and RStudio on top.

My **office hours are Mondays from 4 pm to 6 pm in room 140**. You can drop-in in groups. There is no need to book in advance. Please make use of the office hours.

Syllabus

Moodle

Piazza

1 Introduction: Measurement, Central Tendency, Dispersion, Validity, Reliability

1.1 Seminar

In this seminar session, we introduce working with R. We illustrate some basic functionality and help you familiarise yourself with the look and feel of RStudio. Measures of central tendency and dispersion are easy to calculate in R. We focus on introducing the logic of R first and then describe how central tendency and dispersion are calculated in the end of the seminar.

1.1.1 Getting Started

Install R and RStudio on your computer by downloading them from the following sources:

- Download R from The Comprehensive R Archive Network (CRAN)
- Download RStudio from RStudio.com

1.1.2 RStudio

Let's get acquainted with R. When you start RStudio for the first time, you'll see three panes:

1.1.3 Console

The Console in RStudio is the simplest way to interact with R. You can type some code at the Console and when you press ENTER, R will run that code. Depending on what you type, you may see some output in the Console or if you make a mistake, you may get a warning or an error message.

Let's familiarize ourselves with the console by using R as a simple calculator:

```
2 + 4
```

```
[1] 6
```

Now that we know how to use the `+` sign for addition, let's try some other mathematical operations such as subtraction (`-`), multiplication (`*`), and division (`/`).

```
10 - 4
```

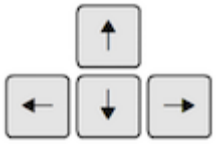
```
[1] 6
```

```
5 * 3
```

```
[1] 15
```

```
7 / 2
```

```
[1] 3.5
```



You can use the cursor or arrow keys on your keyboard to edit your code at the console:- Use the UP and DOWN keys to re-run something without typing it again- Use the LEFT and RIGHT keys to edit

Take a few minutes to play around at the console and try different things out. Don't worry if you make a mistake, you can't break anything easily!

1.1.4 Functions

Functions are a set of instructions that carry out a specific task. Functions often require some input and generate some output. For example, instead of using the `+` operator for addition, we can use the `sum` function to add two or more numbers.

```
sum(1, 4, 10)
```

```
[1] 15
```

In the example above, `1, 4, 10` are the inputs and 15 is the output. A function always requires the use of parenthesis or round brackets `()`. Inputs to the function are called **arguments** and go inside the brackets. The output of a function is displayed on the screen but we can also have the option of saving the result of the output. More on this later.

1.1.5 Getting Help

Another useful function in R is `help` which we can use to display online documentation. For example, if we wanted to know how to use the `sum` function, we could type `help(sum)` and look at the online documentation.

```
help(sum)
```

The question mark `?` can also be used as a shortcut to access online help.

```
?sum
```

Use the toolbar button shown in the picture above to expand and display the help in a new window.

Help pages for functions in R follow a consistent layout generally include these sections:

Description	A brief description of the function
Usage	The complete syntax or grammar including all arguments (inputs)
Arguments	Explanation of each argument
Details	Any relevant details about the function and its arguments
Value	The output value of the function
Examples	Example of how to use the function

1.1.6 The Assignment Operator

Now we know how to provide inputs to a function using parenthesis or round brackets `()`, but what about the output of a function?

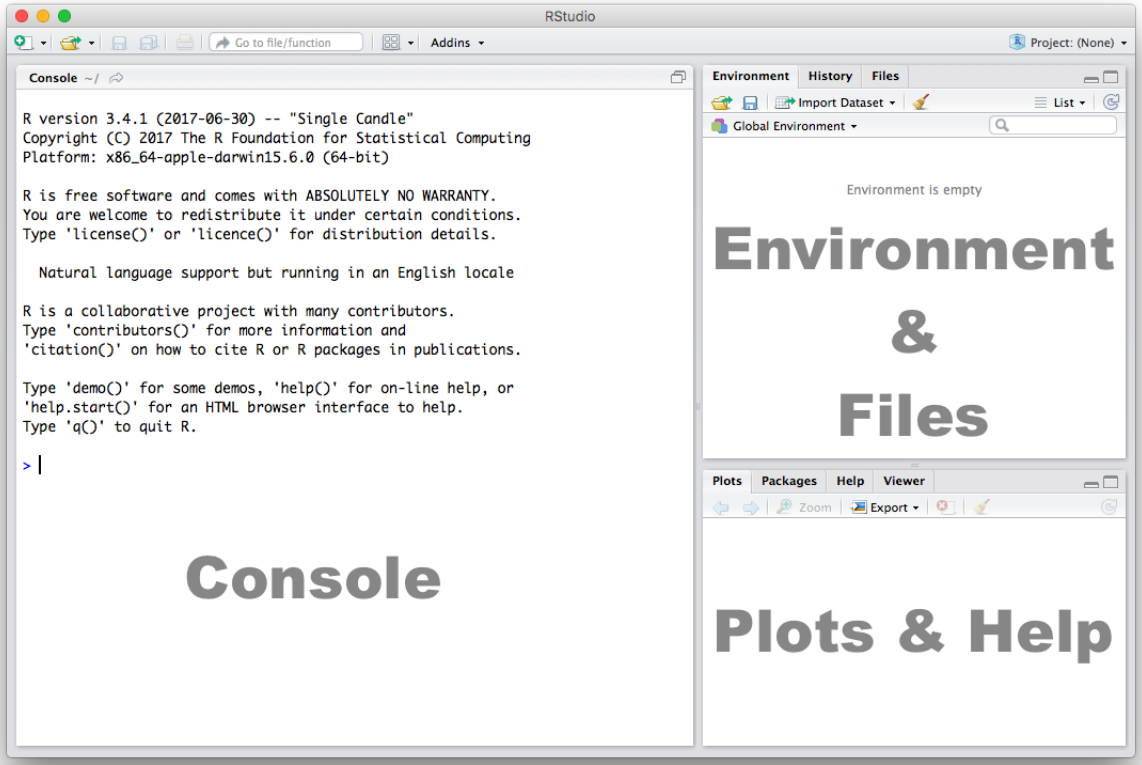


Figure 1:

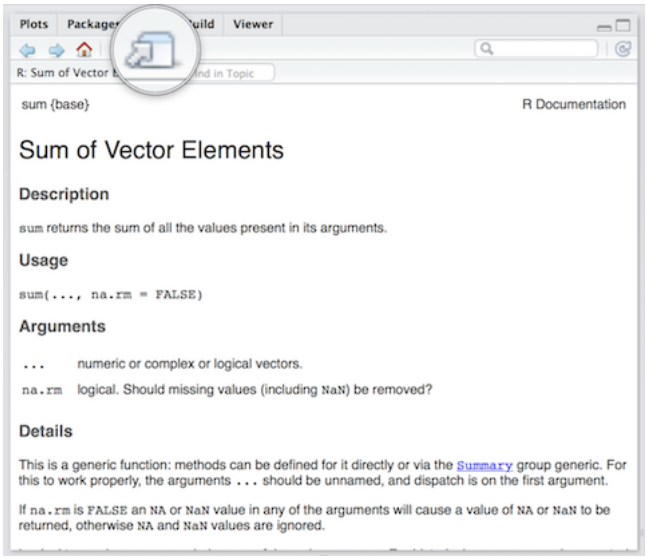


Figure 2:

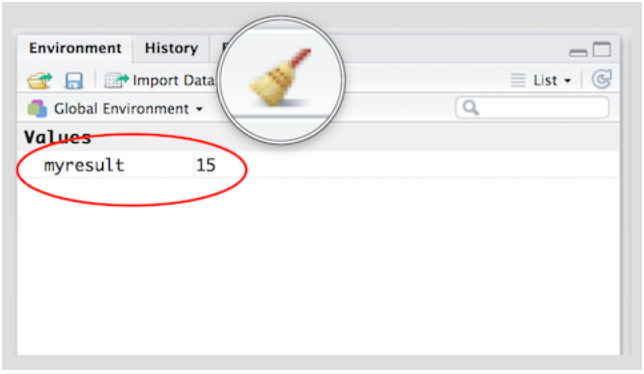


Figure 3:

We use the assignment operator `<-` for creating or updating objects. If we wanted to save the result of adding `sum(1, 4, 10)`, we would do the following:

```
myresult <- sum(1, 4, 10)
```

The line above creates a new object called `myresult` in our environment and saves the result of the `sum(1, 4, 10)` in it. To see what's in `myresult`, just type it at the console:

```
myresult
```

```
[1] 15
```

Take a look at the **Environment** pane in RStudio and you'll see `myresult` there.

To delete all objects from the environment, you can use the **broom** button as shown in the picture above.

We called our object `myresult` but we can call it anything as long as we follow a few simple rules. Object names can contain upper or lower case letters (`A-Z`, `a-z`), numbers (`0-9`), underscores (`_`) or a dot (`.`) but all object names must start with a letter. Choose names that are descriptive and easy to type.

Good Object Names	Bad Object Names
result	a
myresult	x1
my.result	this.name.is.just.too.long
my_result	
data1	

1.1.7 Sequences

We often need to create sequences when manipulating data. For instance, you might want to perform an operation on the first 10 rows of a dataset so we need a way to select the range we're interested in.

There are two ways to create a sequence. Let's try to create a sequence of numbers from 1 to 10 using the two methods:

1. Using the colon `:` operator. If you're familiar with spreadsheets then you might've already used `:` to select cells, for example `A1:A20`. In R, you can use the `:` to create a sequence in a similar fashion:

```
1:10
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

1. Using the `seq` function we get the exact same result:

```
seq(from = 1, to = 10)
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```


The `seq` function has a number of options which control how the sequence is generated. For example to create a sequence from 0 to 100 in increments of `5`, we can use the optional `by` argument. Notice how we

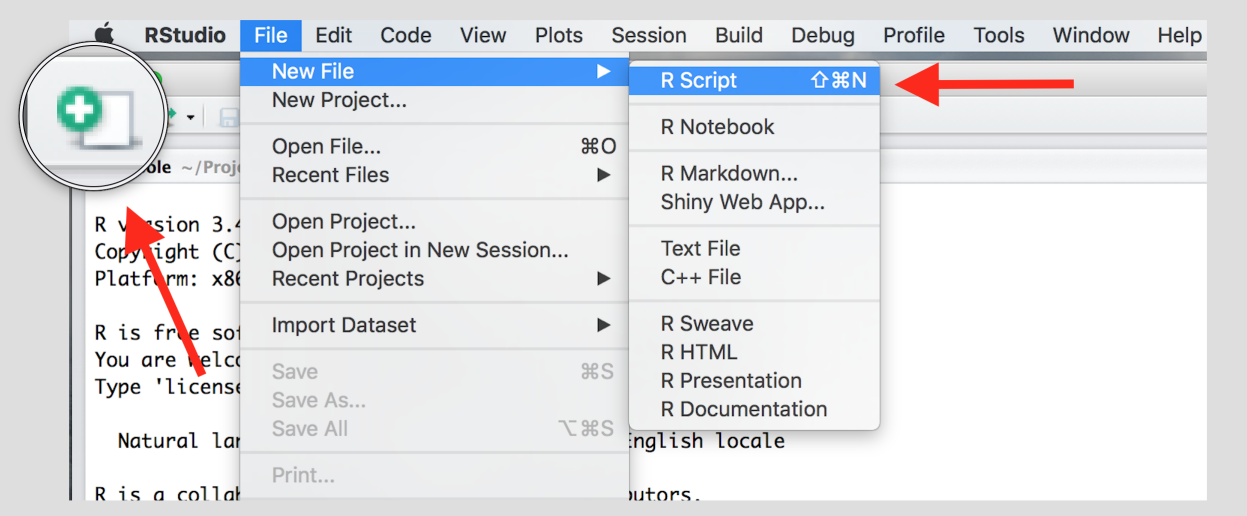


Figure 4:

Another common use of the `seq` function is to create a sequence of a specific length. Here, we create a sequence from 0 to 100 with length 9, i.e., the result is a vector with 9 elements.

```
seq(from = 0, to = 100, length.out =  9)
```

```
[1]   0.0  12.5  25.0  37.5  50.0  62.5  75.0  87.5 100.0
```

Now it's your turn:

- Create a sequence of **odd** numbers between 0 and 100 and save it in an object called `odd_numbers`

```
odd_numbers <- seq(1, 100, 2)
```

- Next, display `odd_numbers` on the console to verify that you did it correctly

```
odd_numbers
```

```
 [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45
[24] 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91
[47] 93 95 97 99
```

- What do the numbers in square brackets `[ ]` mean? Look at the number of values displayed in each line to find out the answer.
- Use the `length` function to find out how many values are in the object `odd_numbers`.
  - HINT: Try `help(length)` and look at the examples section at the end of the help screen.

```
length(odd_numbers)
```

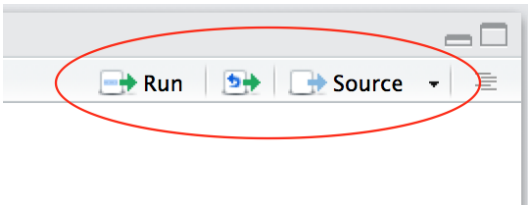
```
[1] 50
```

1.1.8 Scripts

The Console is great for simple tasks but if you're working on a project you would mostly likely want to save your work in some sort of a document or a file. Scripts in R are just plain text files that contain R code. You can edit a script just like you would edit a file in any word processing or note-taking application.

Create a new script using the menu or the toolbar button as shown below.

Once you've created a script, it is generally a good idea to give it a meaningful name and save it immediately. For our first session save your script as **seminar1.R**



Familiarize yourself with the script window in RStudio, and especially the two buttons labeled **Run** and **Source**

There are a few different ways to run your code from a script.

One line at a time	Place the cursor on the line you want to run and hit CTRL-ENTER or use the **Run** button
Multiple lines	Select the lines you want to run and hit CTRL-ENTER or use the **Run** button
Entire script	Use the **Source** button

1.1.9 Central Tendency

The appropriate measure of central tendency depends on the level of measurement of the variable. To recap:

Level of measurement	Appropriate measure of central tendency
Continuous	arithmetic mean (or average)
Ordered	median (or the central observation)
Nominal	mode (the most frequent value)

1.1.9.1 Mean

We calculate the average grade on our eleven homework assignments in statistics 1. We create our vector of 11 (fake) grades first using the `c()` function, where `c` stands for collect or concatenate.

```
hw.grades <- c(80, 90, 85, 71, 69, 85, 83, 88, 99, 81, 92)
```

We now take the sum of the grades.

```
sum.hw.grades <- sum(hw.grades)
```

We also take the number of grades

```
number.hw.grades <- length(hw.grades)
```

The mean is the sum of grades over the number of grades.

```
sum.hw.grades / number.hw.grades
```

```
[1] 83.90909
```

R provides us with an even easier way to do the same with a function called `mean()`.

```
mean(hw.grades)
```

```
[1] 83.90909
```

1.1.9.2 Median

The median is the appropriate measure of central tendency for ordinal variables. Ordinal means that there is a rank ordering but not equally spaced intervals between values of the variable. Education is a common example. In education, more education is better. But the difference between primary school and secondary school is not the same as the difference between secondary school and an undergraduate degree.

Let's generate a fake example with 100 people. We use numbers to code different levels of education.

Code	Meaning	Frequency in our data
0	no education	1
1	primary school	5
2	secondary school	55
3	undergraduate degree	20
4	postgraduate degree	10
5	doctorate	9

We introduce a new function to create a vector. The function `rep()`, replicates elements of a vector. Its arguments are the item `x` to be replicated and the number of `times` to replicate. Below, we create the variable education with the frequency of education level indicated above. Note that the arguments `x` and `times` do not have to be written out.

```
edu <- c( rep(x=0, times=1), rep(x=1, times=5), rep(x=2, times=55),
          rep(x=3, times=20), rep(4,10), rep(5,9) )
```

The median level of education is the level where 50 percent of the observations have a lower or equal level of education and 50 percent have a higher or equal level of education. That means that the median splits the data in half.

We use the `median()` function for finding the median.

```
median(edu)
```

```
[1] 2
```

The median level of education is secondary school.

1.1.9.3 Mode

The mode is the appropriate measure of central tendency if the level of measurement is nominal. Nominal means that there is no ordering implicit in the values that a variable takes on. We create data from 1000 (fake) voters in the United Kingdom who each express their preference on remaining in or leaving the European Union. The options are leave or stay. Leaving is not greater than staying and vice versa (even though we all order the two options normatively).

Code	Meaning	Frequency in our data
0	leave	509
1	stay	491

```
stay <- c(rep(0, 509), rep(1, 491))
```

The mode is the most common value in the data. There is no mode function in R. The most straightforward way to determine the mode is to use the `table()` function. It returns a frequency table. We can easily see the mode in the table. As your coding skills increase, you will see other ways of recovering the mode from a vector.

```
table(stay)
```

```
stay
  0   1
509 491
```

The mode is leaving the EU because the number of 'leavers' (0) is greater than the number of 'remainers' (1).

1.1.10 Dispersion

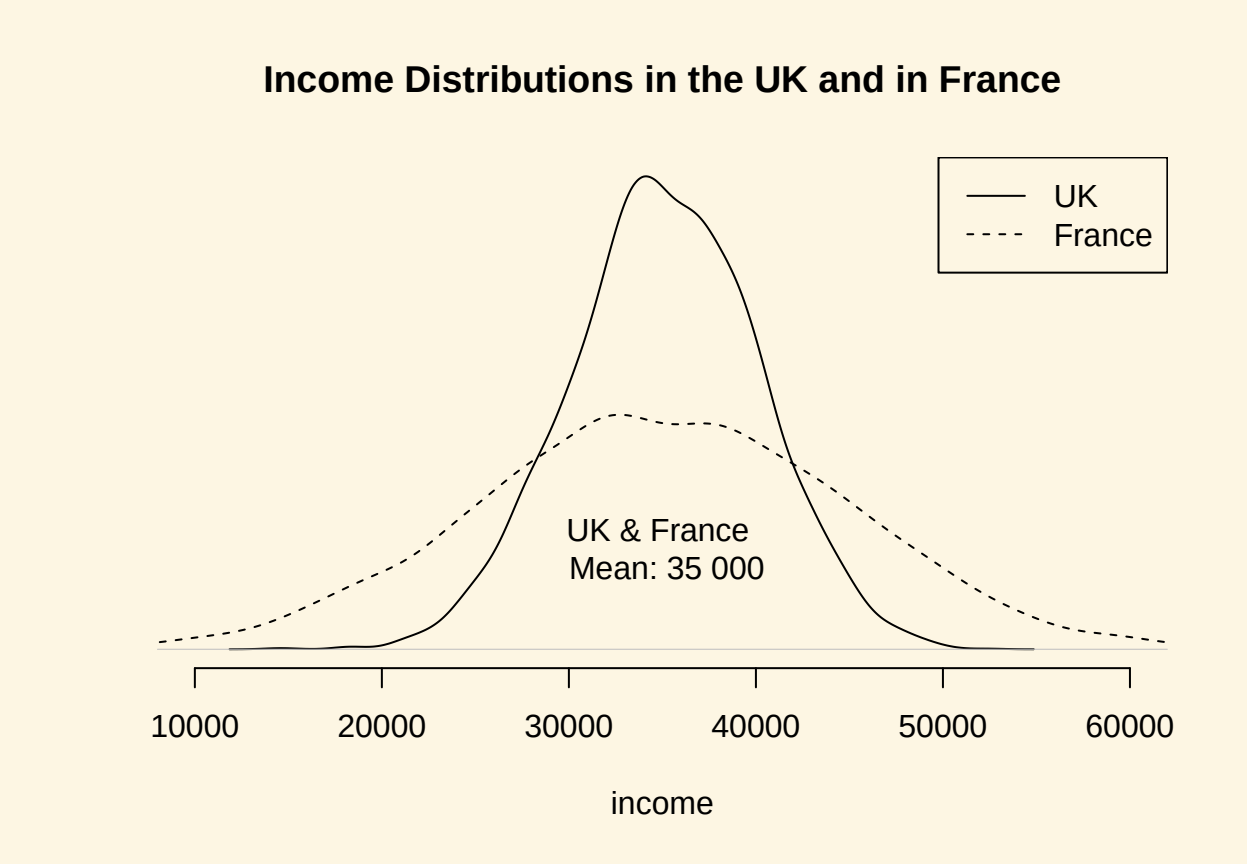
The appropriate measure of dispersion depends on the level of measurement of the variable we wish to describe.

Level of measurement	Appropriate measure of dispersion
Continuous	variance and/or standard deviation
Ordered	range or interquartile range
Nominal	proportion in each category

1.1.10.1 Variance and standard deviation

Both the variance and the standard deviation tell by how much an average realisation of a variable differs from the mean of that variable. Let's assume that our variable is income in the UK. Let's assume that its mean is 35 000 per year. We also assume that the average deviation from 35 000 is 5 000. If we ask 100 people in the UK at random about their income, we get 100 different answers. If we average the differences betweeen the 100 answers and 35 000, we would get 5 000. Suppose that the average income in France is also 35 000 per year but the average deviation is 10 000 instead. This would imply that income is more equally distributed in the UK than in France.

Dispersion is important to describe data as this example illustrates. Although, mean income in our hypothetical example is the same in France and the UK, the distribution is tighter in the UK. The figure below illustrates our example:



The variance gives us an idea about the variability of data. The formula for the variance in the population is

$$\frac{\sum_{i=1}^{n}(x_i - \mu_x)^2}{n}$$

The formula for the variance in a sample adjusts for sampling variability, i.e., uncertainty about how well our sample reflects the population by subtracting 1 in the denominator. Subtracting 1 will have next to no effect if n is large but the effect increases the smaller n. The smaller n, the larger the sample variance. The intuition is, that in smaller samples, we are less certain that our sample reflects the population. We, therefore, adjust variability of the data upwards. The formula is

$$\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

Notice the different notation for the mean in the two formulas. We write μ_x for the mean of x in the population and $\bar{x}$ for the mean of x in the sample. Notation is, however, unfortunately not always consistent.

Take a minute to think your way through the formula. There are 4 setps: (1), In the numerator, we subtract the mean of x from some realisation of x. (2), We square the deviations from the mean because we want positive numbers only. (3) We sum the squared deviations. (4) We divide the sum by $(n-1)$. Below we show this for the homework example. In the last row, we add a 5th step. We take the square root in order to return to the orginial units of the homework grades.

Obs	Var	Dev. from mean	Squared dev. from mean
i	grade	$x_i - \bar{x}$	$(x_i - \bar{x})^2$
1	80	-3.9090909	15.2809917

Obs	Var	Dev. from mean	Squared dev. from mean
2	90	6.0909091	37.0991736
3	85	1.0909091	1.1900826
4	71	-12.9090909	166.6446281
5	69	-14.9090909	222.2809917
6	85	1.0909091	1.1900826
7	83	-0.9090909	0.8264463
8	88	4.0909091	16.7355372
9	99	15.0909091	227.7355372
10	81	-2.9090909	8.4628099
11	92	8.0909091	65.4628099
$\sum_{i=1}^{n}$			762.9090909
$\div n-1$			76.2909091
$\surd$			8.7344667

Our first grade (80) is below the mean (83.9090909). The sum is, thus, negative. Our second grade (90) is above the mean, so that the sum is positive. Both are deviations from the mean (think of them as distances). Our sum shall reflect the total sum of distances and distances must be positive. Hence, we square the distances from the mean. Having done this for all eleven observations, we sum the squared distances. Dividing by 10 (with the sample adjustment), gives us the average squared deviation. This is the variance. The units of the variance—squared deviations—are somewhat awkward. We return to this in a moment.

We take the variance in R by using the `var()` function. By default `var()` takes the sample variance.

```
var(hw.grades)
```

```
[1] 76.29091
```

The average squared difference form our mean grade is 76.2909091. But what does that mean? We would like to get rid of the square in our units. That's what the standard deviation does. The standard deviation is the square root over the variance.

$$\sqrt{\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n-1}}$$

We get the average deviation from our mean grade (83.9090909) with the `sd()` function.

```
sd(hw.grades)
```

```
[1] 8.734467
```

The standard deviation is much more intuitive than the variance because its units are the same as the units of the variable we are interested in. "Why teach us about this awful variance then", you ask. Mathematically, we have to compute the variance before getting the standard deviation. We recommend that you use the standard deviation to describe the variability of your continuous data.

Note: We used the sample variance and sample standard deviation formulas. If the eleven assignments represent the population, we would use the population variance formula. Whether the 11 cases represent a sample or the population depends on what we want to know. If we want learn about all students' assignments or future assignments, the 11 cases are a sample.

1.1.10.2 Range and interquartile range

The proper measure of dispersion of an ordinal variable is the range or the interquartile range. The interquartile range is usually the preferred measure because the range is strongly affected by outlying cases.

Let's take the range first. We get back to our education example. In R, we use the `range()` function to compute the range.

```
range(edu)
```

```
[1] 0 5
```

Our data ranges from no education all the way to those with a doctorate. However, no education is not a common value. Only one person in our sample did not have any education. The interquartile range is the range from the 25th to the 75th percentiles, i.e., it contains the central 50 percent of the distribution.

The 25th percentile is the value of education that 25 percent or fewer people have (when we order education from lowest to highest). We use the `quantile()` function in R to get percentiles. The function takes two arguments: `x` is the data vector and `probs` is the percentile.

```
quantile(edu, 0.25) # 25th percentile
```

```
25%
  2
```

```
quantile(edu, 0.75) # 75th percentile
```

```
75%
  3
```

Therefore, the interquartile range is from 2, secondary school to 3, undergraduate degree.

1.1.10.3 Proportion in each category

To describe the distribution of our nominal variable, support for remaining in the European Union, we use the proportions in each category.

Recall, that we looked at the frequency table to determine the mode:

```
table(stay)
```

```
stay
  0   1
509 491
```

To get the proportions in each category, we divide the values in the table, i.e., 509 and 491, by the sum of the table, i.e., 1000.

```
table(stay) / sum(table(stay))
```

```
stay
    0     1
0.509 0.491
```

1.1.11 Exercises

1. Create a script and call it assignment01. Save your script.
2. Download this cheat-sheet and go over it. You won't understand most of it right a away. But it will become a useful resource. Look at it often.
3. Calculate the square root of 1369 using the `sqrt()` function.
4. Square the number 13 using the `^` operator.
5. What is the result of summing all numbers from 1 to 100?

We take a sample of yearly income in Berlin. The values that we got are: 19395, 22698, 40587, 25705, 26292, 42150, 29609, 12349, 18131, 20543, 37240, 28598, 29007, 26106, 19441, 42869, 29978, 5333, 32013, 20272, 14321, 22820, 14739, 17711, 18749.

6. Create the variable `income` with the values form our Berlin sample in R.
7. Describe Berlin income using the appropriate measures of central tendency and dispersion.
8. Compute the average deviation without using the `sd()` function.

Take a look at the Sunday Question (who would you vote for if the general election were next Sunday?) by following this link Sunday Question Germany. You should be able to translate the website into English by right clicking in your browser and clicking "Translate to English."

9. What is the level of measurement of the variable in the Sunday Question?
10. Take the most recent poll and describe what you see in terms of central tendency and dispersion.
11. Save your script, which should now include the answers to all the exercises.
12. Source your script, i.e. run the entire script without error message. Clean your script if you get error messages.

1.2 Solutions

1.2.1 Exercise 3

Calculate the square root of 1369 using the `sqrt()` function.

```
sqrt(1369)
```

```
[1] 37
```

1.2.2 Exercise 4

Square the number 13 using the ^ operator.

```
13^2
```

```
[1] 169
```

1.2.3 Exercise 5

What is the result of summing all numbers from 1 to 100?

```
# sequence of numbers from 1 to 100 in steps of 1
numbers_1_to_100 <- seq(from = 1, to = 100, by = 1)
# sum over the vector
result <- sum(numbers_1_to_100)
# print the result
result
```

```
[1] 5050
```

The result is 5050.

1.2.4 Exercise 6

Create the variable *income* with the values form our Berlin sample in R.

```r
# create the income variable using the c() function
income <- c(
  19395, 22698, 40587, 25705, 26292, 42150, 29609, 12349, 18131,
  20543, 37240, 28598, 29007, 26106, 19441, 42869, 29978, 5333,
  32013, 20272, 14321, 22820, 14739, 17711, 18749
)
```

1.2.5 Exercise 7

Describe Berlin income using the appropriate measures of central tendency and dispersion.

We use the mean for the central tendency of *income*. The variable is interval scaled and the mean is the appropriate measure of central tendency for interval scaled variables. Our *income* variable is also normally distributed. Income distributions in most countries are right skewed. Therefore, the central tendency of income is often described using the median.

When asked, e.g., in an exam, to describe the central tendency of an interval scaled variable, use the mean. You can also use the median if you tell us why.

```r
# central tendency of income
mean(income)
```

```
[1] 24666.24
```

```r
# dispersion
sd(income)
```

```
[1] 9467.383
```

Average income in our Berlin sample is 24666.24. The average difference from that value is 9467.38.

1.2.6 Exercise 8

Compute the average deviation without using the sd() function.

We do this in several steps. First, we compute the mean.

```r
mean.income <- sum(income) / length(income)

# let's print the mean
mean.income
```

```
[1] 24666.24
```

Second, we take the differences between each individual realisation of income and the mean of *income*. The result must be a vector with the same amount of elements as the *income* vector.

```r
# individual differences between each realisation of income and the mean of income
diffs.from.mean <- income - mean.income

# let's print the vector of differences
diffs.from.mean
```

```
 [1]  -5271.24  -1968.24  15920.76   1038.76   1625.76  17483.76   4942.76
 [8] -12317.24  -6535.24  -4123.24  12573.76   3931.76   4340.76   1439.76
[15]  -5225.24  18202.76   5311.76 -19333.24   7346.76  -4394.24 -10345.24
[22]  -1846.24  -9927.24  -6955.24  -5917.24
```

You may be surprised that this works. After all, *income* is a vector with 25 elements and *mean.income* is a scalar (only one value). R treats all variables as vectors. It notices that *mean.income* is a shorter vector than *income*. The former has 1 element and the latter 25. The vector *mean.income* is recycled, so that it has the same length as *income* where each element is the same: the mean of *income*. If you did not understand this don't worry. The important thing is that it works.

Our next step is to square the differences from the mean.

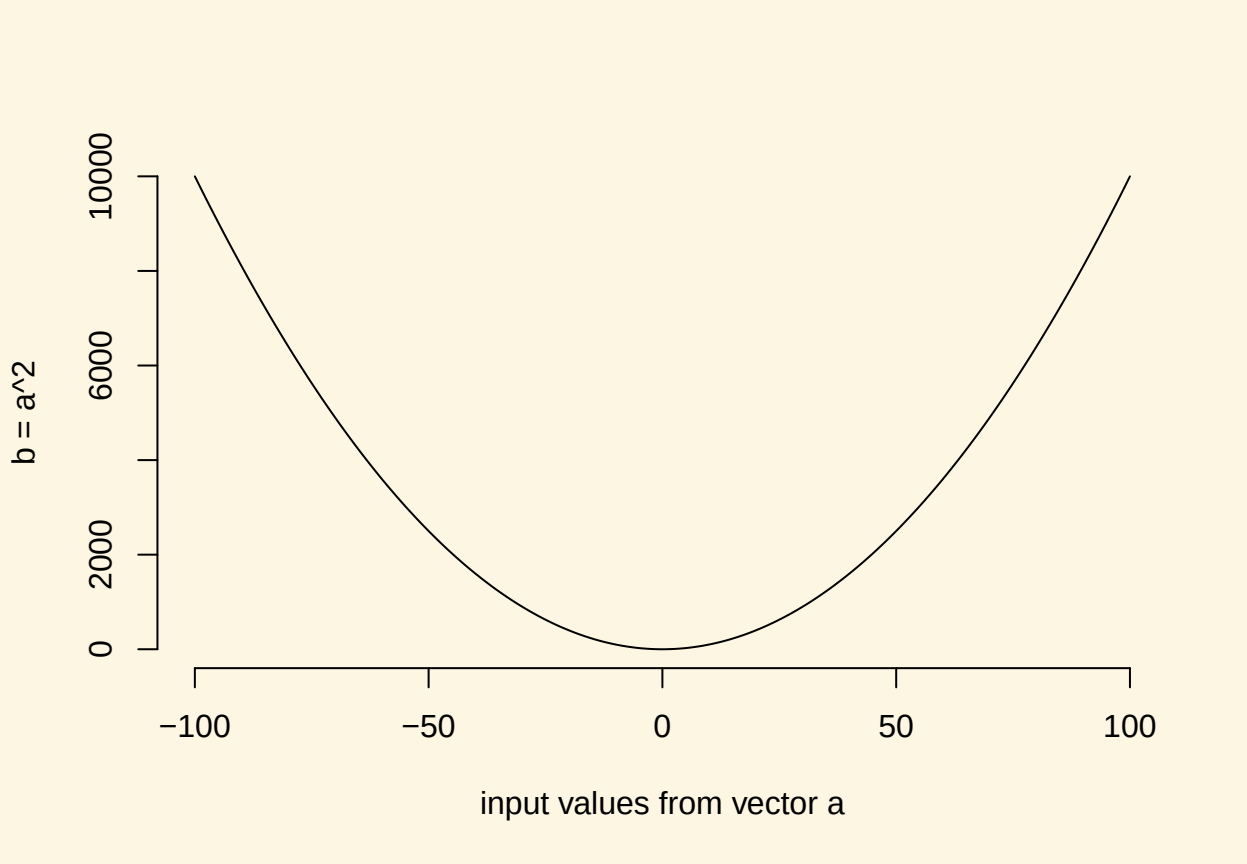
```
# square each element in the diffs.from.mean vector
squared.diffs.from.mean <- diffs.from.mean^2

# print the squared vecto
squared.diffs.from.mean
```

```
 [1]  27785971   3873969 253470599   1079022   2643096 305681864  24430876
 [8] 151714401  42709362  17001108 158099441  15458737  18842197   2072909
[15]  27303133 331340472  28214794 373774169  53974882  19309345 107023991
[22]   3408602  98550094  48375363  35013729
```

We squared each individual element in the vector. Therefore, our new variable *squared.diffs.from.mean* still has 25 elements.

Squaring a value does two things. First, all values in our vector have become positive. Second, the marginal increase increases with distance, i.e., values that are close to the mean are only somewhat larger whereas values that are further from the mean become way larger. To see this, lets plot the square (we haven't shown you the plot function yet, but we will do this next seminar).

```
# a vector of x values from negative 100 to positive 100
a <- seq(from = -100, to = 100, length.out = 200)

# the square of that vector
b <- a^2

# we plot the input vector a against b, where b is on the y-axis
plot(
  x = a, # x-axis values
  y = b, # y-axis values
  bty = "n", # no border around plot
  type = "l", # connect individual dots to a line
  xlab = "input values from vector a", # x axis label
  ylab = "b = a^2" # y axis label
)
```



In this plot, you should see that the slope of the line increases, the further we are from 0. We are taking individual differences from the mean. Hence, if a value is exactly at the mean, the difference is zero. The further, the value is from the mean (in any direction), the larger the output value.

We will sum over the individual elements in the next step. Hence, values that are further from the mean have a larger impact on the sum than values that are closer to the mean.

In the next step, we take the sum over our squared deviations from the mean

```
# sum over squared deviations vector
sum.of.squared.deviations <- sum(squared.diffs.from.mean)

# print the sum
sum.of.squared.deviations
```

```
[1] 2151152127
```

By summing over all elements of a vector, we end up with a scalar. The sum is 2151152126.56.

We divide the sum of squared deviations by $n - 1$. Recall, that n is the number of observations (elements in the vector) and -1 is our sample adjustment.

```
# get the variance
var.income <- sum.of.squared.deviations / ( length(income) - 1 )

# print the variance
var.income
```

```
[1] 89631339
```

The squared average deviation from mean income is 89631338.61.

In the last step, we take the square root over the variance to return to our original units of income.

```
# get the standard deviation
sqrt(var.income)
```

```
[1] 9467.383
```

The average deviation from mean income in Berlin (24666.24) is 9467.38.

1.2.7 Exercise 9

What is the level of measurement of the variable in the Sunday Question?

The variable measures vote choice. The answers are categories, the parties, without any specific ordering. The level of measurement is called categorical or nominal.

1.2.8 Exercise 10

Take the most recent poll and describe what you see in terms of central tendency and dispersion.

The most recent poll was carried out by Infratest/dimap on Thursday, 6 September. The most common value, the mode, is the appropriate measure of central tendency. Christian Democrat (CDU/CSU) is the modal category. Dispersion of a categorical variable is the proportion in each category which we see displayed on the website:

Party	Proportion
CDU/CSU	0.29
SPD	0.18
GREEN	0.14
FDP	0.08
THE LEFT	0.10
AFD	0.16
other	0.05

2 Research Design, Counterfactuals, Forming Hypotheses

2.1 Seminar

In today's seminar, we work with data frames (datasets). We will create our own dataset, we subset datasets (access elements, rows and variables). We load our first dataset into R. We also visualise data using the `plot()` function. Finally, we estimate a treatment effect in R—our first inference.

2.1.1 setting up

We set our working directory. R operates in specific directory (folder) on our computer. We create a folder on our computer where we save our scripts for our statistics 1 class. We name the folder `stats1`. Let's create the folder on our computers now (in finder on Mac and explorer on Windows).

Now, we set our working directory to the folder, we just created like so:

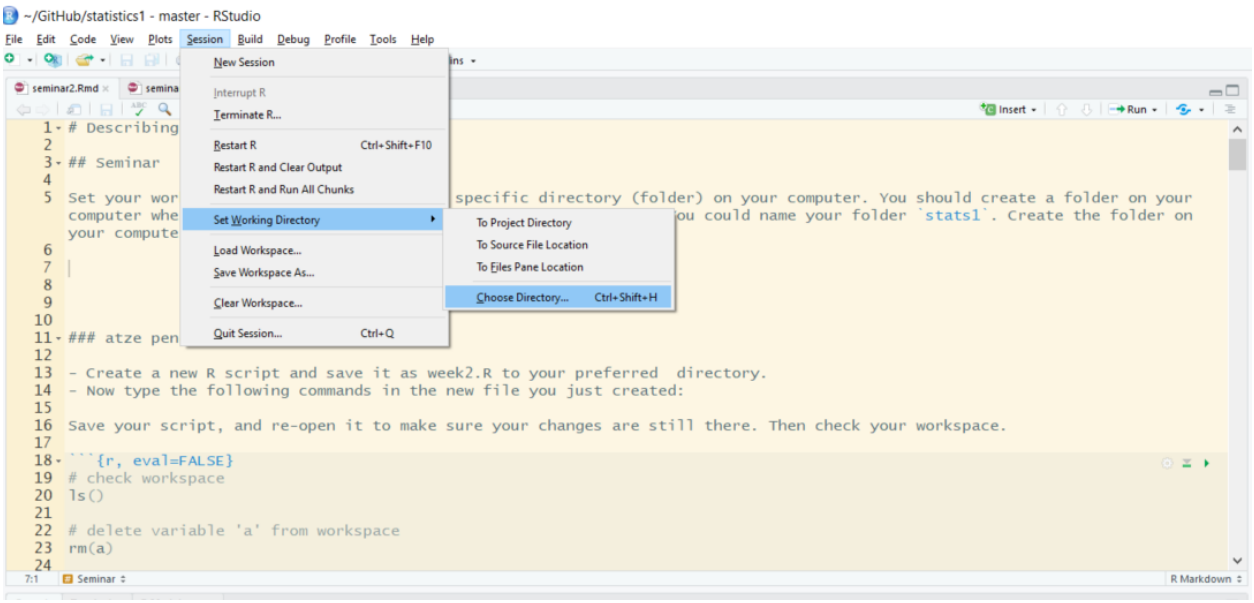


Figure 5:

Create a new R script and save it as week2.R to your `stats1` directory. Now type the following commands in the new file you just created:

```
# Create a numeric and a character variable
a <- 5 # numeric
a <- "five" # character
```

Save your script, and re-open it to make sure your changes are still there. Then check your workspace.

```
# check workspace
ls()

# delete variable 'a' from workspace
rm(a)

# delete everything from workspace
rm( list = ls() )

# to clear console window press Crtl+l on Win or Command+l on Mac
```

2.1.2 vectors and subsetting

Last week we have already worked with vectors. We created a sequence for example. This week, we learn about subsetting (accessing specific elements of our vector).

We create a vector using the `c()` function, where c stands for collect.

```
# Create a vector
my.vector <- c(10,7,99,34,0,5) # a vector
my.vector
```

```
[1] 10  7 99 34  0  5
```

Let's see how many elements our vector contains using the `length()` function.

```
length(my.vector) # how many elements?
```

```
[1] 6
```

Next, we access the first element in our vector. We use square brackets to access a specific element. The number in the square brackets is the vector element that we access

```
# subsetting
my.vector[1] # 1st vector element
```

```
[1] 10
```

To access all elements except the first element, we use the - operator.

```
my.vector[-1] # all elements but the 1st
```

```
[1]  7 99 34  0  5
```

We can access elements 2 to 4 by using the colon.

```
my.vector[2:4] # the 2nd to the 4th elements
```

```
[1]  7 99 34
```

We can access two specific non-adjacent elements, by using the collect function `c()`.

```
my.vector[c(2,5)] # 2nd and 5th element
```

```
[1] 7 0
```

No, we combine the `length()` function with the square brackets to access the last element in our vector.

```
my.vector[length(my.vector)] # the last element
```

```
[1] 5
```

2.1.3 data frames

A data frame is an object that holds data in a tabular format similar to how spreadsheets work. Variables are generally kept in columns and observations are in rows.

Before we work with ready-made data, we create a small dataset ourselves. It contains the populations of the sixteen German states. We start with a vector that contains the names of those states. We call the variable *state*. Our variable shall contain text instead of numbers. In R jargon, this is a character variable, sometimes referred to as a string. Using quotes, we indicate that the variable type is character. We use the `c()` function to create the vector.

```
# create a character vector containing state names
state <- c(
  "North Rhine-Westphalia",
  "Bavaria",
  "Baden-Wurttemberg",
  "Lower Saxony",
  "Hesse",
  "Saxony",
  "Rhineland-Palatinate",
  "Berlin",
  "Schleswig-Holstein",
  "Brandenburg",
  "Saxony-Anhalt",
```

```
  "Thuringia",
  "Hamburg",
  "Mecklenburg-Vorpommern",
  "Saarland",
  "Bremen"
  )
```

Now, we create a second variable for the populations. This is a numeric vector, so we do not use the quotes.

```
population <- c(
  17865516,
  12843514,
  10879618,
  7926599,
  6176172,
  4084851,
  4052803,
  3670622,
  2858714,
  2484826,
  2245470,
  2170714,
  1787408,
  1612362,
  995597,
  671489
)
```

Now with both vectors created, we combine them into a dataframe. We put our vectors in and give them names. In this case the variable names in the dataset correspond to our vector names. The name goes in front of the equal sign and the vector object name, after.

```
popdata <- data.frame(
  state = state,
  population = population
  )
```

You should see the new data frame object in your global environment window. You can view the dataset in the spreadsheet form that we are all used to by clicking on the oject name.

We can see the names of variables in our dataset with the names function

```
names(popdata)
```

```
[1] "state"      "population"
```

Let's check the variable types in our data using the `str()` function.

```
str(popdata)
```

```
'data.frame':   16 obs. of  2 variables:
 $ state     : Factor w/ 16 levels "Baden-Wurttemberg",..: 10 2 1 8 7 13 11 3 15 4 ...
 $ population: num  17865516 12843514 10879618 7926599 6176172 ...
```

The variable *state* is a factor variable. R has turned the character variable into a categorical variable automatically. The variable *population* is numeric. These variable types differ. We can calculate with numeric variables only.

Often we want to access certain observations (rows) or certain columns (variables) or a combination of the

two without looking at the entire dataset all at once. We can use square brackets to subset data frames. In square brackets we put a row and a column coordinate separated by a comma. The row coordinate goes first and the column coordinate second. So `popdata[10, 2]` returns the 10th row and second column of the data frame. If we leave the column coordinate empty this means we would like all columns. So, `popdata[10,]` returns the 10th row of the dataset. If we leave the row coordinate empty, R returns the entire column. `popdata[,2]` returns the second column of the dataset.

We can look at the first five rows of a dataset to get a better understanding of it with the colon in brackets like so: `popdata[1:5,]`. We could display the second and fifth columns of the dataset by using the `c()` function in brackets like so: `popdata[, c(2,5)]`.

It's your turn. Display all columns of the popdata dataset and show rows 10 to 15. Next display all columns of the dataset and rows 4 and 7.

```
popdata[10:15, ] # elements in 10th to 15th row, all columns
```

```
                   state population
10            Brandenburg    2484826
11          Saxony-Anhalt    2245470
12              Thuringia    2170714
13                Hamburg    1787408
14 Mecklenburg-Vorpommern    1612362
15               Saarland     995597
```

```
popdata[c(4, 7), ] # elements in 4th and 7th row, all column
```

```
                 state population
4         Lower Saxony    7926599
7 Rhineland-Palatinate    4052803
```

In order to access individual columns of a data frame we can also use the dollar sign `$`. For example, let's see how to access the `population` column.

```
popdata$population
```

```
 [1] 17865516 12843514 10879618  7926599  6176172  4084851  4052803
 [8]  3670622  2858714  2484826  2245470  2170714  1787408  1612362
[15]   995597   671489
```

Now, access the state column.

```
popdata$state
```

```
 [1] North Rhine-Westphalia Bavaria                Baden-Wurttemberg
 [4] Lower Saxony           Hesse                  Saxony
 [7] Rhineland-Palatinate   Berlin                 Schleswig-Holstein
[10] Brandenburg            Saxony-Anhalt          Thuringia
[13] Hamburg                Mecklenburg-Vorpommern Saarland
[16] Bremen
16 Levels: Baden-Wurttemberg Bavaria Berlin Brandenburg Bremen ... Thuringia
```

2.1.4 Loading data

Before you load the dataset into R, you first download it and save it locally in your `Stats1` folder. Download the data here.

We often load existing data sets into R for analysis. Data come in many different file formats such as `.csv`, `.tab`, `.dta`, etc. Today we will load a dataset which is stored in R's native file format: `.RData`. The function

to load data from this file format is called: `load()`. If you managed to set your working directory correctly just now (`setwd("~/Stats1")`), then you should just be able to run the line of code below.

We load the dataset with the `load()` function:

```
# load perception of non-western foreigners data
load("BSAS_manip.RData")
```

The non-western foreingers data is about the subjective perception of immigrants from non-western countries. The perception of immigrants from a context that is not similar to the one's own ,is often used as a proxy for racism. Whether this is a fair measure or not is debatable but let's examine the data from a survey carried out in Britain.

Let's check the codebook of our data.

Variable	Description
IMMBRIT	Out of every 100 people in Britain, how many do you think are immigrants from non-western countr
over.estimate	1 if estimate is higher than 10.7%.
RSex	1 = male, 2 = female
RAge	Age of respondent
Househld	Number of people living in respondent's household
party identification	1 = Conservatives, 2 = Labour, 3 = SNP, 4 = Greens, 5 = Ukip, 6 = BNP, 7 = other
paper	Do you normally read any daily morning newspaper 3+ times/week?
WWWhourspW	How many hours WWW per week?
religious	Do you regard yourself as belonging to any particular religion?
employMonths	How many mnths w. present employer?
urban	Population density, 4 categories (highest density is 4, lowest is 1)
health.good	How is your health in general for someone of your age? (0: bad, 1: fair, 2: fairly good, 3: good)
HHInc	Income bands for household, high number = high HH income

We can look at the variable names in our data with the `names()` function.

The `dim()` function can be used to find out the dimensions of the dataset (dimension 1 = rows, dimension 2 = columns).

```
dim(data2)
```

```
[1] 1049   19
```

So, the `dim()` function tells us that we have data from 1049 respondents with 19 variables for each respondent.

Let's take a quick peek at the first 10 observations to see what the dataset looks like. By default the `head()` function returns the first 6 rows, but let's tell it to return the first 10 rows instead.

```
head(data2, n = 10)
```

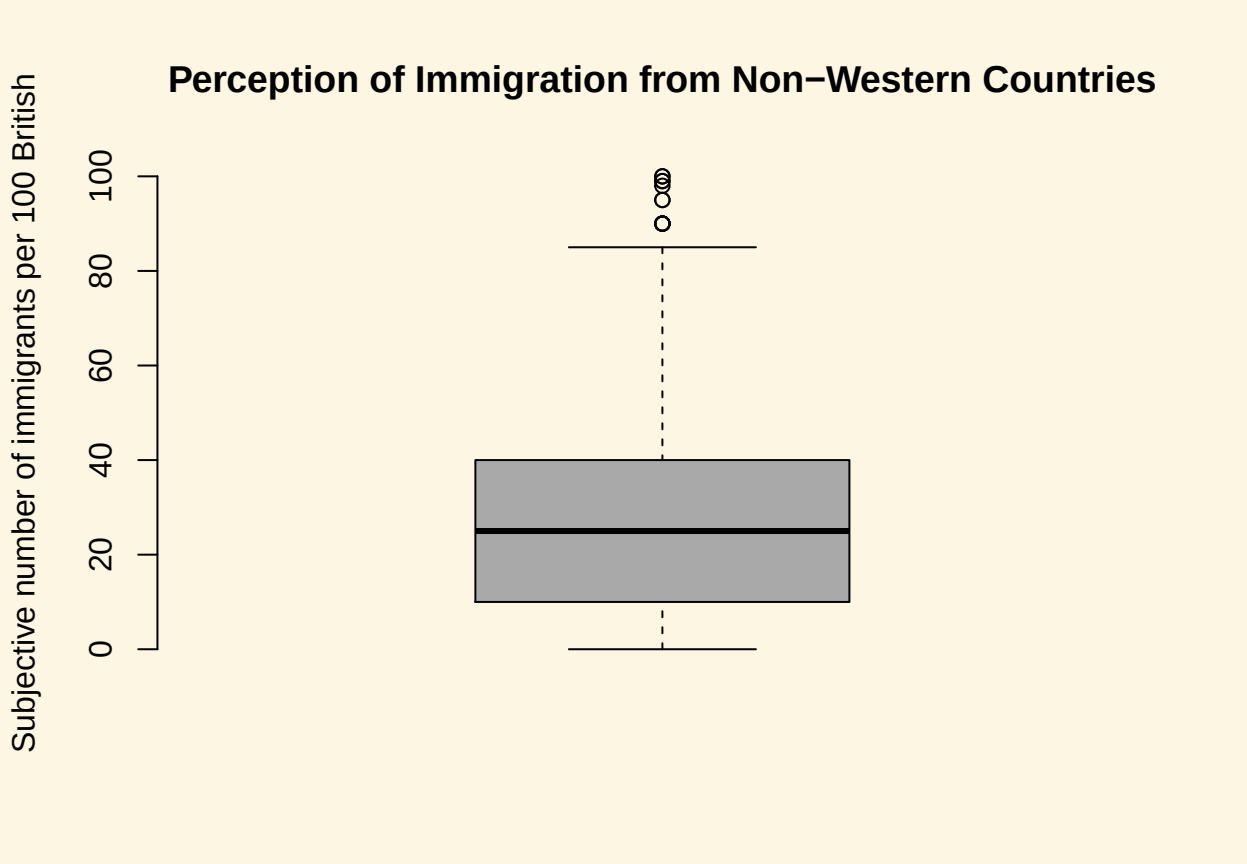
```
   IMMBRIT over.estimate RSex RAge Househld Cons Lab SNP Ukip BNP GP
1        1             0    1   50        2    0   1   0    0   0  0
2       50             1    2   18        3    0   0   0    0   0  0
3       50             1    2   60        1    0   0   0    0   0  0
4       15             1    2   77        2    0   0   0    0   0  0
5       20             1    2   67        1    0   0   0    0   0  0
6       30             1    1   30        4    0   0   0    0   0  0
7       60             1    2   56        2    0   0   1    0   0  0
8        7             0    1   49        1    0   0   0    0   0  0
9       30             1    1   40        4    0   0   1    0   0  0
10       2             0    1   61        3    1   0   0    0   0  0
   party.other paper WWWhourspW religious employMonths urban health.good
1            0     0          1         0           72     4           1
2            1     0          4         0           72     4           2
```

```
3            1     0          1          0          456     3           3
4            1     1          2          1           72     1           3
5            1     0          1          1           72     3           3
6            1     1         14          0           72     1           2
7            0     0          5          1          180     1           2
8            1     1          8          0          156     4           2
9            0     0          3          1          264     2           2
10           0     1          0          1           72     1           3
   HHInc
1     13
2      3
3      9
4      8
5      9
6      9
7     13
8     14
9     11
10     8
```

2.1.5 Plots

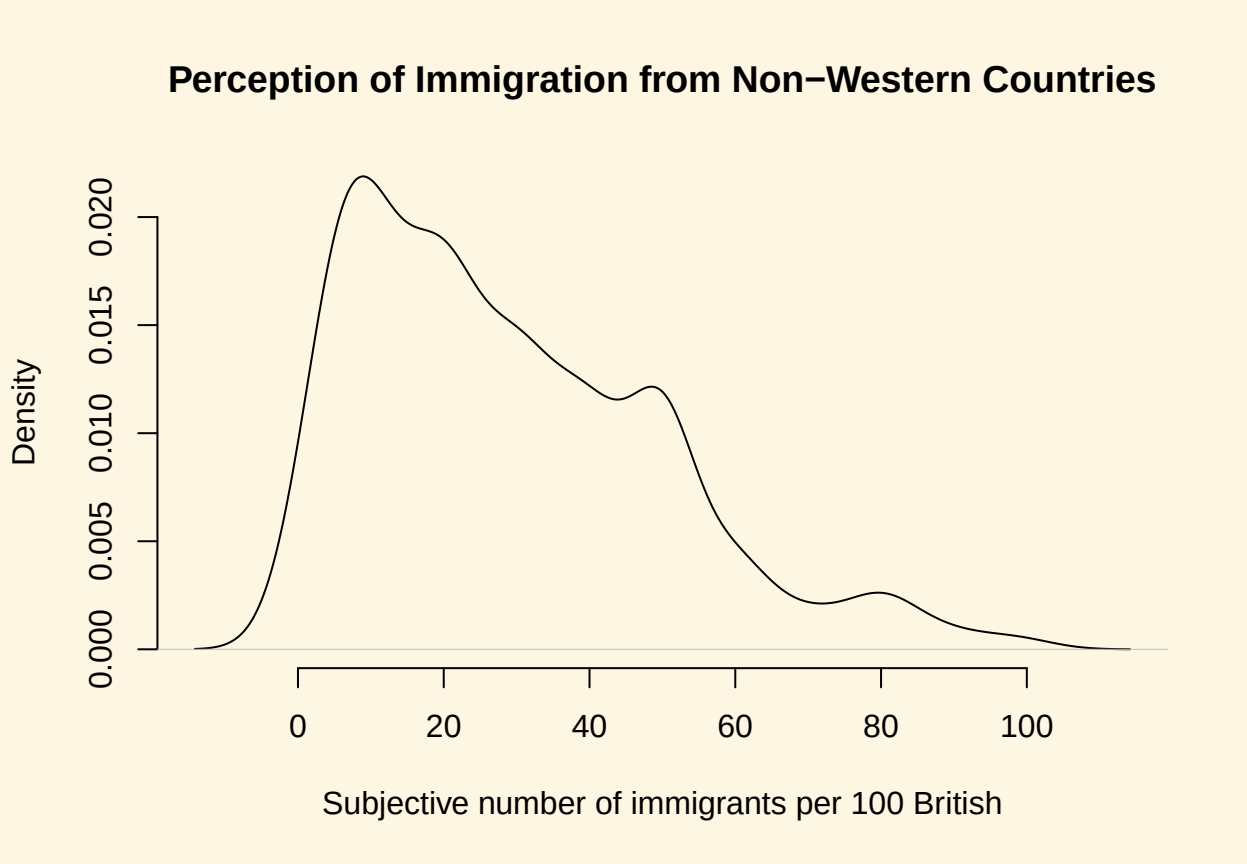
We can visualize the data with the help of a boxplot, so let's see how the perception of the number of immigrants is distributed.

```r
# how good are we at guessing immigration
boxplot(
  data2$IMMBRIT,
  main = "Perception of Immigration from Non-Western Countries",
  ylab = "Subjective number of immigrants per 100 British",
  frame.plot = FALSE, col = "darkgray"
  )
```



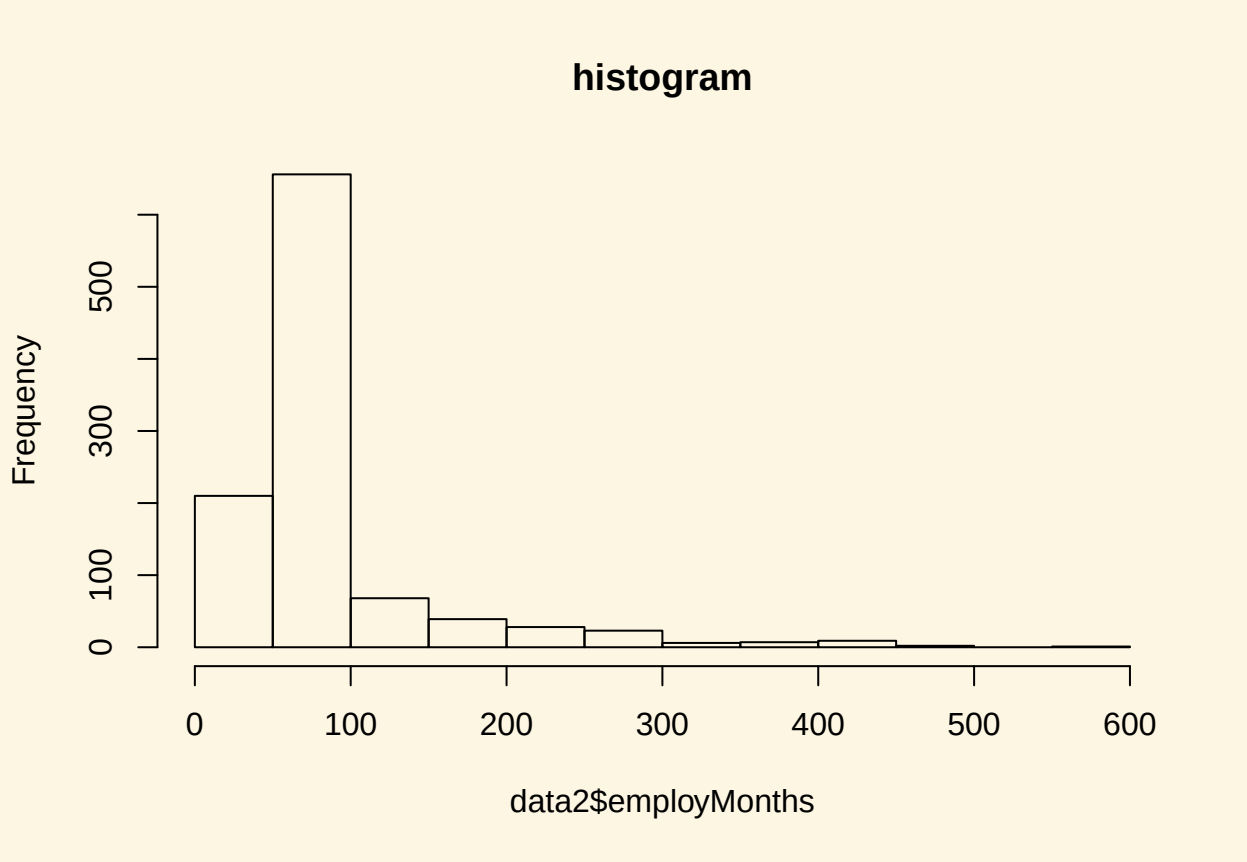
Notice how the lower whisker is much shorter than the upper one. The distribution is right skewed. The right tail (higher values) is a lot longer. We can see this beter using a density plot. We combine R's `denisty()` function with the `plot()` function.

```
plot(
  density(data2$IMMBRIT),
  bty = "n",
  main = "Perception of Immigration from Non-Western Countries",
  xlab = "Subjective number of immigrants per 100 British"
  )
```



We can also plot histograms using the `hist()` function.

```
# histogram
hist( data2$employMonths, main = "histogram")
```



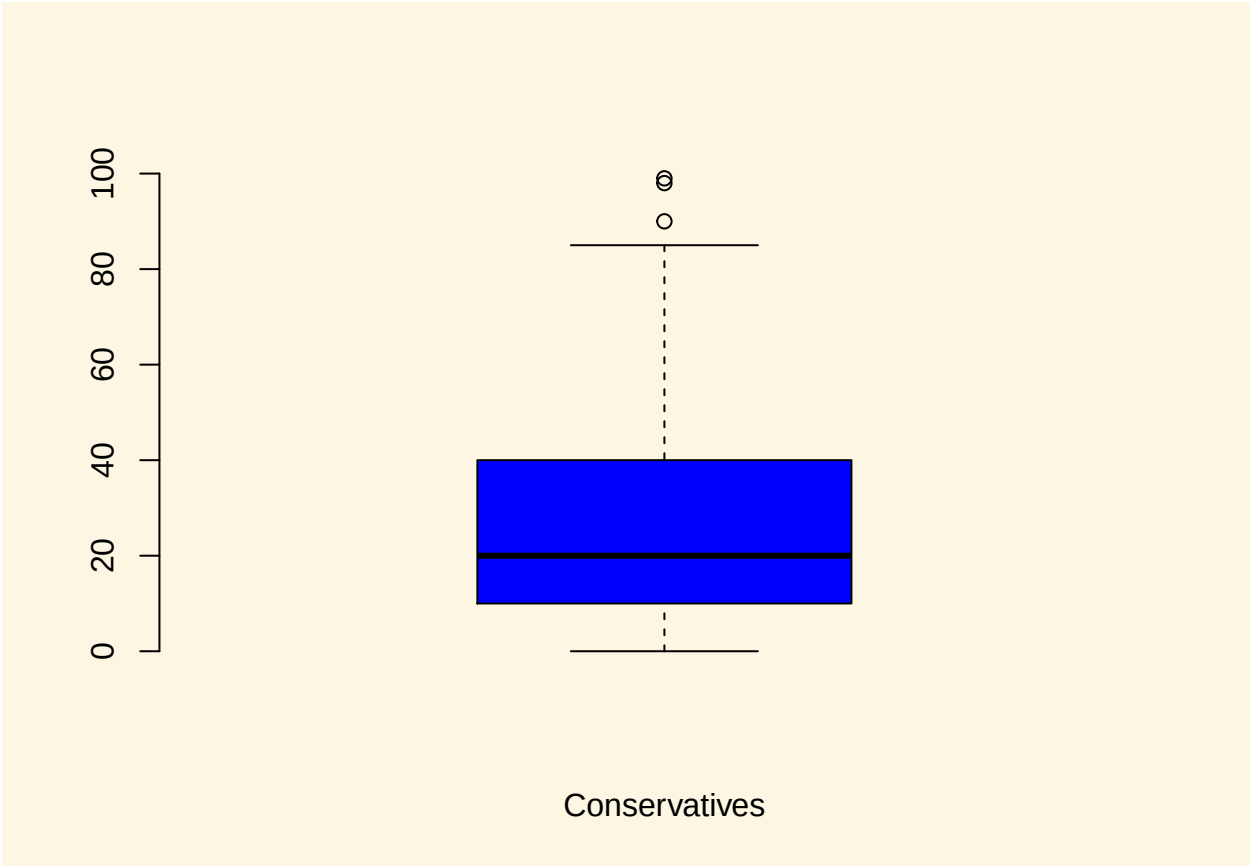
It is plausible that perception of immigration from Non-Western countries is related to party affiliation. In our dataset, we have a some party affiliation dummies (binary variables). We can use square brackets to subset our data such that we produce a boxplot only for members of the Conservative Party. We have a look at the variable *Cons* using the `table()` function first.

```
table(data2$Cons)
```

```
  0   1
765 284
```

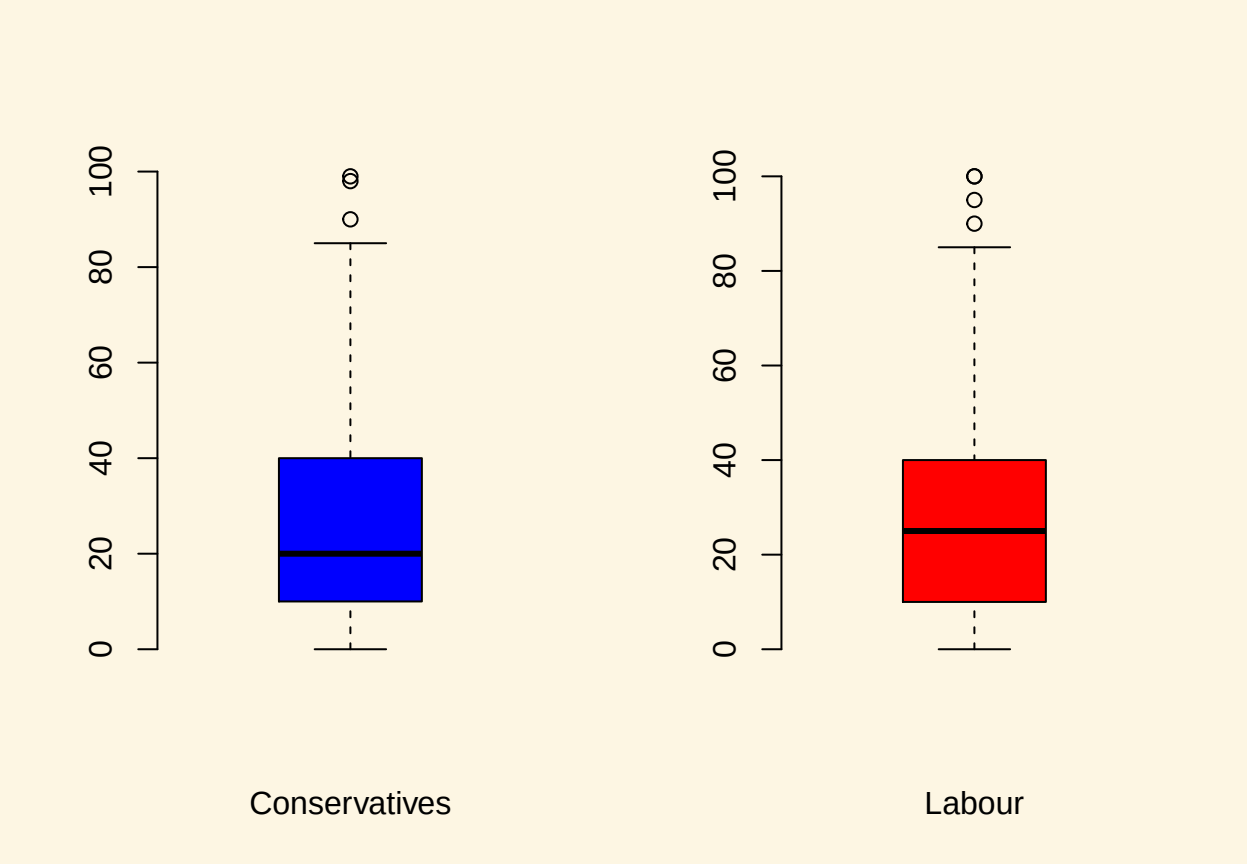
In our data, 284 respondents associate with the Conservative party and 765 do not. We create a boxplot of *IMMBRIT* but only for members of the Conservative Party. We do so by using the square brackets to subset our data.

```
# boxplot of immbrit for those observations where Cons is 1
boxplot(
  data2$IMMBRIT[data2$Cons==1],
  frame.plot = FALSE,
  xlab = "Conservatives",
  col = "blue"
  )
```



We would now like to compare the distribution of the perception fo Conservatives to the distribution among Labour respondents. We can subset the data just like we did for the Conservative Party. In addtion, we want to plot the two plots next to each other, i.e., they should be in the same plot. We can achieve this with the `par()` function and the `mfrow` argument. This will spilt the plot window into rows and columns. We want 2 columns to plot 2 boxplots next to each other.

```
# split plot window into 1 row and 2 columns
par(mfrow = c(1,2))

# plot 1
boxplot(
  data2$IMMBRIT[data2$Cons==1],
  frame.plot = FALSE,
  xlab = "Conservatives",
  col = "blue"
  )

# plot 2
boxplot(
  data2$IMMBRIT[data2$Lab==1],
  frame.plot = FALSE,
  xlab = "Labour",
  col = "red"
  )
```



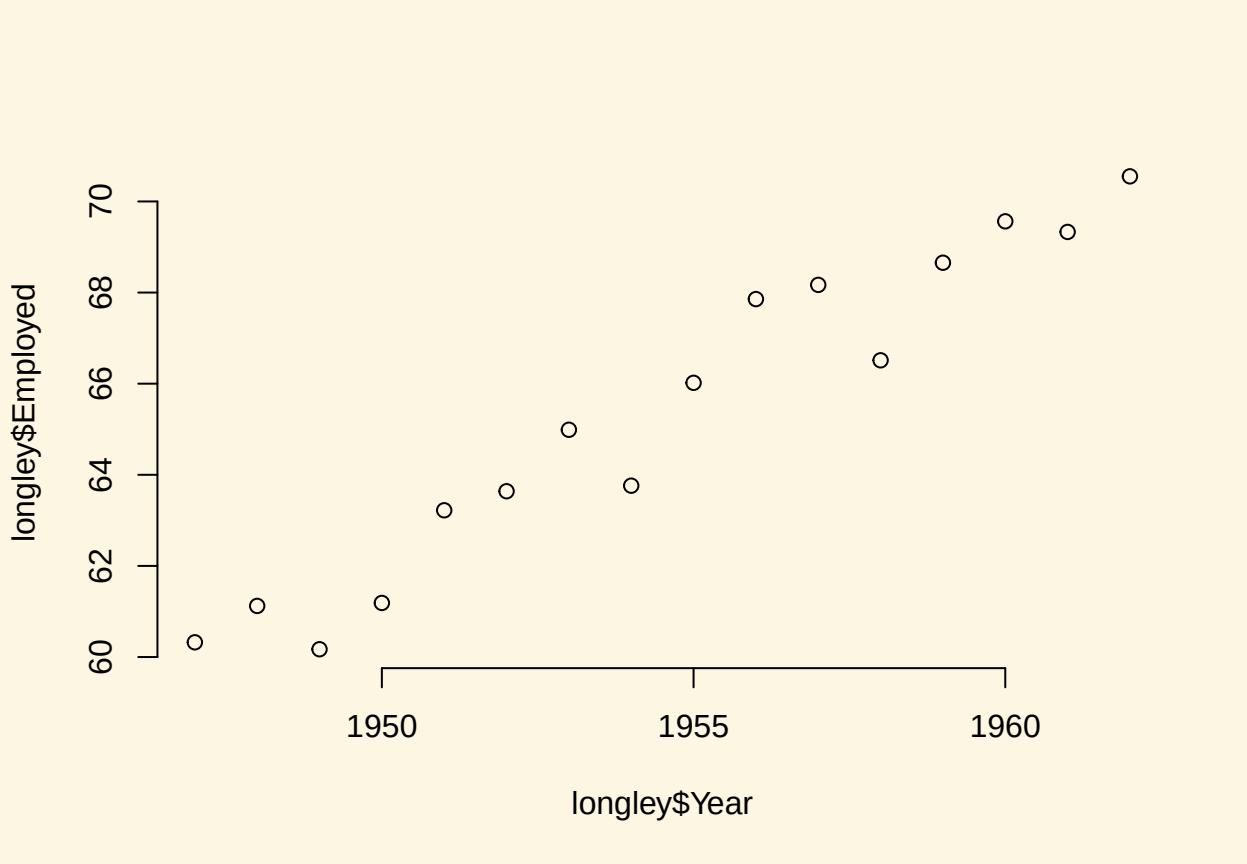
It is very hard to spot differences. The distributions are similar. The median for Labour respondents is larger which mean that the central Labour respondent over-estimates immigration more than the central Conservative respondent.

You can play around with the non-western foreigners data on your own time. We now turn to a dataset that is integrated in R already. It is called `longley`. Use the `help()` function to see what this dataset is about.

```
help(longley)
```

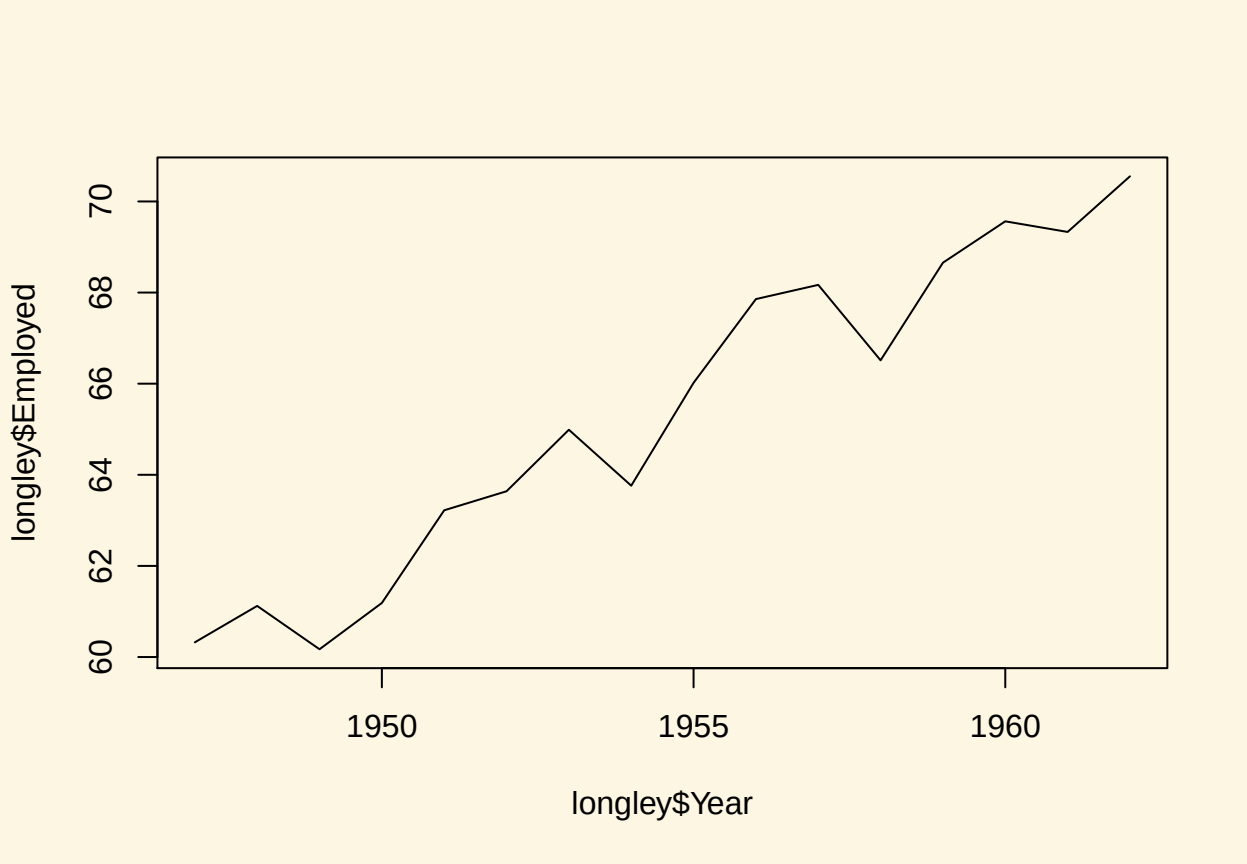
Let's create a scatterplot with the `Year` variable on the x-axis and `Employed` on the y-axis.

```
plot(x = longley$Year, # x-axis variable
     y = longley$Employed, # y-axis variable
     bty = "n" # no box around the plot
     )
```



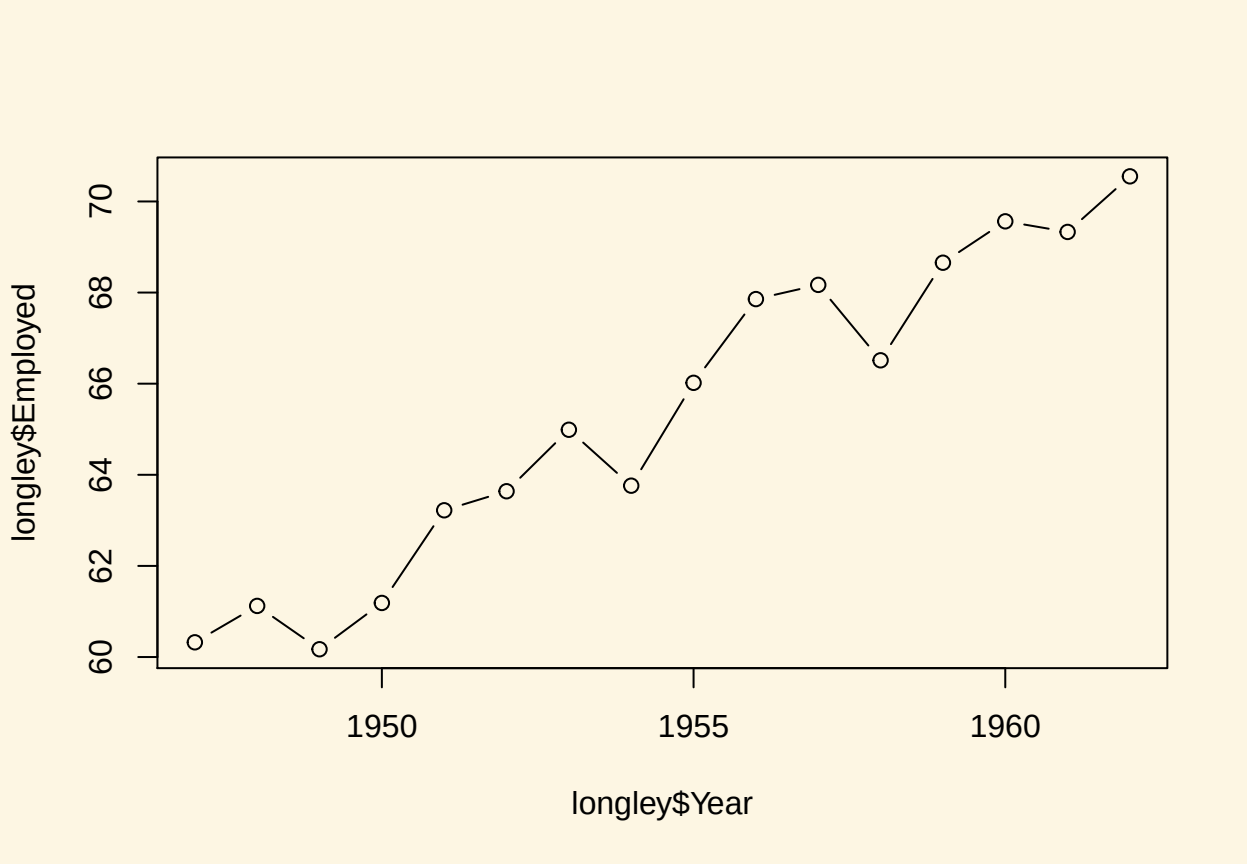
To create a line plot instead, we use the same function with one additional argument `type = "l"`.

```
plot(longley$Year, longley$Employed, type = "l")
```



Create a plot that includes both points and lines.

```
plot(longley$Year, longley$Employed, type = "b")
```



2.1.6 Average Treatment Effect

In the lecture, we estimated the average treatment effect on a small example. We will do this again here. Recall, that the average treatment effect is the difference between two means.

Let's suppose, associating with right-wing parties causes people to over-estimate the number of non-western foreigners. Our treatment variable is whether a respondent assoicates with the UK Independence Party. It is 1 if that is the case and 0 otherwise. Let's inspect the variable *Ukip*.

```
table(data2$Ukip)
```

```
   0    1
1018   31
```

31 respondents identify with Ukip.

The average treatment effect, as we learned, would be the difference between the mean outcomes for those who received the treament minus the mean for those who did not reicive the treatment.

We have all the tools to solve the problem. Let's take the mean of the treated group first.

```
mean.y.treated <- mean(data2$IMMBRIT[data2$Ukip == 1])
mean.y.treated
```

```
[1] 24.29032
```

The double equal sign == is a logical operator and means "is equal to". R returns true or false depending on whether the respondent does identify with Ukip or not. The mean of *IMMBRIT* is then computed only for

respondents who accociate with Ukip.

Let's take the mean of the second group, the untreated group.

```
mean.y.untreated <- mean(data2$IMMBRIT[data2$Ukip == 0])
mean.y.untreated
```

```
[1] 29.17485
```

The treatment effect is the difference in means:

```
mean.y.treated - mean.y.untreated
```

```
[1] -4.88453
```

The result is surprising. Ukip members over-estimate the number of non-western foreigners less members of all other paries. Our claim is not quite supported by the data. We should be very careful with these results, however. We used experimental language but our data is observational. A multitude of confounders could bias our estimate of the causal effect.

2.1.7 Exercises

1. Create a script and call it assignment02. Save your script.
2. Use the `names()` function to display the variable names of the `longley` dataset.
3. Use square brackets to access the 4th column of the dataset.
4. Use the dollar sign to access the 4th column of the dataset.
5. Access the two cells from row 4 and column 1 and row 6 and column 3.
6. Using the `longley` data produce a line plot with GNP on the y-axis and population on the x-axis.
7. Use the help function to find out how to label the y-axis "wealth" and the x-axis "population".
8. Create a boxplot showing the distribution of *IMMBRIT* by each party in the data and plot these in one plot next to each other.
9. Is there a difference between women and men in terms of their subjective estimation of foreingers?
10. What is the difference between women and men?
11. Could you form a hypothesis out of the relationship that you see if any exists?
12. Save your script, which should now include the answers to all the exercises.
13. Source your script, i.e. run the entire script without error message. Clean your script if you get error messages.

2.2 Solutions

2.2.1 Exercise 2

Use the `names()` function to display the variable names of the `longley` dataset.

```
names(longley)
```

```
[1] "GNP.deflator" "GNP"          "Unemployed"   "Armed.Forces"
[5] "Population"   "Year"         "Employed"
```

2.2.2 Exercise 3

Use square brackets to access the 4th column of the dataset.

```
longley[, 4]
```

```
 [1] 159.0 145.6 161.6 165.0 309.9 359.4 354.7 335.0 304.8 285.7 279.8
[12] 263.7 255.2 251.4 257.2 282.7
```

2.2.3 Exercise 4

Use the dollar sign to access the 4th column of the dataset.

```
longley$Armed.Forces
```

```
 [1] 159.0 145.6 161.6 165.0 309.9 359.4 354.7 335.0 304.8 285.7 279.8
[12] 263.7 255.2 251.4 257.2 282.7
```

Note: There is yet another way to access the 4th column of the dataset. We can put the variable name into the square brackets using quotes like so:

```
longley[, "Armed.Forces"]
```

```
 [1] 159.0 145.6 161.6 165.0 309.9 359.4 354.7 335.0 304.8 285.7 279.8
[12] 263.7 255.2 251.4 257.2 282.7
```

2.2.4 Exercise 5

Access the two cells from row 4 and column 1 and row 6 and column 3.

```
# row 4, column 1
longley[4, 1]
```

```
[1] 89.5
```

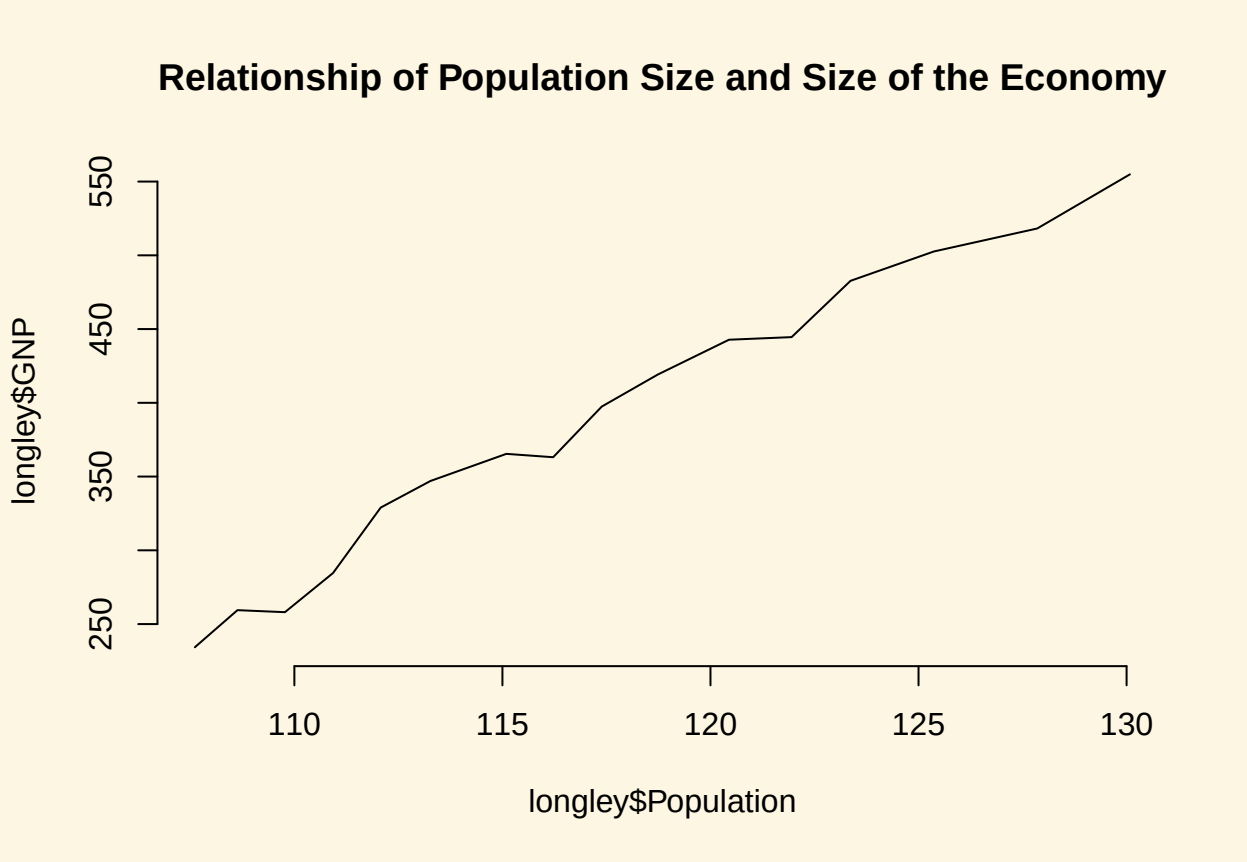
```
# row 6, column 3
longley[6, 3]
```

```
[1] 193.2
```

2.2.5 Exercise 6

Using the `longley` data produce a line plot with GNP on the y-axis and population on the x-axis.

```
plot(
  y = longley$GNP, # y-axis variable
  x = longley$Population, # x-axis variable
  type = "l", # produce a line plot
  bty = "n", # no box around our plot
  main = "Relationship of Population Size and Size of the Economy"
)
```



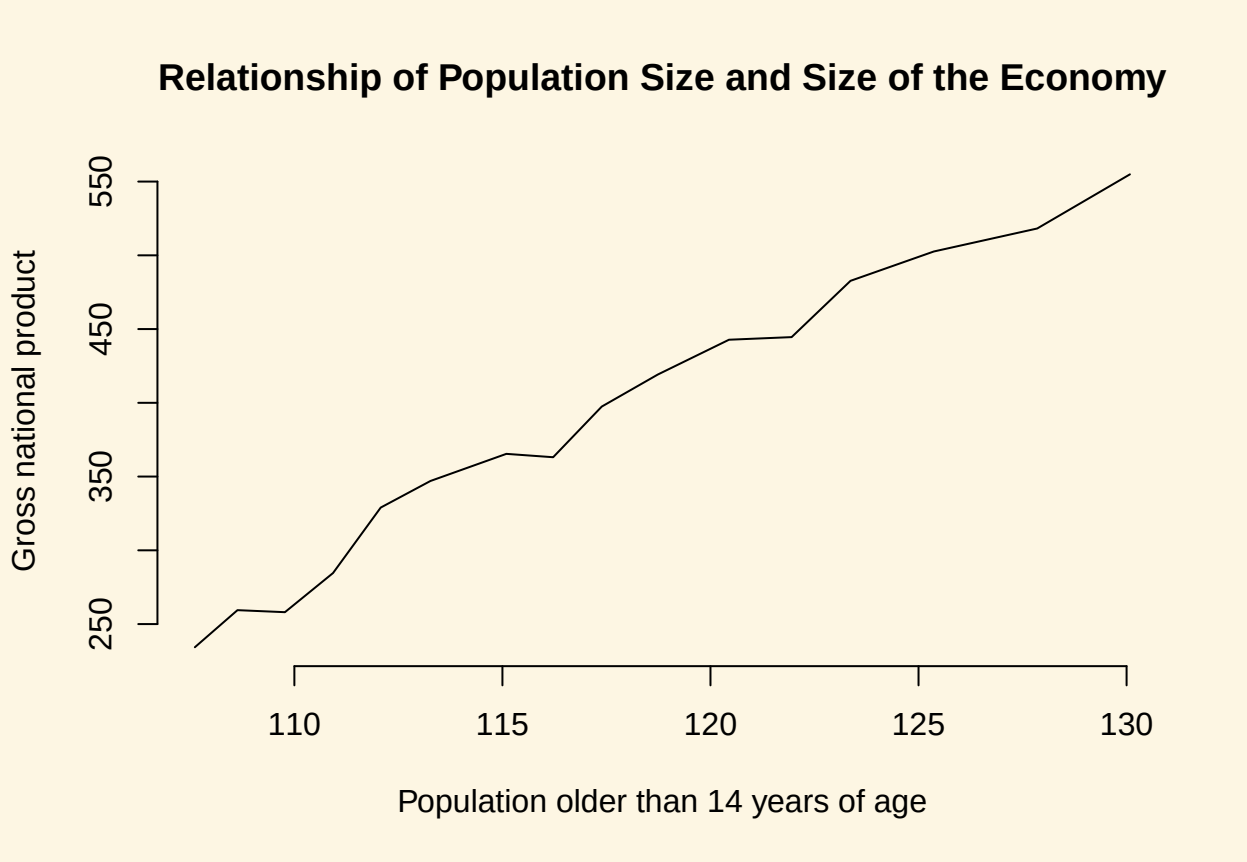
2.2.6 Exercise 7

Use the help function to find out how to label the y-axis "wealth" and the x-axis "population".

```
?plot
```

The `?` is short for the `help()` function. We see that the `xlab` argument lets us label the x-axis and the `ylab` argument lets us label the y-axis. We do so below.

```
plot(
  y = longley$GNP, # y-axis variable
  x = longley$Population, # x-axis variable
  type = "l", # produce a line plot
  bty = "n", # no box around our plot
  main = "Relationship of Population Size and Size of the Economy",
  xlab = "Population older than 14 years of age",
  ylab = "Gross national product"
)
```



2.2.7 Exercise 8

Create a boxplot showing the distribution of *IMMBRIT* by each party in the data and plot these in one plot next to each other.

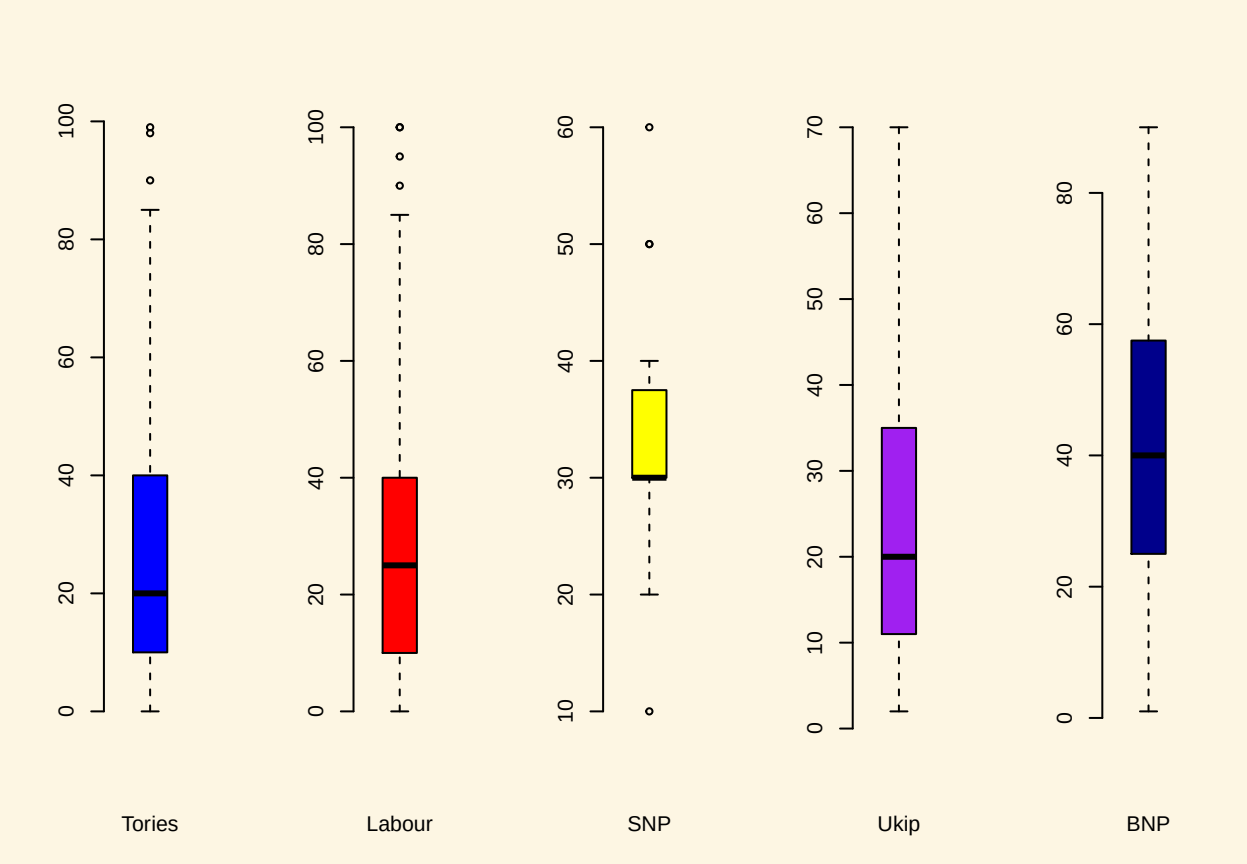
To do that, we load the non-western foreigners dataset first.

Note: You have to set your working directory that R operates in to the location of the dataset.

```
# load perception of non-western foreigners data
load("BSAS_manip.RData")
```

We have five parties in our dataset. We plot 5 boxplots next to each other. Hence, we separate the plot window into 1 row and 5 columns.

```
# plot window to 1 row and 5 columns
par(mfrow = c(1, 5))
boxplot(data2$IMMBRIT[ data2$Cons == 1 ], frame.plot = FALSE, col = "blue", xlab = "Tories")
boxplot(data2$IMMBRIT[ data2$Lab == 1 ], frame.plot = FALSE, col = "red", xlab = "Labour")
boxplot(data2$IMMBRIT[ data2$SNP == 1 ], frame.plot = FALSE, col = "yellow", xlab = "SNP")
boxplot(data2$IMMBRIT[ data2$Ukip == 1 ], frame.plot = FALSE, col = "purple", xlab = "Ukip")
boxplot(data2$IMMBRIT[ data2$BNP == 1 ], frame.plot = FALSE, col = "darkblue", xlab = "BNP")
```



2.2.8 Exercises 9 and 10

We combine the answer to questions 9 and 10.

Question from 9: Is there a difference between women and men in terms of their subjective estimation of foreingers?

Question from 10: What is the difference between women and men?

Women's subjective estimate is the mean of *IMMBRIT* across women and equally, men's subjective estimate is the mean of *IMMBRIT* over all men. Let's get these numbers with the mean function and the square brackets.

```
womens.mean <- mean(data2$IMMBRIT[ data2$RSex == 2 ])
womens.mean
```

```
[1] 32.79159
```

```
mens.mean <- mean(data2$IMMBRIT[ data2$RSex == 1 ])
mens.mean
```

```
[1] 24.53766
```

The difference between women and men is the difference in means. Let's take the difference between them. The difference in means is often referred to as the first difference.

```
first.difference <- womens.mean - mens.mean
first.difference
```

```
[1] 8.253937
```

Let's round that number. We don't like to see so many decimal places. You should usually present precision up to the second decimal place. We can use the `round()` function. The first argument is number to round and the second is the amount of digits.

```
round(first.difference, 2)
```

```
[1] 8.25
```

We do find a difference between men and women. On average, women's estimate of the number of non-western foreingers is 8.25 greater than men's estimate.

At this point we have established that there is a difference in our sample. Samples are subject to sampling variability. That means, we cannot yet say that the difference is systematic, i.e., British women, generally, think that there are more non-western foreingers than British men.

2.2.9 Exercises 11

Could you form a hypothesis out of the relationship that you see if any exists?

Our testable hypothesis could be: Women tend to overestimate the number of foreigners more than men. In our sample, women tend to estimate on the number of foreingers at

3 Sampling and Distributions

3.1 Seminar

In today's seminar, we work with missing data. We will turn a numerical variable into a nominal data type. We then turn to distributions.

```
rm(list=ls())
setwd("~/PUBLG100")
```

3.1.1 Loading Dataset in CSV Format

In this seminar, we load a file in comma separated format (`.csv`). The `load()` function from last week works only for the native R file format. To load our csv-file, we use the `read.csv()` function.

Our data comes from the Quality of Government Institute. Let's have a look at the codebook:

Download the data here

Variable	Description
h_j	1 if Free Judiciary
wdi_gdpc	Per capita wealth in US dollars
undp_hdi	Human development index (higher values = higher quality of life)
wbgi_cce	Control of corruption index (higher values = more control of corruption)
wbgi_pse	Political stability index (higher values = more stable)
former_col	1 = country was a colony once
lp_lat_abst	Latitude of country's captial divided by 90

```
world.data <- read.csv("QoG2012.csv")
```

Go ahead and (1) check the dimensions of `world.data`, (2) the names of the variables of the dataset, (3) print the first six rows of the dataset. (

```r
# the dimensions: rows (observations) and columns (variables)
dim(world.data)
```

```
[1] 194   7
```

```r
# the variable names
names(world.data)
```

```
[1] "h_j"         "wdi_gdpc"    "undp_hdi"    "wbgi_cce"    "wbgi_pse"
[6] "former_col"  "lp_lat_abst"
```

```r
# top 6 rows of the data
head(world.data)
```

```
  h_j   wdi_gdpc undp_hdi   wbgi_cce   wbgi_pse former_col lp_lat_abst
1   0   628.4074       NA -1.5453584 -1.9343837          0   0.3666667
2   0  4954.1982    0.781 -0.8538115 -0.6026081          0   0.4555556
3   0  6349.7207    0.704 -0.7301510 -1.7336243          1   0.3111111
4  NA         NA       NA  1.3267342  1.1980436          0   0.4700000
5   0  2856.7517    0.381 -1.2065741 -1.4150945          1   0.1366667
6  NA 13981.9795    0.800  0.8624368  0.7084046          1   0.1892222
```

3.1.2 Missing Values

Let's inspect the variable *h_j*. It is categorical, where 1 indicates that a country has a free judiciary. We use the `table()` function to find the frequency in each category.

```r
table(world.data$h_j)
```

```
  0   1
105  64
```

We now know that 64 countries have a free judiciary and 105 countries do not.

Conceptually the variable is nominal. To see how the variable is stored in R, we can use the `str()` function.

```r
str(world.data$h_j)
```

```
 int [1:194] 0 0 0 NA 0 NA 0 0 1 1 ...
```

The function returns 'int' which abbreviates 'integer', i.e., a numeric type. The function also shows us the first 10 realisations of the variable. We se zeroes and ones which are the two categories. We also see NA's which abbreviates not available. NAs are missing values. Values can be missing for different reasons. For instance, a coder may have forgotten to code whether a country had been colonised at some point in its history or the country may be new and the categories, therefore, don't apply. It is important for us that we cannot calculate with NAs.

There are different ways of dealing with NAs. We will always delete missing values. Our dataset must maintain its rectangular structure. Hence, when we delete a missing value from one variable, we delete it for the entire row of the dataset. Consider the following example.

Row	Variable1	Variable2	Variable3	Variable4
1	15	22	100	65
2	NA	17	26	75
3	27	NA	58	88
4	NA	NA	4	NA
5	75	45	71	18
6	18	16	99	91

If we delete missing values from *Variable1*, our dataset will look like this:

Row	Variable1	Variable2	Variable3	Variable4
1	15	22	100	65
3	27	NA	58	88
5	75	45	71	18
6	18	16	99	91

The new dataset is smaller than the original one. Rows 2 and 4 have been deleted. When we drop missing values from one variable in our dataset, we lose information on other variables as well. Therefore, you only want to drop missing values on variables that you are interested in. Let's drop the missing values on our variable *h_j*. We do this in several steps.

First, we introduce the `is.na()` function. We supply a vector to the function and it checks for every element, whether it is missing or not. R returns true or false. Let's use the function on our variable.

```
is.na(world.data$h_j)
```

```
  [1] FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE
 [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
 [23]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [34] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [45] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [56] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [67]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [78] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [89] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[100]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
[111] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
[122] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE
[133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
[144] FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE
[155] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
[166] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
[177] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[188] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
```

To see the amount of missingness in the variable *h_j*, we can combine `is.na()` with the `table()` function.

```
table( is.na(world.data$h_j) )
```

```
FALSE  TRUE
  169    25
```

So, we have 25 missing values on *h_j*. Our dataset has 194 rows. Check your global environment to confirm this or use the `nrow()` function. That means, if we drop all missing values from *h_j*, the our dataset *world.data* will lose 25 rows.

Before we drop the missings, we introduce the `which()` function. It returns the row indexes (the rows in the dataset) where some condition is true. So if we use `which()` and `is.na()`, we get the row numbers in the *world.data* dataset where values are missing on *h_j*.

```
which( is.na( world.data$h_j ) )
```

```
 [1]   4   6  11  22  23  67 100 108 112 120 123 129 130 131 141 146 147
[18] 148 149 150 154 165 173 179 191
```

We said that our dataset will lose 25 rows. Let's use the `length()` function to confirm that this is the case.

```
length( which( is.na( world.data$h_j ) ) )
```

```
[1] 25
```

We have, indeed, identified 25 rows that we want to delete from our dataset.

The function `is.na()` returns "TRUE" if an observation is missing. We can use the `!` operator so that the function returns "TRUE" if an observation is **not** missing. The `!` means not.

Let's confirm this:

```
# true = observation is missing
table( is.na(world.data$h_j) )
```

```
FALSE  TRUE
  169    25
```

```
# true = observations is NOT missing
table( !is.na(world.data$h_j) )
```

```
FALSE  TRUE
   25   169
```

We now drop the rows with missings on *h_j* by overwriting our original dataset with a new dataset that is a copy of the old without the missings. We use the square brackets to subset our dataset.

```
world.data <- world.data[ !is.na( world.data$h_j ) , ]
```

Confirm that our new *world.data* dataset has only 169 remaining.

"But what if we want our original dataset back," you ask. We have overwritten the original. It is no longer in our work environment. We have to reload the data set from the disk.

Let's do that:

```
world.data <- read.csv("QoG2012.csv")
```

Right, we now have all observations back. This is important. Let's say we need to drop missings on a variable. We do is. If a later analysis does not involve that variable, we want all the observations back. Otherwise we would have thrown away valuable information. The smaller our dataset, the less information it contains. Less information will make it harder for us to detect systematic correlations. We have to options. Either we reload the original dataset or we create a copy of the original with a different name that we could use later on. Let's do this.

```
full.dataset <- world.data
```

Let's drop missings on *h_j* in the *world.data* dataset.

```
world.data <- world.data[ !is.na( world.data$h_j ) , ]
```

Now, if we want the full dataset back, we can overwrite *world.data* with *full.dataset*. The code would be the following:

```
world.data <- full.dataset
```

If you ran this line. Delete missings from *h_j* in *world.data* again.

This data manipulation may seem boring but it is really important that you know how to do this. Most of the work in data science is not running statistical models but data manipulation. Most of the dataset you will work with in your jobs, as a research assistant or on you dissertation won't be cleaned for you. You will have to do that work. It takes time and is sometimes frustrating. That's unfortunately the same for all of us.

3.1.3 Factor Variables

Categorical/nominal variables can be stored as numeric variables in R. However, the values do not imply an ordering or relative importance. We often store nominal variables as factor variables in R. A factor variable is a nominal data type. The advantage of turning a variable into a factor type is that we can assign labels to the categories and that R will not calculate with the values assigned to the categories.

The function `factor()` lets us turn the variable into a nominal data type. The first argument is the variable itself. The second are the category labels and the third are the levels associated with the categories. To know how those correspond, we have to scroll up and look at the codebook.

We also overwrite the original numeric variable `h_j` with our nominal copy indicated by the assignment arrow `<-`.

```
# factorize judiciary variable
world.data$h_j <- factor(world.data$h_j, labels = c("controlled", "free"), levels = c(0,1))

# frequency table of judiciary variable
table(world.data$h_j)
```

```
controlled       free
       105         64
```

3.1.4 Renaming Variables

We want to rename *h_j* into something more meaningful. The new name should be *free.judiciary*. We can use the `names()` function to get a vector of variable names.

```
names(world.data)
```

```
[1] "h_j"         "wdi_gdpc"    "undp_hdi"    "wbgi_cce"    "wbgi_pse"
[6] "former_col"  "lp_lat_abst"
```

We want to change the first element of that vector. We know that we can use square brackets to subset vectors. Let's display the first element of the vector of variable names only.

```
names(world.data)[1]
```

```
[1] "h_j"
```

Now we simply change the name using the assignment arrow `<-` and our new variable names goes in quotes.

```
names(world.data)[1] <- "free.judiciary"
```

We now check the variable names to confirm that we successfully changed the name.

```
names(world.data)
```

```
[1] "free.judiciary" "wdi_gdpc"       "undp_hdi"       "wbgi_cce"
[5] "wbgi_pse"       "former_col"     "lp_lat_abst"
```

3.1.5 Distributions

A marginal distribution is the distribution of a variable by itself. Let's look at the summary statistics of the United Nations Development Index *undp_hdi* using the `summary()` function.

```
summary(world.data$undp_hdi)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
 0.2730  0.5272  0.7455  0.6946  0.8350  0.9560       9
```

How nice. This returns summary stats. We see the range(minimum to maximum). We see the interquartile range (1st quartile to 3rd quartile). We also see mean and median. Finally, we see the number of NAs.

Oh we forgot. We said, when we drop missing on variable, we may lose information when we work on a new variable. Let's restore our dataset *world.data* to its original state.

```
world.data <- full.dataset
```

Now, we check the summary stats again.

```
summary(world.data$undp_hdi)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
 0.2730  0.5390  0.7510  0.6982  0.8335  0.9560      19
```

In the smaller dataset (where we had dropped missings from *h_j*), we had 9 missings. Now, we have 19 missings. The difference is 10. Our smaller dataset had 25 rows less than the bigger dataset. Therefore, we would have thrown away 6 good observations. That is not nothing. It's 3 percent of our data.

Let's drop missing on *undp_hdi* and rename it to *hdi*.

```
world.data <- world.data[ which( !is.na(world.data$undp_hdi) ) , ]
```

Let's change the name.

```
names(world.data)[3] <- "hdi"
names(world.data)
```

```
[1] "h_j"         "wdi_gdpc"    "hdi"         "wbgi_cce"    "wbgi_pse"
[6] "former_col"  "lp_lat_abst"
```

Let's take the mean of *hdi*.

```
hdi.mean <- mean( world.data$hdi )
hdi.mean
```

```
[1] 0.69824
```

The mean of *hdi* is the mean in the sample. We would like the mean of hdi in the population. Remember that sampling variability causes us to estimate a different mean every time we take a new sample.

We learned that the means follow a distribution if we take the mean repeatedly in different samples. In expectation the population mean is the sample mean. How certain are we about the mean. Well, we need to know how the sampling distribution looks like.

To find out we estimate the standard error of the mean. The standard error is the standard deviation of the sampling distribution. The name is not standard deviation but standard error to indicate that we are talking about the distribution of a statistic (the mean) and not a random variable.

The formula for the standard error of the mean is:

$$s_{\bar{x}} = \frac{\sigma}{\sqrt(n)}$$

The σ is the real population standard deviation of the random variable *hdi* which is unknown to us. We replace the population standard deviation with our sample estimate of it.

$$s_{\bar{x}} = \frac{s}{\sqrt(n)}$$

The standard error of the mean estimate is then

```
se.hdi <- sd(world.data$hdi) / sqrt( nrow(world.data) )
se.hdi
```

```
[1] 0.01362411
```

Okay, so the mean is 0.69824 and the standard error of the mean is 0.0136241.

We know that the sampling distribution is approximately normal. That means that 95 percent of all observations are within 1.96 standard deviations (standard errors) of the mean.

$$\bar{x} \pm 1.96 \times s_{\bar{x}}$$

So what is that in our case?

```
lower.bound <- hdi.mean - 1.96 * se.hdi
lower.bound
```

```
[1] 0.6715367
```

```
upper.bound <- hdi.mean + 1.96 * se.hdi
upper.bound
```

```
[1] 0.7249432
```

That now means the following. Were we to take samples of *hdi* again and again and again, then 95 percent of the time, the mean would be in the range from 0.6715367 to 0.7249432.

What is a probability? "The long-run relative frequency," you all scream in unison. Given that definition, you can say: "With 95 percent probability, the mean is in the range 0.6715367 to 0.7249432."

Sometimes people like to former way of phrasing this relationship better than the latter. In this case you tell them: "a probability is the long-run relative frequency of an outcome."

Now, let's visualise our sampling distribution. We haven't actually taken many samples, so how could we visualise the sampling distribution? Well, we know the sampling distribution looks normal. We know that the mean is our mean estimate in the sample. And finally, we know that the standard deviation is the standard error of the mean.

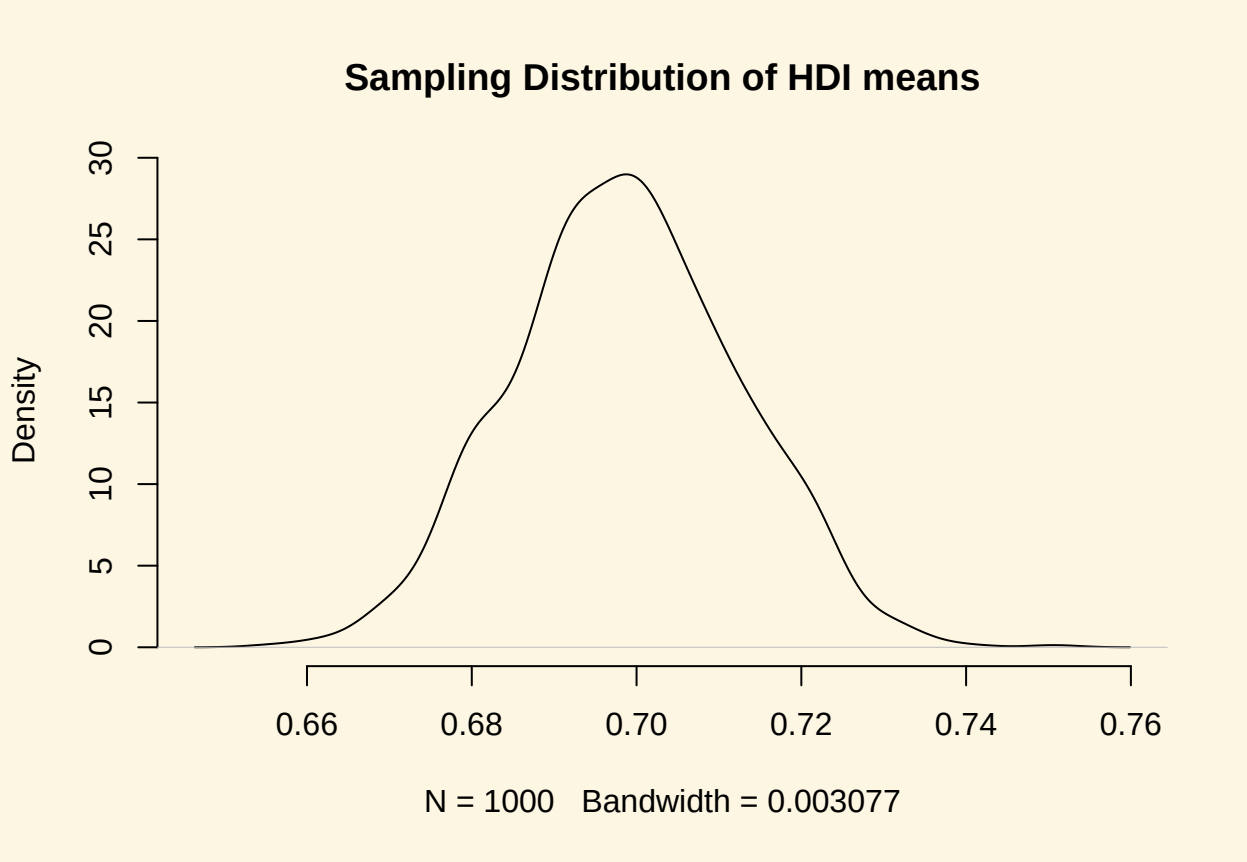
We can randomly draw values from a normal distribution with mean 0.69824 and standard deviation 0.0136241. We do this with the `rnorm()` function. It's first argument is the number of values to draw at random from the normal distribution. The second argument is the mean and the third is the standard deviation.

Recall, that a normal distribution has two parameters that characterise it completely: the mean and the standard deviation. So with those two we can draw the distribution.

```
draw.of.hdi.means <- rnorm( 1000, mean = hdi.mean, sd = se.hdi )
```

We have just drawn 1000 mean values at random from the distribution that looks like our sampling distribution.

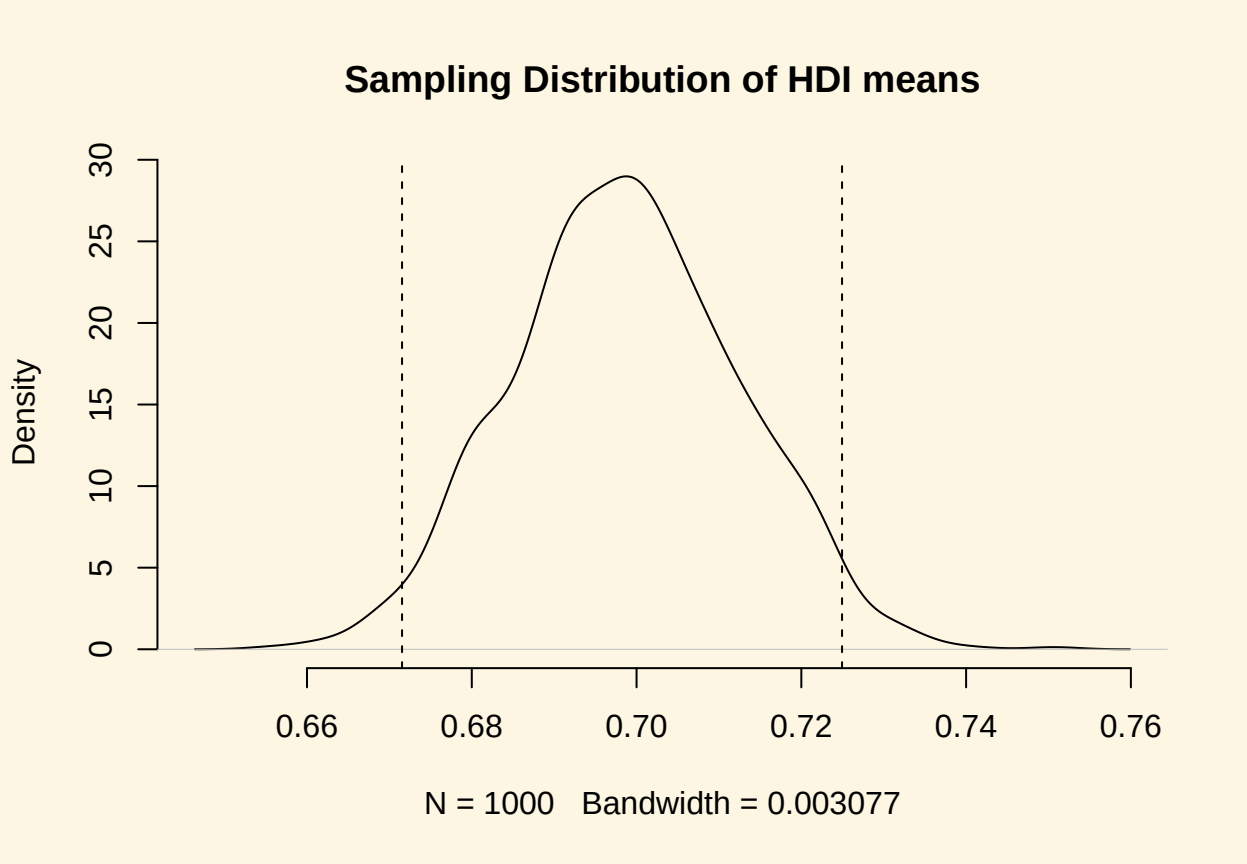
```
plot(
 density( draw.of.hdi.means ),
 bty = "n",
 main = "Sampling Distribution of HDI means"
)
```



Beautiful Let's add the 95 percent confidence interval around our mean estimate. The confidence interval quantifies our uncertainty. We said 95 percent of the time the mean would be in the interval from 0.6715367 to 0.7249432."

```
abline( v = lower.bound, lty = "dashed")
abline( v = upper.bound,  lty = "dashed")
```

You do not need to run the plot function again. You can just add to the plot. Check the help function of `abline()` to see what its arguments refer to.



Fantastic! You can see that values below and above our confidence interval are quite unlikely. Those values in the tails would not occur often.

Not often, but not impossible.

Let's say that we wish know the probability that we take a sample and our estimate of the mean is greater or equal 0.74. We would need to integrate over the distribution from $-\inf$ to 0.74. Fortunately R has a function that does that for us. We need the `pnorm()`. It calculates the probability of a value that is smaller or equal to the value we specify. In other words, it gives us the probability from the cumulative normal.

As the first argument `pnrom()` wants the value; 0.74 in our case. The second and third arguments are the mean and the standard deviation that characterise the normal distribution.

```
pnorm(0.74, mean = hdi.mean, sd = se.hdi)
```

```
[1] 0.9989122
```

What!? The probability to draw a mean 0.74 is 99.9 percent!? That cannot be the value is so far in the tail of the distribution.

Well, this is the cumulative probability of drawing a value that is equal to or smaller than 0.74. All probabilities sum to 1. So if we want to know the probability of drawing a value that is greater than 0.74, we subtract the probability, we just calculated, from 1.

```
1 - pnorm(0.74, mean = hdi.mean, sd = se.hdi)
```

```
[1] 0.001087785
```

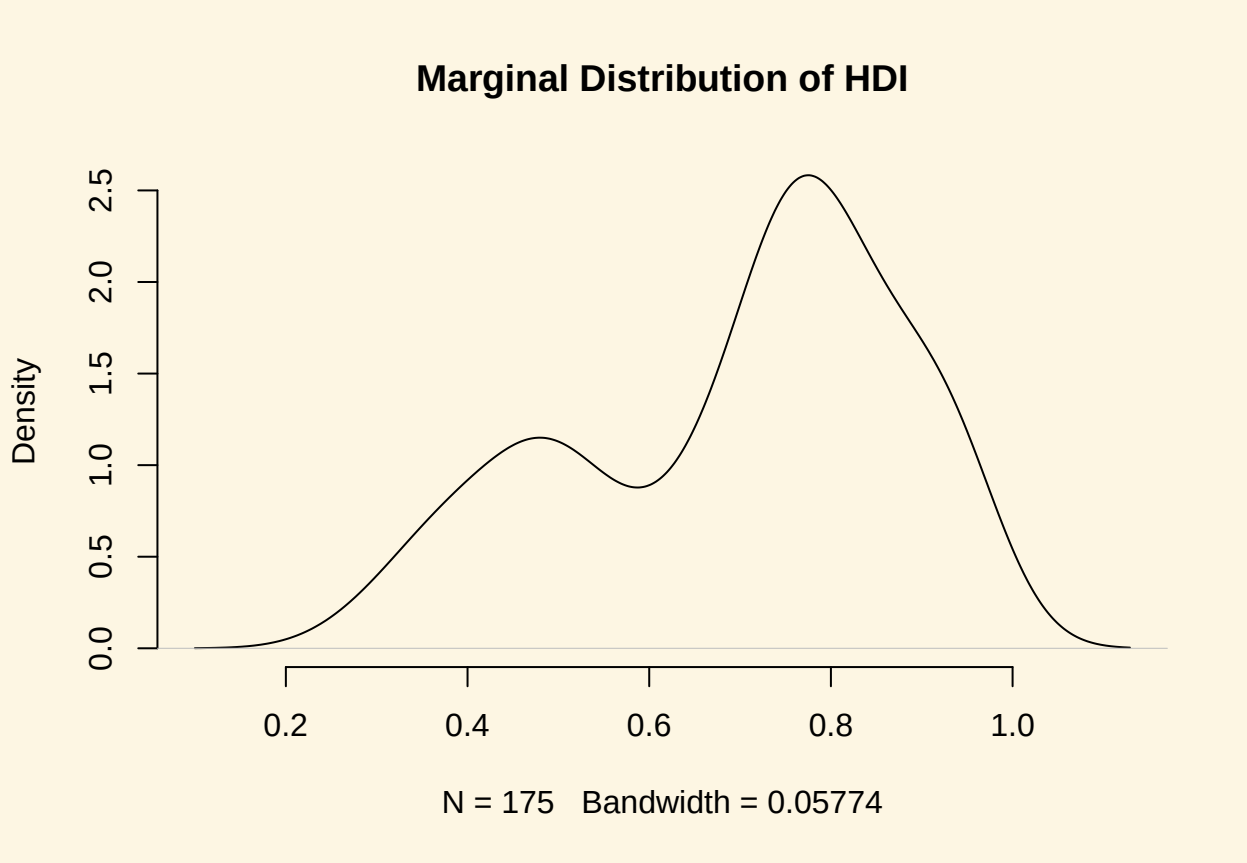
Right, so the probability of getting a mean of *hdi* in a sample is 0.1 percent.

3.1.6 Conditional Distributions

Let's look at *hdi* by *former_col*. The variable *former_col* is 1 if a country is a former colony and 0 otherwise. The variable *hdi* is continuous.

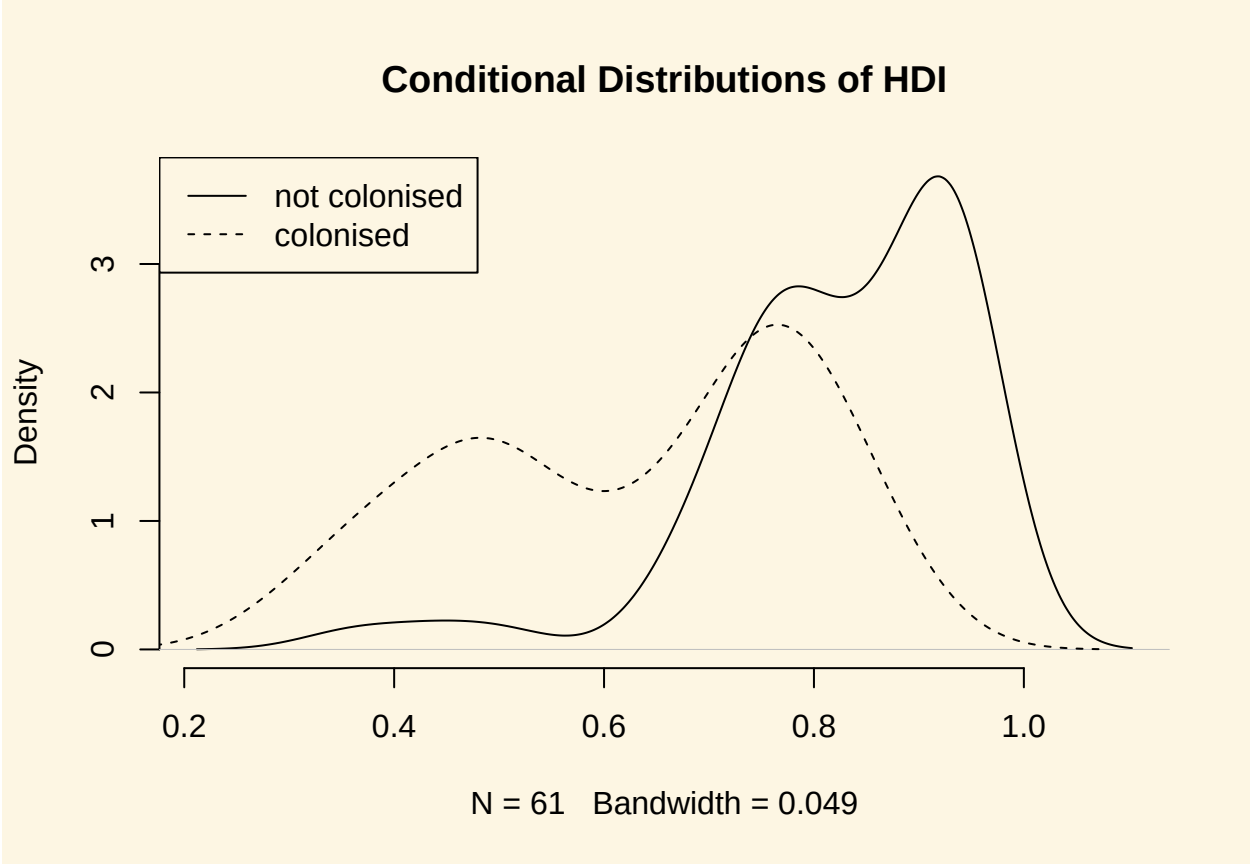
Before we start, we plot the marginal pdf of *hdi*.

```
plot(
  density(world.data$hdi),
  bty = "n",
  main = "Marginal Distribution of HDI"
)
```



The distribution is bimodal. There is one peak at the higher development end and one peak at the lower development end. Could it be that these two peaks are conditional on whether a country was colonised or not? Let's plot the conditional distributions.

```
plot(
  density(world.data$hdi[world.data$former_col == 0]),
  bty = "n",
  main = "Conditional Distributions of HDI"
)
lines(density(world.data$hdi[world.data$former_col == 1]), lty = "dashed")
legend("topleft", c("not colonised", "colonised"), lty = c("solid", "dashed"))
```



It's not quite like we expected. The distribution of human development of not colonised countries is shifted to right of the distribution of colonised countries and it is clearly narrower. Interestingly though, the distribution of former colonies has a greater variance. Evidently, some former colonies are doing very well and some are doing very poorly. It seems like knowing whether a country was colonised or not tells us something about its likely development but not enough. We cannot, e.g., say colonisation is the reason why countries do poorly. Probably, there are differences among types of colonial institutions that were set up by the colonisers.

Let's move on and examine the probability that a country has .8 or more on *hdi* given that it is a former colony.

We can get the cumulative probability with the `ecdf()` function. It returns the empirical cumulative distribution, i.e., the cumulative distribution of our data. We know that we can subset using square brackets. That's all we need.

```
cumulative.p <- ecdf(world.data$hdi[ world.data$former_col == 1 ])
1 - cumulative.p(.8)
```

```
[1] 0.1666667
```

Okay, the probability that a former colony has .8 or larger on the *hdi* is 16.6 percent. Go ahead figure out the probability for not former colonies on your own.

3.1.7 Exercises

1. Create a script and call it assignment03. Save your script.
2. Load the *world.data* dataset from your disk.
3. Rename the variable *wdi_gdpc* into *gdpc*.
4. Delete missing values from *gdpc*.

5. Inspect *former_col* and delete missing values from it.
6. Turn *former_col* into a factor variable with the appropriate lables.
7. Compute the probability that a county is richer than 55 000 per capita.
8. Compute the same probability given that a country is a former colony.
9. Compute the conditional expectation of wealth (gdp per capita) for a former colony.
10. Compute the conditional expectation of wealth for country that is not a former colony.
11. What is the probability that a former colony is 2 standard deviations below the mean wealth level?
12. What is the corresponding probability for a country that has not been colonised?
13. Compute the probability that a former colony is the wealth interval from 25 000 to 31 000.
14. Copmute the probability that a **not** former colony is in the top 2.5 percent of the wealth distribution.
15. At which wealth level is a country in the bottom 2.5 percent of the wealth distribution?

3.2 Solutions

3.2.0.1 Exercise 2

Load the world.data dataset from your disk.

```
world.data <- read.csv("QoG2012.csv")
```

3.2.0.2 Exercise 3

Rename the variable wdi_gdpc into gdpc.

```
# to see all variable names
names(world.data)
```

```
[1] "h_j"         "wdi_gdpc"    "undp_hdi"    "wbgi_cce"    "wbgi_pse"
[6] "former_col"  "lp_lat_abst"
```

```
# wdi_gdpc is the second variable. We rename the second element of the names vector
names(world.data)[2] <- "gdpc"
```

3.2.0.3 Exercise 4

Delete missing values from gdpc.

```
# to check whether there are any missings or not
summary(world.data$gdpc)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  226.2  1768.0  5326.1 10184.1 12976.5 63686.7      16
```

```
# we have missings, let's make a copy of world.data before deleting
full.world.data <- world.data
# now let's delete the 16 rows with missings on gdpc
world.data <- world.data[ which(!is.na(world.data$gdpc)) , ]
```

3.2.0.4 Exercise 5

Inspect former_col and delete missing values from it.

```
# we instpect the variable and check for missings
summary(world.data$former_col)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.0000  1.0000  0.6348  1.0000  1.0000
```

```
# there are none, so there is nothing to delete
```

3.2.0.5 Exercise 6

Turn former_col into a factor variable with the appropriate labels.

```
# we check the current storage type
str(world.data$former_col)
```

```
 int [1:178] 0 0 1 1 1 0 1 0 0 1 ...
```

```
# it's numeric, so we change it to nominal
world.data$former_col <- factor(world.data$former_col,
                                levels = c(0,1),
                                labels = c("not colonised", "former colony" ))
# let's check the results
table(world.data$former_col)
```

```
not colonised former colony
           65           113
```

Wait a minute. Is there something wrong here? The mean of the variable is 0.63. That means 63 percent of all countries are former colonies. Let's check whether we got that.

```
table(world.data$former_col) / sum(table(world.data$former_col))
```

```
not colonised former colony
    0.3651685     0.6348315
```

Ah, that's correct. It's good to make sure, we did not mess up the re-coding.

3.2.0.6 Exercise 8.

Compute the probability that a county is richer than 55 000 per capita.

Wealth is never normally distributed. We don't even need to check. We use the `ecdf()` function for the empirical cumulative distribution.

The probability that a country is richer than 55 000 is 1 minus the cumulative probability of 55000.

```
# get the empirical cumulative distribution of wealth
c.dist.of.wealth <- ecdf(world.data$gdpc)
# the prob
1 - c.dist.of.wealth(55000)
```

```
[1] 0.01123596
```

The probability is 0.01. Put differently 1 percent of countries is richer than 55 000 US dollars per capita.

3.2.0.7 Exercise 8

Compute the same probability given that a country is a former colony.

The approach is similar but we want the conditional cumulative distribution, where the condition is that a country is a former colony.

```
# conditional cumulative distribution
c.dist.of.wealth2 <- ecdf(world.data$gdpc[world.data$former_col == "former colony"])
1 - c.dist.of.wealth2(55000)
```

```
[1] 0.008849558
```

The probability is 0.009. The probability that a former colony is that rich is slightly lower than that any country is.

3.2.0.8 Exercise 9

Compute the conditional expectation of wealth (gdp per capita) for a former colony.

The conditional expectation is the mean of wealth among all former colonies. We know how to do this from last week. We take the mean of wealth and subset using square brackets.

```
mean( world.data$gdpc[world.data$former_col == "former colony"] )
```

```
[1] 6599.714
```

The conditional expectation of wealth for former colonies is 6600 US dollars per capita.

3.2.0.9 Exercise 10

Compute the conditional expectation of wealth for a country that is not a former colony.

```
mean( world.data$gdpc[world.data$former_col == "not colonised"] )
```

```
[1] 16415.39
```

The corresponding expectation for countries that have not been a colony is 16415 US dollars.

3.2.0.10 Exercise 11

What is the probability that a former colony is 2 standard deviations below the mean wealth level?

We first find out what a standard deviation of wealth is in the conditional distribution of wealth for former colonies.

```
# standard deviation of wealth for former colonies
sd.wealth.cols <- sd(world.data$gdpc[world.data$former_col == "former colony"])
sd.wealth.cols
```

```
[1] 9783.914
```

Interesting, the standard deviation is greater than the mean. Apparently, former colonies are very different. Some do poorly and some extremely well.

Doesn't this pose a problem for the task though? How can a country's wealth be 2 standard deviations below the mean if the mean is smaller than the standard deviation?

Well, that's not possible. Negative wealth does not exist. Consequently, that probability is 0.

3.2.0.11 Exercise 12

We get the standard deviation of the wealth distribution of countries that have not been colonised.

```
# standard deviation of wealth for not former colonies
sd.wealth.not.cols <- sd(world.data$gdpc[world.data$former_col == "not colonised"])
```

The result is similar to exercise 11. The mean is 16 415 and the standard deviation is 13 766. 2 standard deviations below the mean would be negative. The probability of that is 0.

3.2.0.12 Exercise 13

Compute the probability that a former colony is the wealth interval from 25 000 to 31 000.

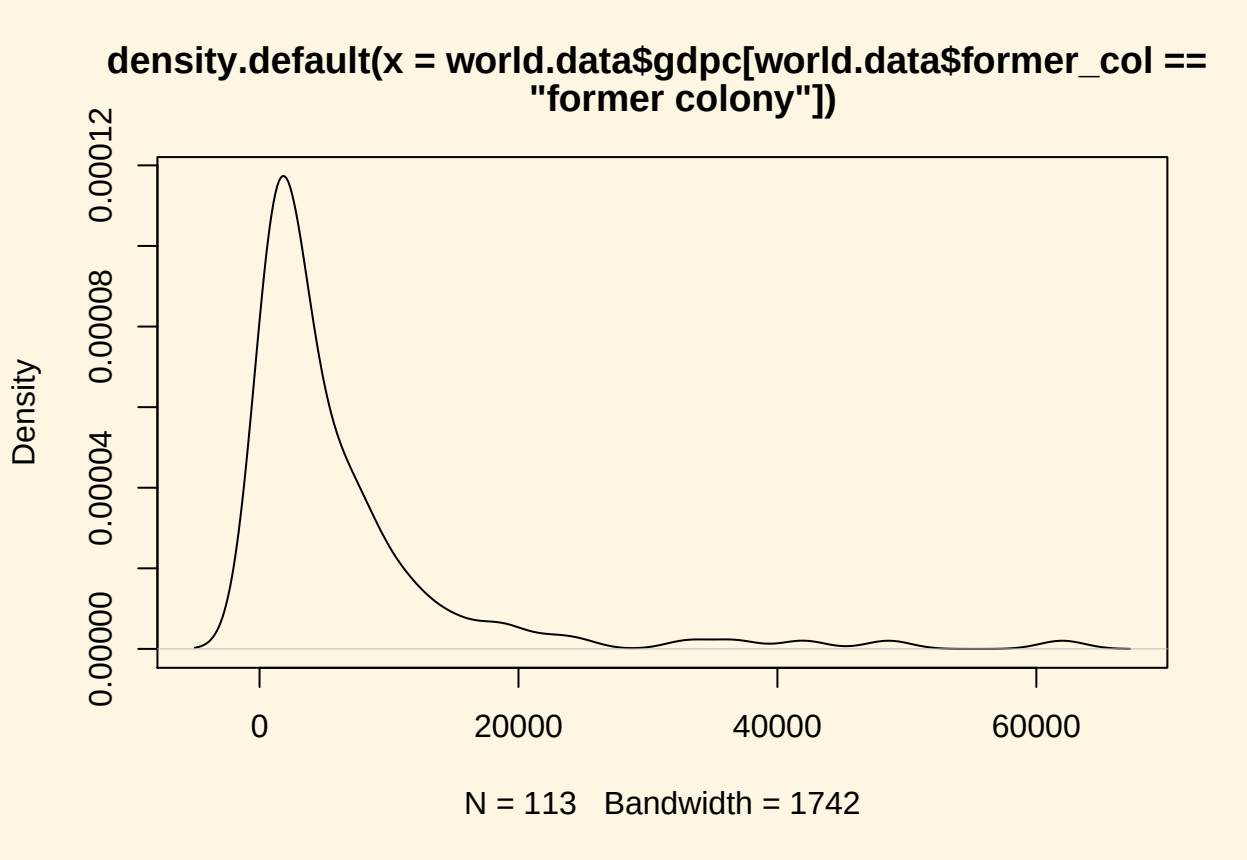
We compute the cumulative probabilities that a country has 31 000 and 25 000 and then take the difference.

```
# cumulative probability of 31 000
p1 <- c.dist.of.wealth2(31000)
# cumulative probability of 25 000
p2 <- c.dist.of.wealth2(25000)
# probability of country in the interval
p1 - p2
```

```
[1] 0
```

The answer is again: 0. Let's have a look at the distribution to see what's going on there.

```
plot(density( world.data$gdpc[world.data$former_col == "former colony"] ))
```



It seems like there are no countries in that interval. Let's check:

```
# let's look at summary stats first
summary(world.data$gdpc[ world.data$former_col == "former colony"  ])
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  226.2  1263.7  3157.5  6599.7  7938.3 62005.6
```

```
# so there is at least 1 country that is richer. Let's look at all
# countries that are former colonies and also richer than 25 000
world.data[which( world.data$gdpc > 25000 & world.data$former_col == "former colony"), ]
```

```
    h_j     gdpc undp_hdi  wbgi_cce    wbgi_pse    former_col lp_lat_abst
24    0 48585.73    0.867 0.3304393  0.99980551 former colony   0.0477778
92    1 33079.87    0.838 1.0835209 -0.02684768 former colony   0.3255555
142   0 62005.56    0.833 0.9734111  0.76874268 former colony   0.2811111
156   1 36732.23    0.902 2.3715818  1.34370053 former colony   0.0135556
175   0 42004.04    0.824 1.0449655  0.80680293 former colony   0.2666667
```

That's it. There are exactly zero countries in that interval.

Sub-setting with 2 conditions in square brackets may be new to you. We did that with `&` operator which means "and".

3.2.0.13 Exercise 14

Compute the probability that a not former colony is in the top 2.5 percent of the wealth distribution.

We find the value first. The top 2.5 percent are in the 97.5 percentile of the distribution. We use the `quantile()` function to get the value of the 97.5th percentile.

```
rich.countries <- quantile(world.data$gdpc[ world.data$former_col == "not colonised"], .975)
rich.countries
```

```
   97.5%
41447.31
```

The value that puts a country in the top 2.5 percent of the conditional wealth distribution (where the condition is that a country was not colonised) is 41447 US dollars.

We now take the empirical cumulative distribution and get the probability of being richer.

```
# conditional cumulative distribution
c.dist.of.wealth3 <- ecdf(world.data$gdpc[world.data$former_col == "not colonised"])
# conditional probability of being in the top 2.5 percent of not colonised countries
1 - c.dist.of.wealth3(rich.countries)
```

```
[1] 0.03076923
```

The probability is 0.03.

3.2.0.14 Exercise 15

At which wealth level is a country in the bottom 2.5 percent of the wealth distribution?

The question asks for the wealth level that puts a country in the bottom 2.5 percent independent of whether it was a colony or not. The `quantile()` function returns that value.

```
quantile( world.data$gdpc, .025 )
```

```
    2.5%
526.8697
```

At 527 US dollars per capita, a country is in the bottom 2.5 percent of the wealth distribution.

4 T-test for Difference in Means and Hypothesis Testing

4.1 Seminar

Let's remove all objects from our workspace and set the working directory.

```
rm(list=ls())
setwd("~/statistics1")
```

We load the data from the Quality of Government Institute again. Let's have a look at the codebook:

Variable	Description
h_j	1 if Free Judiciary
wdi_gdpc	Per capita wealth in US dollars
undp_hdi	Human development index (higher values = higher quality of life)
wbgi_cce	Control of corruption index (higher values = more control of corruption)
wbgi_pse	Political stability index (higher values = more stable)
former_col	1 = country was a colony once
lp_lat_abst	Latitude of country's captial divided by 90

Go ahead and load the data from last week yourself.

```
world.data <- read.csv("QoG2012.csv")
```

We can get summary statistics of each variable in the dataset by using the `summary()` function over the dataset.

```
summary(world.data)
```

```
      h_j            wdi_gdpc          undp_hdi         wbgi_cce
 Min.   :0.0000   Min.   :  226.2   Min.   :0.2730   Min.   :-1.69953
 1st Qu.:0.0000   1st Qu.: 1768.0   1st Qu.:0.5390   1st Qu.:-0.81965
 Median :0.0000   Median : 5326.1   Median :0.7510   Median :-0.30476
 Mean   :0.3787   Mean   :10184.1   Mean   :0.6982   Mean   :-0.05072
 3rd Qu.:1.0000   3rd Qu.:12976.5   3rd Qu.:0.8335   3rd Qu.: 0.50649
 Max.   :1.0000   Max.   :63686.7   Max.   :0.9560   Max.   : 2.44565
 NA's   :25       NA's   :16        NA's   :19       NA's   :2
    wbgi_pse          former_col      lp_lat_abst
 Min.   :-2.46746   Min.   :0.0000   Min.   :0.0000
 1st Qu.:-0.72900   1st Qu.:0.0000   1st Qu.:0.1343
 Median : 0.02772   Median :1.0000   Median :0.2444
 Mean   :-0.03957   Mean   :0.6289   Mean   :0.2829
 3rd Qu.: 0.79847   3rd Qu.:1.0000   3rd Qu.:0.4444
 Max.   : 1.67561   Max.   :1.0000   Max.   :0.7222
                                     NA's   :7
```

4.1.1 The Standard Error

The standard error of an estimate quantifies uncertainty that is due to sampling variability. Recall that we infer from a sample to the population. Let's have a look at *wdi_gdpc* which is gdp per capita. We re-name the variable to *wealth*. Do so on your own.

```
names(world.data)[2] <- "wealth"
names(world.data)
```

```
[1] "h_j"         "wealth"      "undp_hdi"    "wbgi_cce"    "wbgi_pse"
[6] "former_col"  "lp_lat_abst"
```

Now, have a look at the mean of the new *wealth* variable.

```
mean(world.data$wealth)
```

```
[1] NA
```

R returns NA because there are missing values on the *wealth* variable and we cannot calculate with NAs. For instance, `2 + NA` will return NA. We make a copy of the full data set and then delete missing values. We did this last week. Go ahead do so on your own.

```
# copy of the dataset
full.data <- world.data

# delete rows from dataset that have missings on wealth variable
world.data <- world.data[ !is.na(world.data$wealth) , ]
```

Now, compute the mean of *wealth* again.

```
mean(world.data$wealth)
```

```
[1] 10184.09
```

The mean estimate in our sample is 10184.09. We are generally interested in the population. Therefore, we infer from our sample to the population. Our main problem is that samples are subject to sampling variability. If we take another sample, our mean estimate would be different. The standard error quantifies this type of uncertainty.

The formula for the standard error of the mean is:

$$SE(\bar{Y}) = \frac{s_Y}{\sqrt{n}}$$

Where s_Y is the standard deviation (of *wealth*) and n is the number of observations (of *wealth*).

We compute the standard error in R:

```
se.y_bar <- (sd(world.data$wealth) / sqrt( length(world.data$wealth) ))
se.y_bar
```

```
[1] 922.7349
```

The standard error is ~922.73. The mean of the sampling distribution is the population mean (or close to it — the more samples we take, the closer is the mean of the sampling distribution to the population mean). The standard error is the average difference from the population mean. We have taken 1 sample. When taking any random sample, the average difference between the mean in that sample and the population mean is the standard error.

We need the standard error for hypothesis testing. You will see how in the following.

4.1.2 T-test (one sample hypothesis test)

A knowledgeable friend declares that worldwide wealth stands at exactly 10 000 US dollars per capita today. We would like to know whether she is right and tease her relentlessly if she isn't. To that end, we assume that her claim is the population mean. We then estimate the mean of *wealth* in our sample. If the difference is large enough, so that it is unlikely that it could have occurred by chance alone, we can reject her claim.

So, first we take the mean of the *wealth* variable.

```
mean(world.data$wealth)
```

```
[1] 10184.09
```

Wow, our friend is quite close. Substantially, the difference of our friends claim to our estimate is small but we could still find that the difference is statistically significant (it's a noticeable systematic difference).

Because we do not have information on all countries, our 10184.09 is an estimate and the true population mean – the population here would be all countries in the world – may be 10000 as our friend claims. We test this statistically.

In statistics jargon: we would like to test whether our estimate is statistically different from the 10000 figure (the null hypothesis) suggested by our friend. Put differently, we would like to know the probability that we estimate 10184.09 if the true mean of all countries is 10000.

Recall, that the standard error of the mean (which is the estimate of the true standard deviation of the population mean) is estimated as:

$$\frac{s_Y}{\sqrt{n}}$$

Before we estimate the standard error, let's get n (the number of observations). We have done this above but to make our code more readable, we save the number of observations in an object that we call `n`. Go ahead and do this on your own.

```
n <- length(world.data$wealth)
n
```

```
[1] 178
```

With the function `length(world.data$world)` we get all observations in the data. Now, let's take the standard error of the mean again.

```
se.y_bar <- sd(world.data$wealth) / sqrt(n)
```

We know that 1 standard error is one average deviation from the population mean. The sampling distribution is approximately normal. 95 percent of the observations under the normal distribution are within 2 standard deviations of the mean.

We construct the confidence interval within which the population mean lies with 95 percent probability in the following way. First, we take our mean estimate of *wealth*. That's the sample mean and not the population mean. Second, we go 2 standard errors to the left of it. This is the lower bound of our confidence interval. Third, we go 2 standard deviations to the right of the sample mean. That is the upper bound of our confidence interval.

The 95 percent confidence interval around the sample means gives the interval within which the population mean lies with 95 percent probability.

We want to know what the population mean is, right? Yes, that's right. Therefore, we want the confidence interval to be as narrow as possible. The narrower the confidence interval, the more precise we are about the population mean. For instance, saying the population mean of income is between 9 950 and 10 050 is more precise than saying the population mean is between 5 000 and 15 000.

We construct the confidence interval with the standard error. That means, the smaller the standard error, the more precise our estimate. The formula for the 95 percent confidence interval is:

$$\bar{Y} \pm 1.96 \times SE(\bar{Y})$$

"Where does the 1.96 come from", you ask. It's a critical value. More on that later. For now, just recall that in a normal distribution 95 percent of all observations are within 1.96 standard errors of the mean.

We now construct our confidence interval. Our sample is large enough to assume that the sampling distribution is approximately normal. So, we can go 1.96 standard deviations to the left and to the right of the mean to construct our 95% confidence interval.

```
# lower bound
lb <- mean(world.data$wealth) - 1.96 * se.y_bar
# upper bound
ub <- mean(world.data$wealth) + 1.96 * se.y_bar
# results (the population mean lies within this interval with 95% probability)
lb # lower bound
```

```
[1] 8375.531
```

```
mean(world.data$wealth) # sample mean
```

```
[1] 10184.09
```

```
ub # upper bound
```

```
[1] 11992.65
```

You can make this look a little more like a table like so:

```
ci <- cbind(lower_bound = lb, mean = mean(world.data$wealth), upper_bound = ub)
ci
```

```
     lower_bound     mean upper_bound
[1,]    8375.531 10184.09    11992.65
```

The `cbind()` function stands for column-bind and creates a 1×3 matrix.

So we are 95% confident that the population average level of wealth is between 8375.53 US dollars and 11992.65 US dollars. You can see that we are not very certain about our estimate and we most definitely cannot rule out that our friend is right (she claimed that the population mean is 10 0000—that is within our interval). Hence, we cannot reject it.

A different way of describing our finding is to emphasize the logic of (hypothetical) repeated sampling. In a process of repeated sampling we can expect that the confidence interval that we calculate for each sample will include the true population value 95% of the time. That is equivalent to what we said earlier because a probability is the long-run relative frequency of an outcome.

4.1.2.1 The t value

We now estimate the t value. Recall that our friend claimed that the population mean was 10 000. This is the null hypothesis that we wish to falsify. We estimated something else in our data, namely 10184.0910395. The t value is the difference between our estimate (the result we get by looking at data) and the population mean under the null hypothesis divided by the standard error of the mean.

$$\frac{\bar{Y} - \mu_0}{SE(\bar{Y})}$$

Where $\bar{Y}$ is the mean in our data, μ_0 is the population mean under the null hypothesis and $SE(\bar{Y})$ is the standard error of the mean.

Okay, let's compute this in R:

```
t.value <- (mean(world.data$wealth) - 10000) / se.y_bar
t.value
```

```
[1] 0.1995059
```

Look at the formula until you understand what is going on. In the numerator we take the difference between our estimate and the population mean under the null hypothesis. In expectation that difference should be

0—assuming that the null hypothesis is true. The larger that difference, the less likely that the null hypothesis is true.

We divide by the standard error to transform the units of the difference into standard deviations. Before we transformed the units, our difference was in the units of whatever variable we are looking at (US dollars in our example). By dividing by the standard error, we have normed the variable. Its units are now standard deviations from the mean.

Assume that the null hypothesis is true. In expectation the difference between our estimate in the data and the population mean should be **0 standard deviations**. The more standard deviations our estimate is away from the population mean under the null hypothesis, the less likely it is that the null hypothesis is true.

Within **1.96 standard deviations** from the mean lie 95 percent of all observations. That means, it is very unlikely that the null hypothesis is true, if the difference that we estimated is further than 1.96 standard deviations from the mean. "How unlikely," you ask. We would need the p value, for the exact probability. However, the probability is less than 5 percent, if the estimated difference is more than 1.96 standard deviations from the population mean under the null hypothesis.

Back to our t value. We estimated a t value of 0.1995059. That means that a sample estimate of 10184.0910395 is 0.1995059 standard deviations from the population mean under the null hypothesis—which is 10 000 in our sample.

Our t value suggests that our sample estimate would only be 0.1995059 standard deviations away from the population mean under the null. That is not unlikely at all. We can only reject the null hypothesis if we are more than 1.96 standard deviations away from the mean.

4.1.2.2 The p value

Let's estimate the precise p-value by calculating how likely it would be to observe a t-statistic of 0.1995059 from a t-distribution with n - 1 (177) degrees of freedom.

The function `pt(t.value, df = n-1)` is the cumulative probability that we get the t.value we put into the formula if the null is true. The cumulative probability is estimated as the interval from minus infinity to our t.value. So, 1 minus that probability is the probability that we see anything larger (in the right tale of the distribution). But we are testing whether the true mean is different from 10000 (including smaller). Therefore, we want the probability that we see a t.value in the right tale *or* in the left tale of the distribution. The distribution is symmetric. So we can just calculate the probability of seeing a t-value in the right tale and multiply it by 2.

```
2* ( 1 - pt(t.value, df = (n-1) ))
```

```
[1] 0.8420961
```

The p-value is way too large to reject the null hypothesis (the true population mean is 10 000). If we specified an alpha-level of 0.05 in advance, we would reject it only if the p-value was smaller than 0.05. If we specified an alpha-level of 0.01 in advance, we would reject it only if the p-value was smaller than 0.01, and so on.

Let's verify our result using the the t-test function `t.test()`. The syntax of the function is:

```
t.test(formula, mu, alt, conf)
```

Lets have a look at the arguments.

Arguments	Description
`formula`	Here, we input the vector that we calculate the mean of. For the one-sample t test, in our example, this is the mean of *wealth*. For the t test for the difference in means, we would input both vectors and separate them by a comma.
`mu`	Here, we set the null hypothesis. The null hypothesis is that the true population mean is 10000. Thus, we set `mu = 10000`.

Arguments	Description
`alt`	There are two alternatives to the null hypothesis that the difference in means is zero. The difference could either be smaller or it could be larger than zero. To test against both alternatives, we set `alt = "two.sided"`.
`conf`	Here, we set the level of confidence that we want in rejecting the null hypothesis. Common confidence intervals are: 95%, 99%, and 99.9%—they correspond to alpha levels of 0.05, 0.01 and 0.001 respectively.

```
t.test(world.data$wealth, mu = 10000, alt = "two.sided")
```

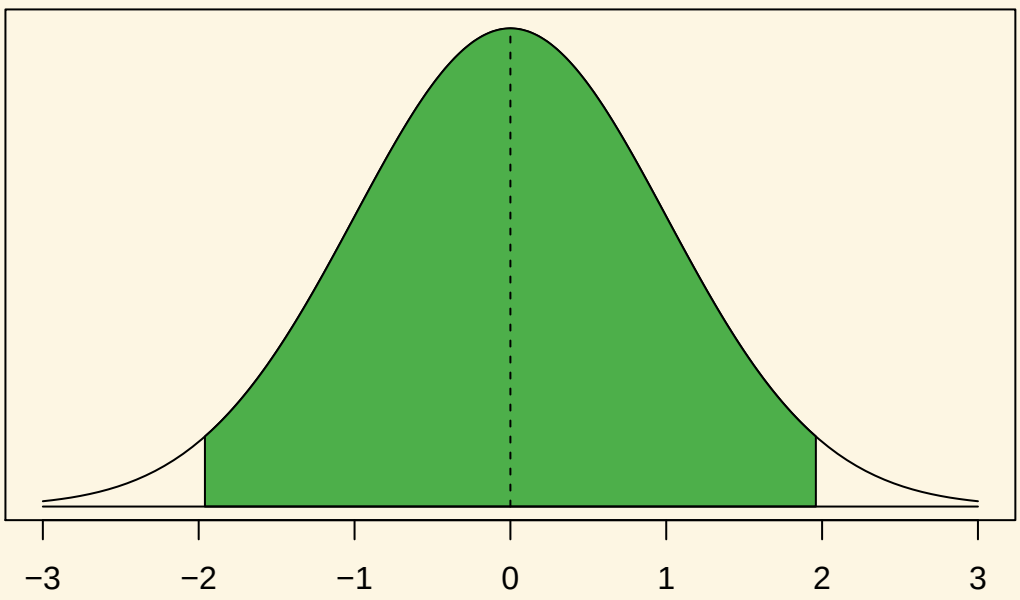
```

    One Sample t-test

data:  world.data$wealth
t = 0.19951, df = 177, p-value = 0.8421
alternative hypothesis: true mean is not equal to 10000
95 percent confidence interval:
  8363.113 12005.069
sample estimates:
mean of x
 10184.09
```

The results are similar. Therefore we can conclude that we are unable to reject the null hypothesis suggested by our friend that the population mean is equal to 10000. Let's see how we determine critical values.

4.1.2.3 Critical Values

In social sciences, we usually operate with an alpha level of 0.05. That means, we reject the null hypothesis if the p value is smaller than 0.05. Or put differently, we reject the null hypothesis if the 95 percent confidence interval does not include the population mean under the null hypothesis—which is always the case if our estimate is further than two standard errors from the mean under null hypothesis (usually 0).

We said earlier that the critical value is 1.96 for an alpha level of 0.05. That is true in large samples where the distribution of the t value follows a normal distribution. 95 percent of all observations are within 1.96 standard deviations of the mean.



The green area under the curve covers 95 percent of all observations. There are 2.5 percent in each tail. We reject the null hypothesis if our estimate is in the tails of the distribution. It must be further than 1.96 standard deviations from the mean. But how did we know that 95 percent of the area under the curve is within 1.96 standard deviations from the mean?

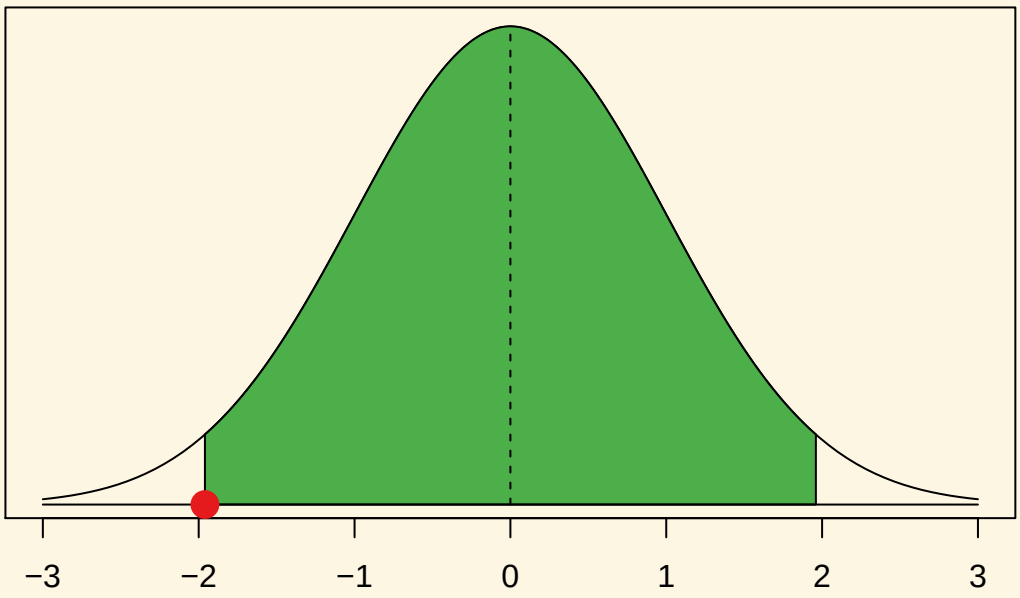
Let's separate the curve in you mind into 3 pieces. The left tail covers 2.5 percent of the area under the curve. The green middle bit covers 95 percent and the right tail again 2.5 percent. Now we do this as cumulative probabilities. The left tail ends at 2.5 percent cumulative probability. The green area ends at 97.5 percent cumulative probability and the right tail ends at 100 percent.

The critical value is were the left tail ends or the right tail starts (looking at the curve from left to right). Let's get the value where the cumulative probability is 2.5 percent—where the left tail ends.

```
#
qnorm(0.025, mean = 0, sd = 1)
```

```
[1] -1.959964
```

If you look at the x-axis of our curve that is indeed where the left tail ends. We add a red dot to our graph to highlight it.

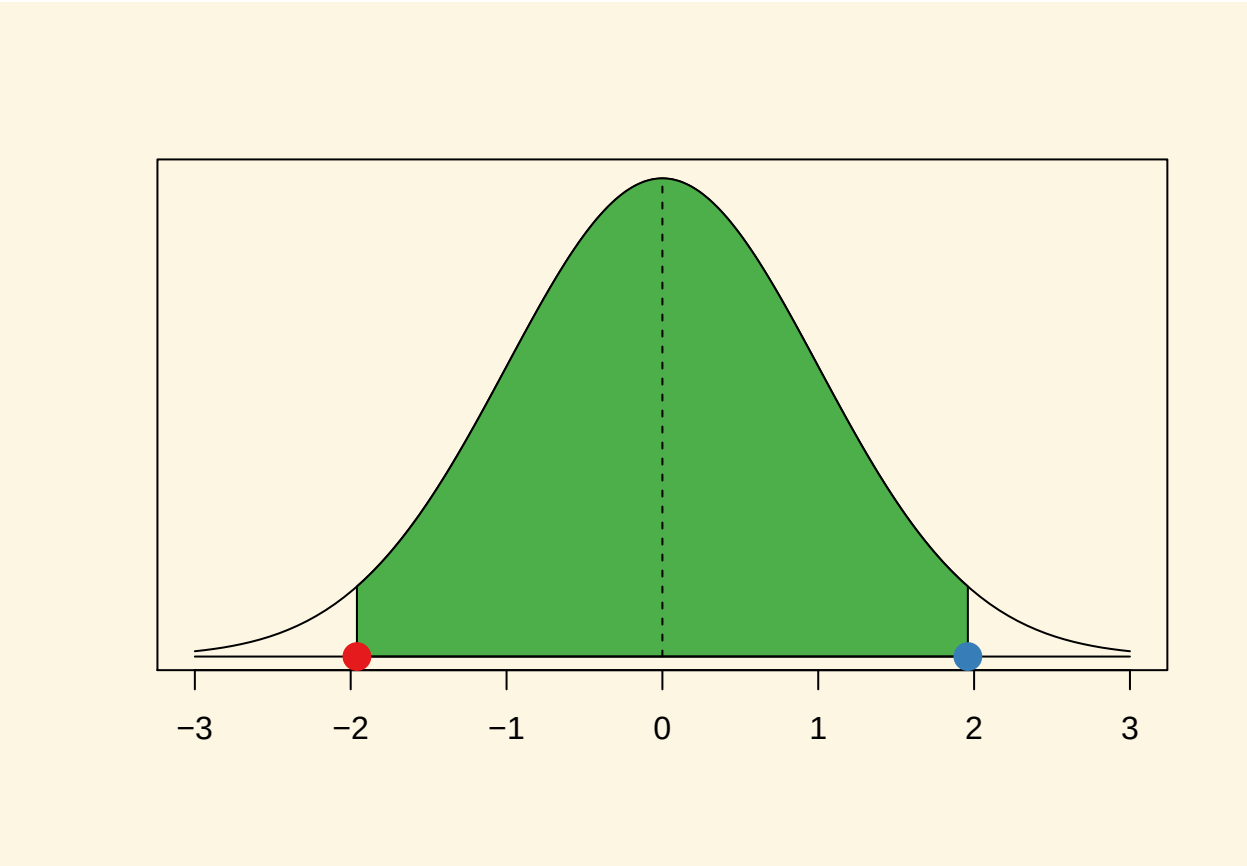


Now, let's get the critical value of where the right tail starts. That is at the cumulative probability of 97.5 percent.

```
qnorm(0.975, mean = 0, sd = 1)
```

```
[1] 1.959964
```

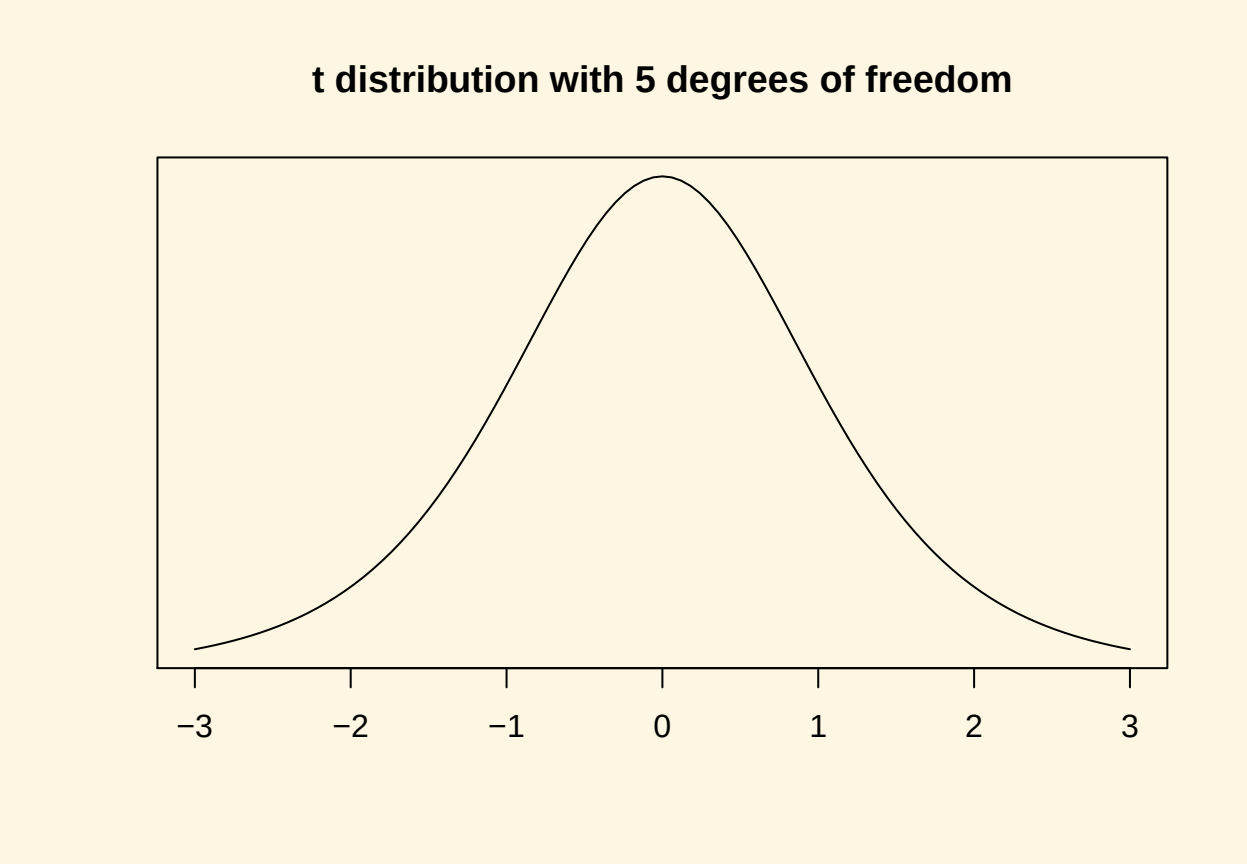
As you can see, this is the same number, only positive instead of negative. That's always the case because the normal distribution is symmetric. Let's add that point in blue to our graph.



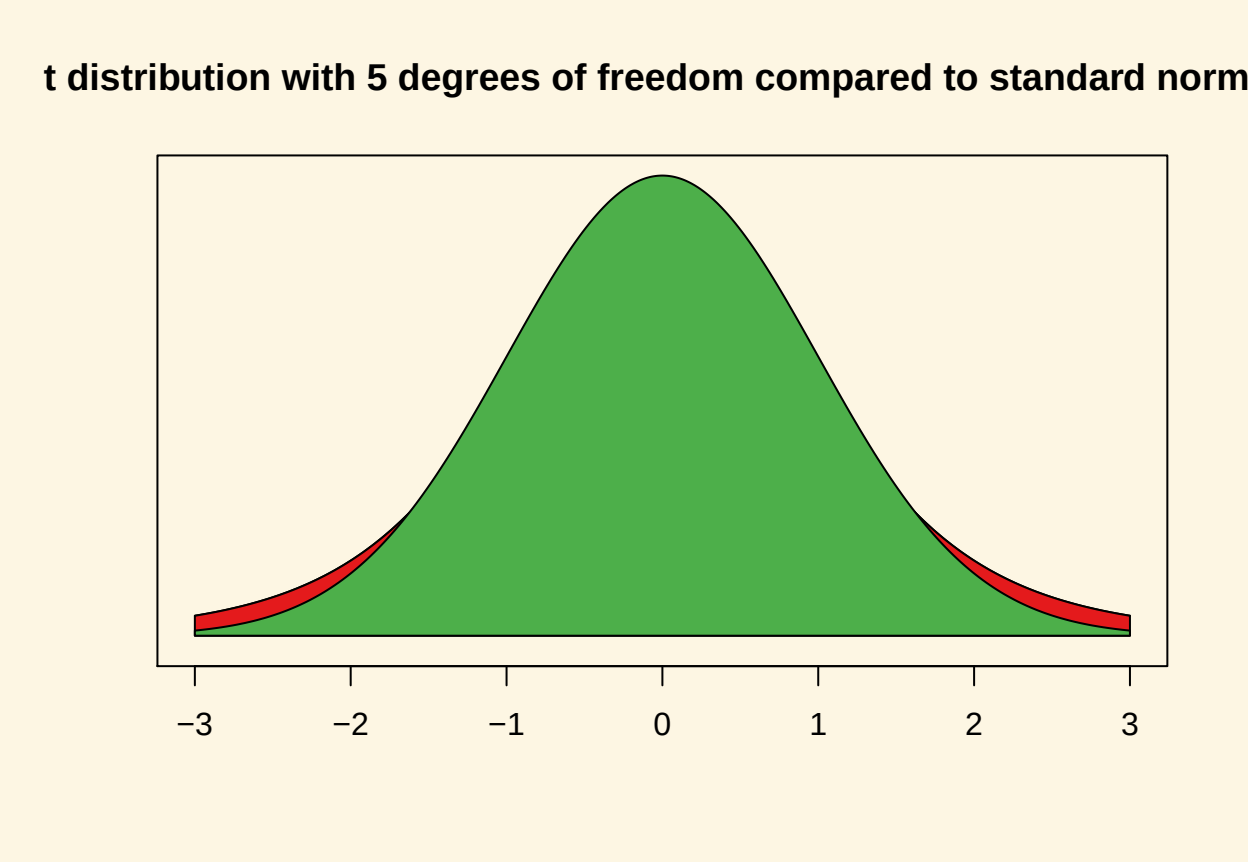
This is how we get the critical value for the 95 percent confidence interval. By the way, back in the day you would have to look up critical values in critical values tables at the end of statistics textbooks (you can find the tables in Stock and Watson and Kellstedt and Whitten.)

As you can see our red and blue dots are the borders of the green area, the 95 percent interval around the mean. You can get the critical values for any other interval (e.g., the 99 percent interval) similar to what we did just now.

We now do the same for the t distribution. In the t distribution, the critical value depends on the shape of the t distribution which is characterised by its degrees of freedom. Let's draw a t distribution with 5 degrees of freedom.



Although, it looks like a standard normal distribution, it is not. The t with 5 degrees of freedom has fatter tails. We show this by overlaying the t with a standard normal distribution.



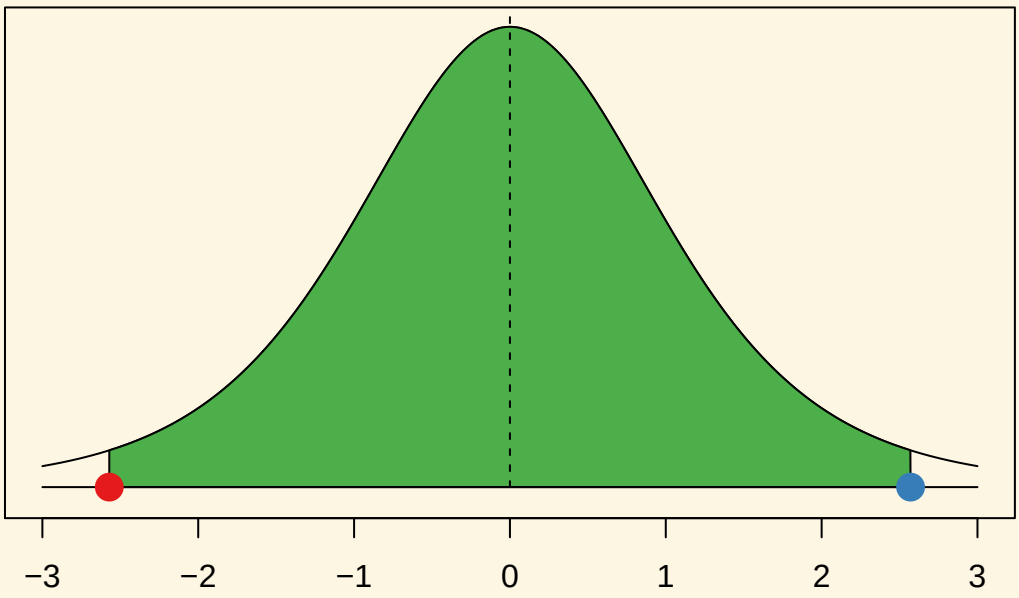
The red area is the difference between the standard normal distribution and the t distribution with 5 degrees of freedom.

The tails are fatter and that means that the probabilities of getting a value somewhere in the tails is larger. Lets calculate the critical value for a t distribution with 5 degrees of freedom.

```
# value for cumulative probability 95 percent in the t distribution with 5 degrees of freedom
qt(0.975, df = 5)
```

```
[1] 2.570582
```

See how much larger that value is than 1.96. Under a t distribution with 5 degrees of freedom 95 percent of the observations around the mean are within the interval from negative 2.5705818 to positive 2.5705818.



Let's illustrate that.

Remember the critical values for the t distribution are always more extreme or similar to the critical values for the standard normal distribution. If the t distribution has few degrees of freedom, the critical values (for the same percentage area around the mean) are much more extreme. If the t distribution has many degrees of freedom, the critical values are very similar.

4.1.3 T-test (difference in means)

We are interested in whether there is a difference in income between countries that have an independent judiciary and countries that do not have an independent judiciary. Put more formally, we are interested in the difference between two conditional means. Recall that a conditional mean is the mean in a subpopulation such as the mean of income given that the country has a free judiciary (conditional mean 1).

The t-test is the appropriate test statistic. Our interval-level dependent variable is *wealth* which is GDP per capita taken from the World Development Indicators of the World Bank. Our binary independent variable is h_j which is 1 if a country has a free judiciary and 0 otherwise.

Let's check the summary statistics of our dependent variable GDP per captia using the `summary()`.

```
summary(world.data$wealth)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  226.2  1768.0  5326.1 10184.1 12976.5 63686.7
```

Someone claims that countries with free judiciaries are usually richer than countries with controlled judiciaries. From the output of the `summary()` fucntion, we know that average wealth is 10184.0910395 US dollars across all countries—countries with and without free judiciaries.

We use the `which()` function from last week again, to identify the row-numbers of the countries in our dataset that have free judiciaries. Use the `which()` to get the row numbers of countries with free judiciaries.

```
which(world.data$h_j==1)
```

```
 [1]   8   9  13  14  18  23  28  33  39  40  41  42  43  44  50  52  54
[18]  55  60  70  71  72  74  75  76  77  80  82  84  85  90  93  94 104
[35] 105 107 110 112 114 115 118 126 127 131 143 144 145 146 149 153 154
[52] 155 157 160 163 165 166 167 168 169 170 171 178
```

Now, all we need is to index the dataset like we did last week. We access the variable that we want (*wealth*) with the dollar sign and the rows in square brackets. Take the mean of *wealth* for countries with a free judiciary on your own.

```
mean( world.data$wealth[which(world.data$h_j==1)])
```

```
[1] 17826.59
```

Go ahead and find the mean per capita wealth of countries with controlled judiciaries.

```
mean( world.data$wealth[which(world.data$h_j==0)])
```

```
[1] 5884.882
```

Finally, we run the t-test for the difference between two means.

```
# t.test for the difference between 2 means
t.test(world.data$wealth[which(world.data$h_j==1)], # mean 1
       world.data$wealth[which(world.data$h_j==0)], # mean 2
       mu = 0, # difference under the null hypothesis
       alt = "two.sided",  # two sided test (difference in means could be smaller or larger than 0)
       conf = 0.95) # confidence interval
```

```
    Welch Two Sample t-test

data:  world.data$wealth[which(world.data$h_j == 1)] and world.data$wealth[which(world.data$h_j == 0)]
t = 6.0094, df = 98.261, p-value = 0.00000003165
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  7998.36 15885.06
sample estimates:
mean of x mean of y
17826.591  5884.882
```

Let's interpret the results you get from `t.test()`. The first line tells us which groups we are comparing. In our example: Do countries with independent judiciaries have different mean income levels than countries without independent judiciaries?

In the following line you see the t-value, the degrees of freedom and the p-value. Knowing the t-value and the degrees of freedom you can check in a table on t distributions how likely you were to observe this data, if the null-hypothesis was true. The p-value gives you this probability directly. For example, a p-value of 0.02 would mean that the probability of seeing this data given that there is no difference in incomes between countries with and without independent judiciaries *in the population*, is 2%. Here the p-value is much smaller than this: 3.165e-08 = 0.00000003156!

In the next line you see the 95% confidence interval because we specified `conf=0.95`. If you were to take 100 samples and in each you checked the means of the two groups, 95 times the difference in means would be within the interval you see there.

At the very bottom you see the means of the dependent variable by the two groups of the independent variable. These are the means that we estimated above. In our example, you see the mean income levels in countries were the executive has some control over the judiciary, and in countries were the judiciary is independent.

Note that we are analysing a bi-variate relationship. The dependent variable is *wealth* and the independent variable is *h_j*.

Furthermroe, note that in the t test for the differences in means, the degrees of freedom depend on the variances in each group. You do not have to compute degrees of freedom for t tests for the differences in means yourself in this class—just use the `t.test()` function.

4.1.4 Estimating p values from t values

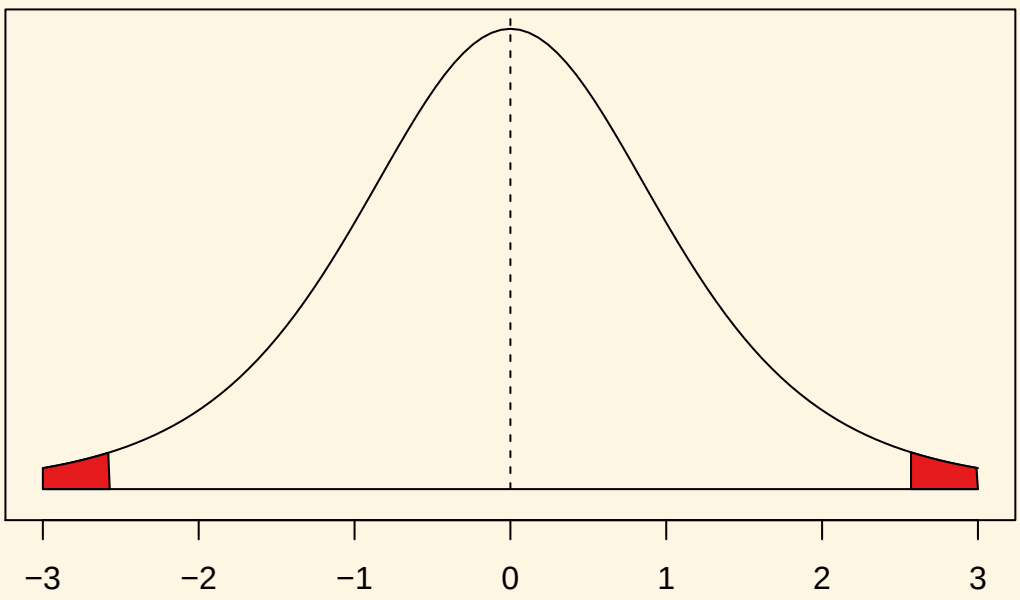
Estimating the p value is the reverse of getting a critical value. We have a t value and we want to know what the probability is to get such a value or an even more extreme value.

Let's say that we have a t distribution with 5 degrees of freedom. We estimated a t value of 2.9. What is the corresponding p value?

```
(1 - pt(2.9, df = 5)) *2
```

```
[1] 0.0337907
```

This is the probability of getting a t value of 2.9 or larger (or -2.9 or smaller) given that the null hypothesis is true. `pt(2.9, df = 5)` is the cumulative probability of getting a t value of 2.9 or smaller. But we want the probability of getting a value that is as large (extreme) as 2.9 or as small as -2.9. Therefore, we do `1 - pt(2.9, df = 5)`. We multiply by 2 to get both tails `(1 - pt(2.9, df = 5))*2`. This is the probability of getting a t value in the red tails of the distribution if the null hypothesis was true.



Clearly, the probability of getting such an extreme value (or something larger) under the assumption that the null hypothesis is true is very unlikely. The exact probability is ~0.03 (3 percent). We, therefore, think that the null hypothesis is false.

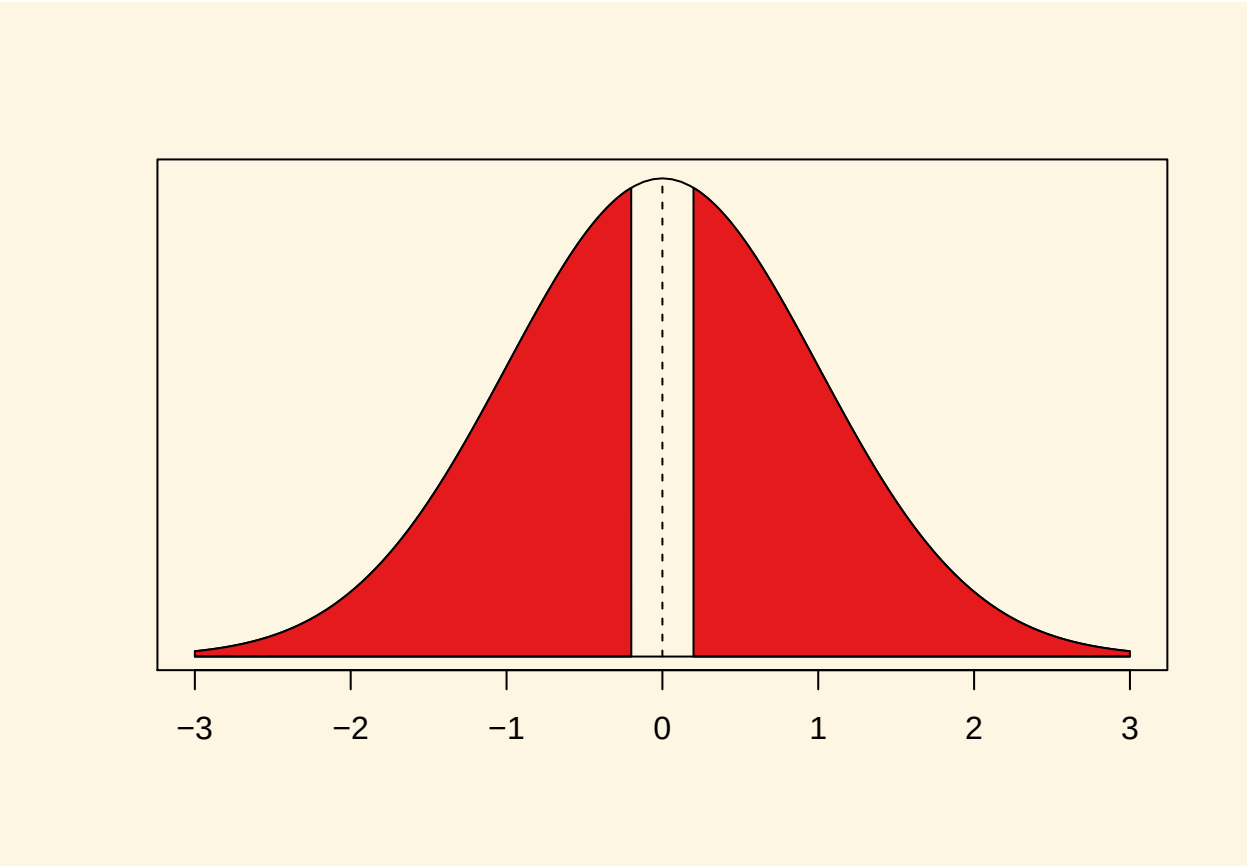
Let's estimate the p value in a normal distribution (it's actually better to always use the t distribution but the difference is negligible if the t distribution has many degrees of freedom).

Let's take our earlier example where we had estimated a t value of 0.1995059. Our friend claimed world income is 10 000 per capita on average and we estimated something slightly larger.

Let's check what the exact p value is in a normal distribution given a t value of 0.1995059.

```
(1 - pnorm(0.1995059))*2
```

```
[1] 0.841867
```



Clearly, it was not very unlikely to find a t value of 0.1995059 (that's the absolute value, i.e., a t value of negative or postive 0.1995059) under the assumption that the null hypothesis is true. Therefore, we cannot reject the null. The probability is 0.84 (84 percent)—highly likely.

4.1.5 Exercises

1. Create a new file called "assignment4.R" in your `statistics 1` folder and write all the solutions in it.
2. Turn former colonies into a factor variable and choose appropriate labels.
3. How many countries were former colonies? How many were not?
4. Find the means of political stability in countries that (1) were former colonies, (2) were not former colonies.
5. Is the the difference in means statistically significant?
6. In layman's terms, are countries which were former colonies more or less stable than those that were not?
7. How about if we choose an alpha level of 0.01?
8. What is the level of measurement of the United Nations Development index variable `undp_hdi`?
9. Check the claim that its true population mean is 0.85.
10. Calculate the t statistic.
11. Calculate the p value.
12. Construct a confidence interval around your mean estimate.
13. Discuss your findings in terms of the original claim. Interpret the t value, the p value, and the confidence interval.
14. Compute the critical value for the 99.9 percent confidence interval in a standard normal distribution.
15. Compute the critical value for the 99.9 percent confidence interval in a t distribution with 11 degrees of freedom.

16. Save the script that includes all previous tasks.
17. Source your script, i.e. run the entire script all at once without error message.

4.1.6 Optional Exercises that require reading Extra Info below

18. Create a scatter plot with latitude on the x-axis and political stability on the y-axis.
19. What is the correlation coefficient of political stability and latitude?
20. If we move away from the equator, how does political stability change?
21. Does it matter whether we go north or south from the equator?

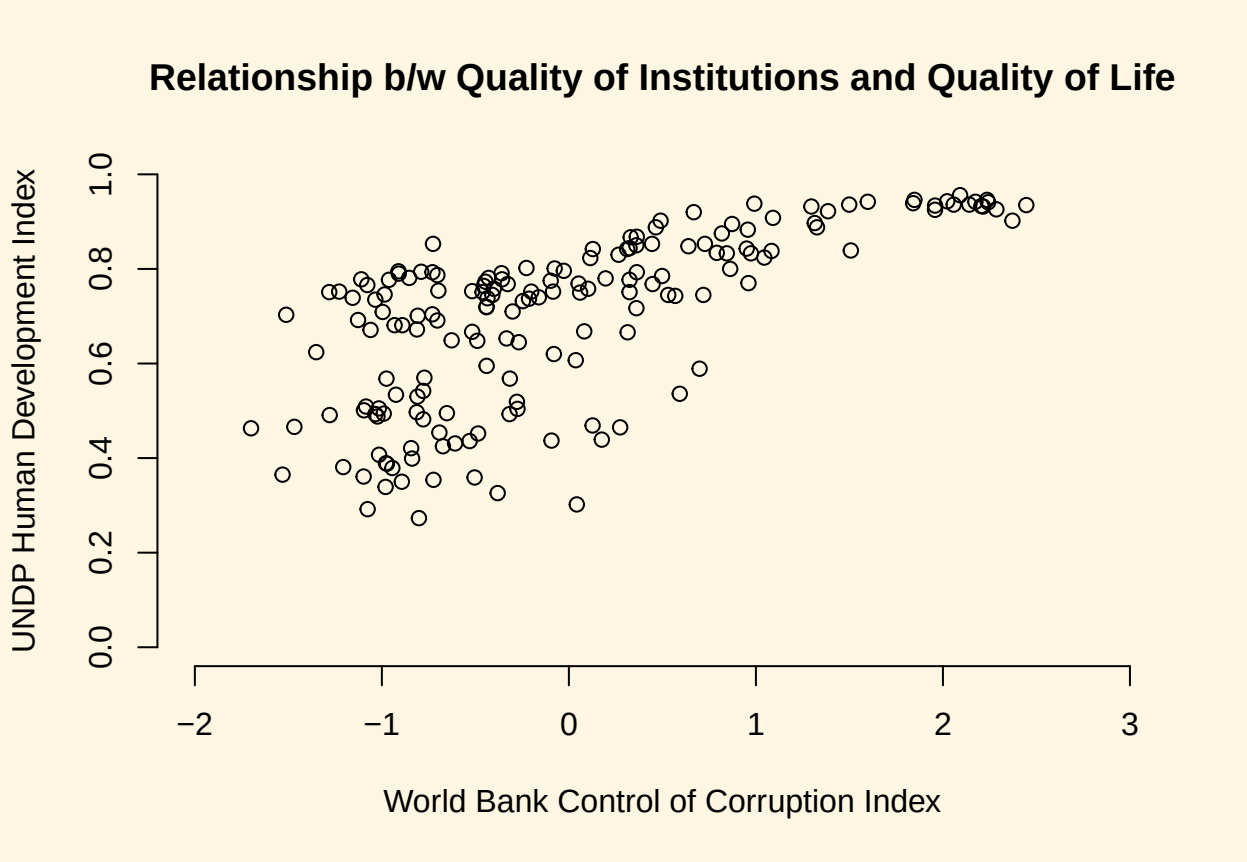
4.1.7 Advanced Exercises

22. Calculate the numerical difference in means (political stability conditional on colonialization) using the `means()` function.
23. Calculate the standard deviation of the difference in means (hint: using just the `sd()` function is incorrect in this context).
24. Is the difference in means more than 1.96 standard deviations away from zero? Interpret the result.
25. We claim the difference in means in terms of political stability between countries that were former colonies and those that were not is 0.3. Check this hypothesis.
26. An angry citizen who wants to defund the Department of International Development (DFID) claims that countries that were former colonies have reached 75% of the level of wealth of countries that were not colonised. Check this claim.

4.1.8 Extra Info

When we want to get an idea about how two continuous variables change together, the best way is to plot the relationship in a scatterplot. A scatterplot means that we plot one continuous variable on the x-axis and the other on the y-axis. Here, we illustrate the relation between the human development index `undp_hdi` and control of corruption `wbgi_cce`.

```
# scatterplot
plot(world.data$undp_hdi ~ world.data$wbgi_cce,
  xlim = c(xmin = -2, xmax = 3),
  ylim = c(ymin = 0, ymax = 1),
  frame = FALSE,
  xlab = "World Bank Control of Corruption Index",
  ylab = "UNDP Human Development Index",
  main = "Relationship b/w Quality of Institutions and Quality of Life"
  )
```



Sometimes people will report the correlation coefficient which is a measure of linear association and ranges from -1 to +1. Where -1 means perfect negative relation, 0 means no relation and +1 means perfect positive relation. The correlation coefficient is commonly used as as summary statistic. It's disadvantage is that you cannot see the non-linear relations which can using a scatterplot.

We take the correlation coefficient like so:

```
cor(y = world.data$undp_hdi, x = world.data$wbgi_cce, use = "complete.obs")
```

```
[1] 0.6813353
```

Argument	Description
`x`	The x variable that you want to correlate.
`y`	The y variable that you want to correlate.
`use`	How R should handle missing values. `use="complete.obs"` will use only those rows where neither `x` nor `y` is missing.

4.2 Solutions

4.2.0.1 Exercise 2

Turn former colonies into a factor variable and choose appropriate labels.

First, we load the dataset and then factorise the former colonies variable.

```
# load data
world.data <- read.csv("QoG2012.csv")
```

```
# turn variable into a factor
world.data$former_col <- factor(world.data$former_col, labels = c("not colonised", "was colonised"), le
```

4.2.0.2 Exercise 3

How many countries were former colonies? How many were not?

We can get the numbers from a frequency table.

```
table(world.data$former_col)
```

```
not colonised was colonised
           72           122
```

122 countries were victims of colonialization.

4.2.0.3 Exercise 4

Find the means of political stability in countries that (1) were former colonies, (2) were not former colonies.

We use the mean function to get the mean of political stability and subset for the groups of countries using the square brackets.

```
# mean of political stability in countries that were not colonised
mean(world.data$wbgi_pse[world.data$former_col=="not colonised"])
```

```
[1] 0.2858409
```

```
# mean of political stability in countries that were colonised
mean(world.data$wbgi_pse[world.data$former_col=="was colonised"])
```

```
[1] -0.231612
```

The average level of political stability in countries that were not colonised is 0.2858409. Mean political stability in countries that were colonised is -0.231612. The variable political stability `wbgi_pse` is an index. Larger values correspond with more political stability. We see that political stability is higher in countries that were not colonised.

Looking at this difference, we might conclude that the legacy of colonialism is still visible today and manifests itself in lower political stability. Let's investigate further to see whether the difference in means is statistically significant.

4.2.0.4 Exercise 5

Is the the difference in means statistically significant?

Let's first compute the difference in means. We call it `fd` here. We could name it anything but `fd` is shorthand for first difference. Differences between two means are sometimes referred to as first differences.

```
fd <- mean(world.data$wbgi_pse[world.data$former_col=="not colonised"]) - mean(world.data$wbgi_pse[worl
fd
```

```
[1] 0.5174529
```

The numerical difference is 0.5174529. Is this difference small or large? That is difficult to say because the variable is not measured in intuitive units (income in dollars is an example of a variable that is measured in intuitive units).

Let's look at the variable a little closer to understand this difference in substantive terms.

```r
# the range
range(world.data$wbgi_pse)
```

```
[1] -2.467461  1.675609
```

```r
# the distance from the minimum to the maximum value
diff(range(world.data$wbgi_pse))
```

```
[1] 4.14307
```

```r
# the difference in means as percentage change over the range of the variable
fd / diff(range(world.data$wbgi_pse))
```

```
[1] 0.124896
```

Doing the above was not necessary to answer the question but it is helpful to understand the variable in substantive terms. The difference in means is 0.5174529. That constitutes 0.124896 (12%) of the range of the variable. So the difference in political stability between countries that were colonies and those that were not is large.

We now test whether the difference is statistically significant using the t-test.

```r
# t test for difference in means
t.test(world.data$wbgi_pse[world.data$former_col == "not colonised"],
       world.data$wbgi_pse[world.data$former_col == "was colonised"],
       mu = 0, alt = "two.sided")
```

```
    Welch Two Sample t-test

data:  world.data$wbgi_pse[world.data$former_col == "not colonised"] and world.data$wbgi_pse[world.data$
t = 3.4674, df = 139.35, p-value = 0.0006992
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.2224004 0.8125053
sample estimates:
 mean of x  mean of y
 0.2858409 -0.2316120
```

As we can see, the difference is not only large it is also a noticeable systematic difference. The p value is small. Smaller than the conventional alpha level of 0.05. We can also look at the confidence interval which ranges from 0.2224004 to 0.8125053. So, if we were to repeatedly sample, the confidence interval of each sample would include the true population mean 95% of the time. Or more intuitively, we are 95% confident that the average population level of political stability is within our interval.

4.2.0.5 Exercise 6

In layman's terms, are countries which were former colonies more or less stable than those that were not?

The results from the t-test show that countries that were colonies are less stable than those that were not. The difference is large and systematic.

4.2.0.6 Exercise 7

How about if we choose an alpha level of 0.01?

The p-value is 0.0006992. That is smaller than an alpha level of 0.01 as well. Therefore, picking an alpha level of 0.01 does not change our results.

4.2.0.7 Exercise 8

What is the level of measurement of the United Nations Development index variable `undp_hdi`?

We googled united nations human development index and found that the variable is a composite index of three key dimensions: a long and healthy life, being knowledgeable and having a decent standard of living. The description goes on to tell us how the dimensions are measured and that the index is the geometric average of the components. The variable is, therefore, continuous.

4.2.0.8 Exercise 9

Check the claim that its true population mean is 0.85.

Let's estimate the mean from our sample

```
summary(world.data$undp_hdi)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
 0.2730  0.5390  0.7510  0.6982  0.8335  0.9560      19
```

Our estimate is 0.69824. The claim is that it is 0.85.

Null hypothesis: The true population mean of the human development index is: 0.85. Alternative hypothesis: The true population mean is different from 0.85.

We pick an alpha level of 0.05 for our test.

```
t.test(world.data$undp_hdi, mu = 0.85, alt = "two.sided")
```

```

    One Sample t-test

data:  world.data$undp_hdi
t = -11.139, df = 174, p-value < 0.00000000000000022
alternative hypothesis: true mean is not equal to 0.85
95 percent confidence interval:
 0.6713502 0.7251298
sample estimates:
mean of x
  0.69824
```

The p-value is lower than 0.05 and hence we reject the null hypothesis (hdi is 0.85). Looking at our confidence interval, we expect that if we were to repeatedly sample, the population mean would fall into the interval 0.6713502 to 0.7251298 95% of the time.

4.2.0.9 Exercise 10

Calculate the t statistic.

We could take the t statistic from the t test above. But we are going through the individual steps here.

From the `summary()` function above, we know that there are 19 missings on the variable which we drop. We create a copy of our data in the original state and then drop the missing rows from **world.data**.

```
# copy of world data
world.data.full <- world.data

# drop missings on undp_hdi
world.data <- world.data[ !is.na(world.data$undp_hdi),  ]
```

Step 1: We get the number of observations.

```
n <- length(world.data$undp_hdi)
n
```

```
[1] 175
```

Step 2: We calculate the standard variation of the random variable `undp_hdi`.

```
# sample standard deviation of undp_hdi
sd_hdi <- sd(world.data$undp_hdi)
```

Step 3: We calculate the standard error of the mean which is:

$$SE(\bar{Y}) = \frac{s_Y}{\sqrt{n}}$$

```
# standard error of the mean of undp_hdi
se.y_bar <- sd_hdi / sqrt(n)
```

Step 4: We compute the t-statistic as the difference between our estimated mean and the mean under null and then divide by the standard error of the estimated mean.

$$t = \frac{\bar{Y} - \mu_0}{SE(\bar{Y})}$$

Note: By dividing by the standard error of the estimated mean, we are making the difference of the means (our estimated mean - the mean under the null), free of the units that the variables were measured in. The t-value is the difference of the means, were its units are standard deviations.

The t-value follows a t-distribution with n-1 (174) degrees of freedom. It is well approximated by a standard normal distribution.

```
hdi_mean <- mean(world.data$undp_hdi)
t.statistic <- ( hdi_mean - 0.85 ) / se.y_bar
t.statistic
```

```
[1] -11.13908
```

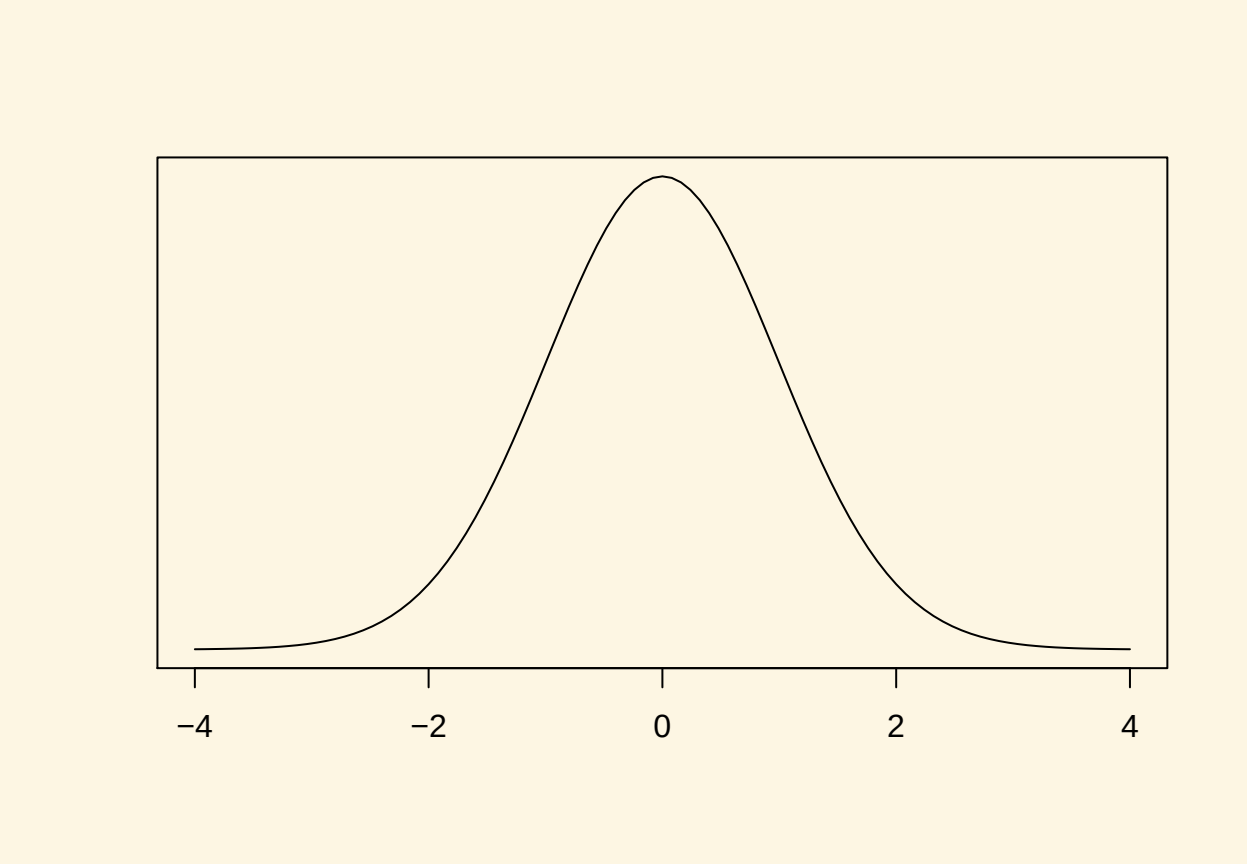
The t.statistic is -11.13908. Given that this t statistic follows a standard normal distribution approximately (because the sample is large), we would reject the null hypothesis if the t-value is more than 1.96 standard deviations away from 0. This t-value is 11 standard deviations away from zero. The t-value is very far in the tails of the distribution. It is very unlikely that we would have estimated the mean that we did estimate (0.69824) if the population average of HDI really was 0.85.

We can reject the null hypothesis.

4.2.0.10 Exercise 11

Calculate the p value.

We have the t statistic already. We will follow the steps to get the p value. But first, take a look at what the distribution of our t statistic looks like. Our t statistic is -11.13908. This is very very far in the left tail.



Step 1: We compute the cumulative probability to get our t-value or something smaller from this distribution.

```
cum.prob.at.least.t <- pt(t.statistic, df = (n-1) )
cum.prob.at.least.t
```

```
[1] 0.000000000000000000000002127538
```

Notice that our t value is negative. By calculating the probability to get a t-value as large or smaller, we alredy have the proability that we see a t value like this in the left tail. We are conducting a two-sided test. Therefore, we want the proability in the right tail as well, i.e., to see a t value that is 11.139 or larger. So in the next step we take the probability we got times 2.

```
cum.prob.at.least.t * 2
```

```
[1] 0.000000000000000000000004255075
```

The p value is extremely small. In fact, it is so small that the t-test did not report the correct p value either but simply stated that the p value is smaller than 2.2e-16.

Notice also that in the seminar we calucated `(1 - pt(t.statistic, df = n-1))*2`. This was because our t value was positive in the seminar. Here it is negative. Here in this task, we could have calculated `(1 - pt(abs(t.statistic), df = n-1))*2` where `abs(t.statistic)` returns the absolute value of the t statistic and hence turns it into a positive number.

If you do that, R will round and return the p value as 0. We are not holier than the pope and would have accepted any of these answers (0, <2.2e-16, 4.255075e-22).

4.2.0.11 Exercise 12

Construct a confidence interval around your mean estimate.

There are two ways to do this which yield slighly different answers.

Approach 1: Our sample is large. The t statistic follows approximately a standard normal distribution. We can, therefore, use the critical value from the standard normal distribution. 95% of the area under the standard normal distribution is within 1.96 standard deviations from the mean.

```
# lower bound
lb <- hdi_mean - 1.96 * se.y_bar
# upper bound
ub <- hdi_mean + 1.96 * se.y_bar
# results
lb
```

```
[1] 0.6715367
```

```
ub
```

```
[1] 0.7249432
```

```
# best guess of the population mean
hdi_mean
```

```
[1] 0.69824
```

Our best guess of the population mean is 0.69824. We are 95% confident that our interval from 0.6715367 to 0.7249432 includes the population mean.

Approach 2: We get the critical value from a t distribution with n-1 (174) degrees of freedom. To find the critical value, we use the t distribution's quantile function `qt(cumulative.probability, degrees of freedom)`.

```
t.critical <- qt(0.975, df = 174)
t.critical
```

```
[1] 1.973691
```

So, in our t distribution 95% are within precisely 1.973691 standard deviations from the mean.

Notice, that the confidence interval covers 95% of the area under the curve and is centered on the mean. There are 2.5% in each tail. `qt()` gives us the critical value at the cumulative probability of whatever we enter into the function. We input .975, so we get the critical value at the point where the right tail starts.

```
# lower bound
lb2 <- hdi_mean - t.critical * se.y_bar
# upper bound
ub2 <- hdi_mean + t.critical * se.y_bar
# results
lb2
```

```
[1] 0.6713502
```

```
ub2
```

```
[1] 0.7251298
```

Using this more exact approach, we show that the 95% confidence interval covers values of the human development index from 0.6713502 to 0.7251298.

4.2.0.12 Exercise 13

Discuss your findings in terms of the original claim. Interpret the t value, the p value, and the confidence interval.

We can reject the original claim that the mean of HDI is 0.85. Given, an alpha level of 0.05, our large t-value (-11.13908 - the absolute value is large) implies that our estimate (0.69824) is extremely unlikely assuming that 0.85 is the population mean. From exercise 11, we know the p-value which is the probability that we have mistakenly rejected a correct null hypothesis. That probability is

$$4.255075e^{-22} = 4.255075 * 10^{-22} = \frac{4.255075}{10^{22}} = 0.0000000000000000000004255075$$

.

Our best guess of the population mean is 0.7 ± 0.03 (rounded to two digits).

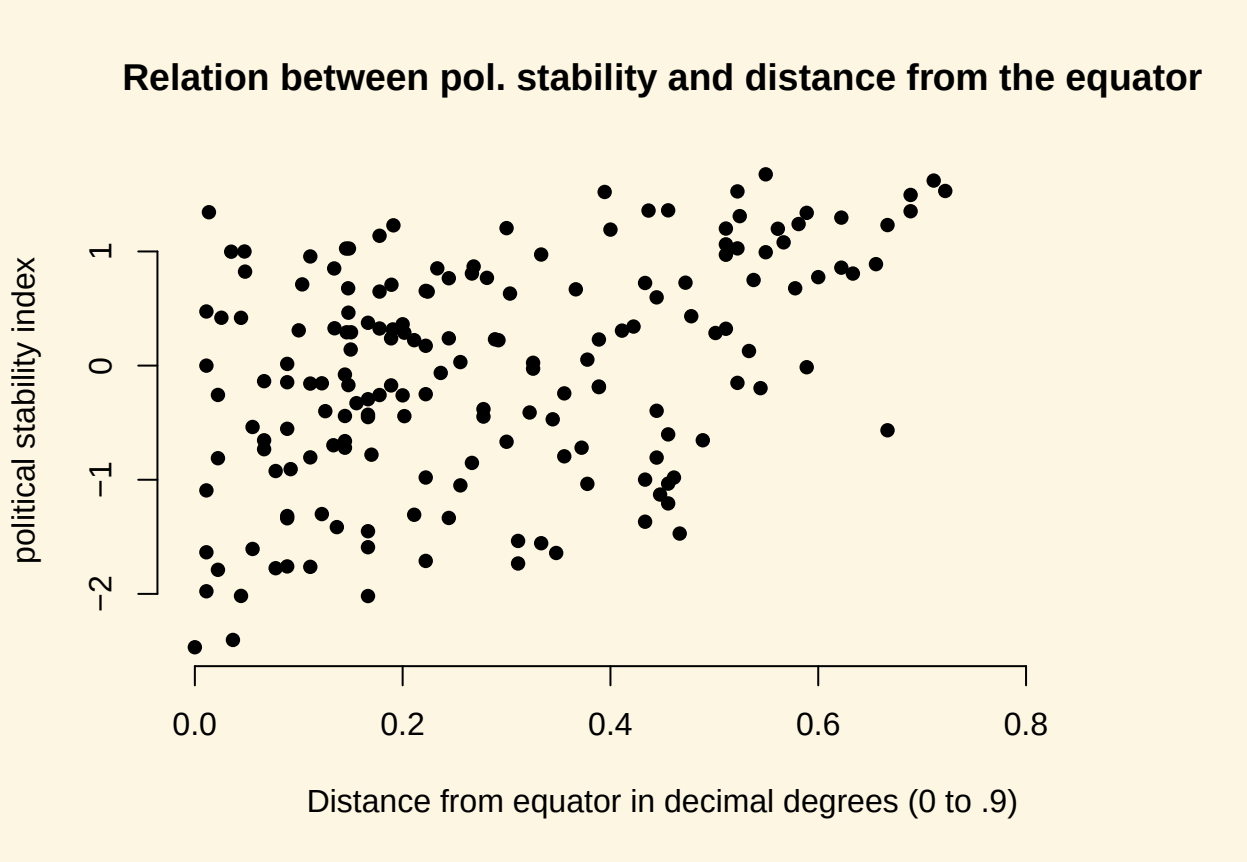
4.2.1 Optional Exercises that require reading Extra Info below

4.2.1.1 Optional Exercise 1

Create a scatter plot with latitute on the x-axis and political stability on the y-axis.

The relationship between two continuous variables is best illustrated with a scatterplot.

```
plot(
  y = world.data$wbgi_pse,
  x = world.data$lp_lat_abst,
  pch = 16,
  bty = "n",
  xlim = c(0, 0.9),
  main = "Relation between pol. stability and distance from the equator",
  xlab = "Distance from equator in decimal degrees (0 to .9)",
  ylab = "political stability index"
  )
```



4.2.1.2 Optional Exercise 2

What is the correlation coefficient of political stability and latitude?

```
cor(world.data$wbgi_pse, world.data$lp_lat_abst, use = "complete.obs")
```

```
[1] 0.4238086
```

the correlation coefficient ranges from -1 to 1 and is a measure of linear association of two continuous variables. The variables are positively related. That means, as we move away from the equator, we tend to observe higher levels of political stability.

4.2.1.3 Optional Exercise 3

If we move away from the equator, how does political stability change?

Political stability tends to increase as we move away from the equator.

4.2.1.4 Optional Exercise 4

Does it matter whether we go north or south from the equator?

It does not matter whether we go north or south. Our latitude is measured in decimal degrees. Thus, a value of 0.9 could correspond to either the North Pole or the South Pole.

4.2.2 Advanced Exercises

These exercises were hard and go beyond what we expect from you at this point. Good job, if you were able to solve them!

4.2.2.1 Advanced Exercise 1

Calculate the numerical difference in means (political stability conditional on colonialization) using the `means()` function.

Notice: Earlier we dropped missing values on **undp_hdi**. We thereby dropped values that were not missing on **wbgi_pse** from the dataset. That was valuable information. Not reloading the dataset would be a mistake.

```
# use full dataset
world.data <- world.data.full

table(world.data$former_col) # check the labels of the former colonies variable
```

```
not colonised was colonised
           72           122
```

```
# mean political stability of not colonised group
mean.not.col <- mean(world.data$wbgi_pse[world.data$former_col=="not colonised"])
mean.not.col
```

```
[1] 0.2858409
```

```
# mean political stability of colonised group
mean.was.col <- mean(world.data$wbgi_pse[world.data$former_col=="was colonised"])
mean.was.col
```

```
[1] -0.231612
```

```
# difference in means
fd <- mean.not.col - mean.was.col
fd
```

```
[1] 0.5174529
```

The difference in means is 0.5174529. Countries that were not colonised are more politically stable.

4.2.2.2 Advanced Exercise 2

Calculate the standard deviation of the difference in means (hint: using just the `sd()` function is incorrect in this context).

The standard error of the difference between two means is:

$$\sqrt{\frac{\sigma_{Y_A}^2}{n_A} + \frac{\sigma_{Y_B}^2}{n_B}}$$

Where A and B are the two groups.

```
# variance of political stability in the two groups
var_not_col <- var(world.data$wbgi_pse[world.data$former_col=="not colonised"])
var_was_col <- var(world.data$wbgi_pse[world.data$former_col=="was colonised"])
```

```
# number of observations in each group
n_not_col <- length(world.data$wbgi_pse[world.data$former_col=="not colonised"])
n_was_col <- length(world.data$wbgi_pse[world.data$former_col=="was colonised"])

# standard error of the difference in means
fd.se <- sqrt( (var_not_col/n_not_col) + (var_was_col/n_was_col)   )
fd.se
```

```
[1] 0.1492324
```

The standard error of the difference in means is 0.1492324.

4.2.2.3 Advanced Exercise 3

Is the difference in means more than 1.96 standard deviations away from zero? Interpret the result.

```
fd - 2*fd.se
```

```
[1] 0.2189881
```

The difference in means is further than two standard deviations away from zero. Given an alpha level of 0.05, we can reject the null hypothesis that political stability is the same in countries that were colonised and those that were not.

4.2.2.4 Advanced Exercise 4

We claim, the difference in means in terms of political stability between countries that were former colonies and those that were not is 0.3. Check this hypothesis.

We use the t-test for the difference in means to test this claim. Normally, our null would be that the difference in means is zero, i.e. there is no difference. Here, the claim is that the difference is 0.3. So, all we have to do is adjust the null hypothesis accordingly.

```
t.test(world.data$wbgi_pse[world.data$former_col == "not colonised"],
       world.data$wbgi_pse[world.data$former_col == "was colonised"],
       mu = 0.3, alt = "two.sided")
```

```
    Welch Two Sample t-test

data:  world.data$wbgi_pse[world.data$former_col == "not colonised"] and world.data$wbgi_pse[world.data$
t = 1.4571, df = 139.35, p-value = 0.1473
alternative hypothesis: true difference in means is not equal to 0.3
95 percent confidence interval:
 0.2224004 0.8125053
sample estimates:
 mean of x  mean of y
 0.2858409 -0.2316120
```

We cannot reject the claim that the true difference in means is 0.3.

4.2.2.5 Advanced Exercise 5

First, we drop missings from **wdi_gdpc**.

```
world.data <- world.data[ !is.na(world.data$wdi_gdpc), ]
```

An angry citizen who wants to defund the Department of International Development (DFID) claims that countries that were former colonies have reached 75% of the level of wealth of countries that were not colonised. Check this claim.

The null hypothesis is that there is no difference between the level of wealth in countries that were former colonies and 0.75 times the level of wealth in countries that were not former colonies

```
# the claim of the citizen
claim <- mean(world.data$wdi_gdpc[world.data$former_col=="not colonised"]) * 0.75
claim
```

```
[1] 12311.54
```

```
# estimated level of wealth in countries that were colonised
estimate <- mean(world.data$wdi_gdpc[world.data$former_col=="was colonised"])
estimate
```

```
[1] 6599.714
```

Our estimate is roughly half the angry citizen's claim. The substantial difference is huge. To cover all our bases, let's perform the t-test for the difference in means.

How would we do this though? This one is tricky because the citizen's claim is not actually in our data. At the same time we don't know the true level of wealth in countries that were not colonised, we only have an estimate. We have to manipulate our data to get there.

First, we create a copy of the wealth variable with a new name.

```
world.data$angry_gdp <- world.data$wdi_gdpc
```

Now, we adjust the level of wealth in the group of countries that were not colonised down to the citizen's claim. The citizen's claim is that we should then not find a difference in means between the two groups ("was colonised" and "not colonised") anymore.

```
world.data$angry_gdp[world.data$former_col=="not colonised"] <- world.data$angry_gdp[world.data$former_
mean(world.data$angry_gdp[world.data$former_col=="not colonised"])
```

```
[1] 12311.54
```

We can see that the level of mean wealth of our manipulated variable for countries that were not colonised now corresponds to the citizen's claim. We can now check the difference in means.

```
t.test(world.data$angry_gdp[world.data$former_col=="not colonised"],
       world.data$angry_gdp[world.data$former_col=="was colonised"],
       mu = 0, alt = "two.sided")
```

```
    Welch Two Sample t-test

data:  world.data$angry_gdp[world.data$former_col == "not colonised"] and world.data$angry_gdp[world.da
t = 3.6218, df = 127.72, p-value = 0.0004206
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 2591.29 8832.37
sample estimates:
mean of x mean of y
12311.544  6599.714
```

Clearly, we can reject the citizen's claim. Our p value implies that the probability that we see this huge difference in our data, given that there really is no difference, is 0.04%. Our conventional alpha level is 5%.

5 Revision: Sample Variance and Sample Standard Deviation; Hypothesis testing and Confidence Intervals

In this seminar, we revise the concepts of the standard deviation, the sampling variance, hypothesis testing and confidence intervals. We will also learn some more data manipulation and working with the random number generator.

5.1 Seminar

Let's remove all objects from our workspace and set the working directory.

```
rm(list=ls())
setwd("~/statistics1")
```

5.1.1 Sample Variance and Sample Standard Deviation

The sample variance and sample standard deviation inform us about the degree of variability of our data. Suppose, we were to roll the dice 5 times. We could then compute the mean value that we roll. The sample standard deviation measures by how much an average roll of the dice will deviate from the mean value.

We start rolling the dice, using R's random number generator and the `runif()` function. The function randomly draws numbers from a uniform distribution. In a uniform distribution each value has the same probability of being drawn. All six sides of a die should be equally likely if the die is fair. Hence, the uniform distribution.

`runif()` takes three arguments. `n` is the number of values to be drawn. `min` is the minimum value and `max` is the maximum value.

```
# random draw of 10 values from a uniform distribution
dice <- runif(n = 10, min = 1, max = 7)
dice
```

```
 [1] 3.139090 6.424829 2.061919 1.941946 5.979947 5.961752 2.637821
 [8] 1.071156 6.098825 1.694422
```

We have indeed drawn 10 numbers but they are not integers as we would like—we want to simulate a die, so the values should be 1, 2, 3, 4, 5 or 6. We will return to this in a moment but for now let's return to the randomness. Let's draw 10 numbers again:

```
# random draw of 10 values from a uniform distribution
dice2 <- runif(n = 10, min = 1, max = 7)

# first draw
dice
```

```
 [1] 3.139090 6.424829 2.061919 1.941946 5.979947 5.961752 2.637821
 [8] 1.071156 6.098825 1.694422
```

```
# second draw
dice2
```

```
 [1] 4.621894 3.806171 3.045458 4.204994 5.187980 2.319311 6.362910
 [8] 3.725839 4.536423 3.768513
```

The numbers of the first and second roll differ because we have drawn values at random. To make our results replicate and to ensure that everyone in the seminar works with the same numbers, we set R's random number

generator with the `set.seed()` function. As argument we plug in some arbitrary value (it does not matter which but using a different one will lead to a different quasi-random draw).

```
# set random number generator
set.seed(123)

# random draw of 10 values from a uniform distribution
dice <- runif(n = 10, min = 1, max = 7)
dice
```

```
 [1] 2.725465 5.729831 3.453862 6.298104 6.642804 1.273339 4.168633
 [8] 6.354514 4.308610 3.739688
```

You should all have the same values. If not, run `set.seed()` again and then do the random draw once. If you do it more than once, the numbers will change. Let's see how this works:

```
# set random number generator
set.seed(123)

# 1st random draw of 10 values from a uniform distribution
dice <- runif(n = 10, min = 1, max = 7)
dice
```

```
 [1] 2.725465 5.729831 3.453862 6.298104 6.642804 1.273339 4.168633
 [8] 6.354514 4.308610 3.739688
```

```
# 2nd random draw of 10 values from a uniform distribution
dice2 <- runif(n = 10, min = 1, max = 7)
dice2
```

```
 [1] 6.741000 3.720005 5.065424 4.435800 1.617548 6.398950 2.476526
 [8] 1.252357 2.967524 6.727022
```

```
# reset random number generator
set.seed(123)

# 3rd random draw of 10 values from a uniform distribution
dice3 <- runif(n = 10, min = 1, max = 7)
dice3
```

```
 [1] 2.725465 5.729831 3.453862 6.298104 6.642804 1.273339 4.168633
 [8] 6.354514 4.308610 3.739688
```

```
# 4th random draw of 10 values from a uniform distribution
dice4 <- runif(n = 10, min = 1, max = 7)
dice4
```

```
 [1] 6.741000 3.720005 5.065424 4.435800 1.617548 6.398950 2.476526
 [8] 1.252357 2.967524 6.727022
```

As you can see, the the draws from **dice** and **dice3** are the same and the draws from **dice2** and **dice4** are the same as well. Let's make the values integers with the `as.integer()` function which simply cuts off all decimal places.

```
# reset random number generator
set.seed(123)
# random draw of 10 numbers from a uniform distribution with minimum 1 and maximum 7
dice <- runif(10, 1, 7)
# cut off decimals places
dice <- as.integer(dice)
```

```
dice
```

```
 [1] 2 5 3 6 6 1 4 6 4 3
```

```
# frequency of dice rolls
table(dice)
```

```
dice
1 2 3 4 5 6
1 1 2 2 1 3
```

We have rolled a six relatively often. All sides should be equally likely but due to sampling variability, we have rolled the six most often. The expected value of a die is 3.5. That is:

$$1 \times \frac{1}{6} + 2 \times \frac{1}{6} + 3 \times \frac{1}{6} + 4 \times \frac{1}{6} + 5 \times \frac{1}{6} + 6 \times \frac{1}{6} = 3.5$$

We compute the mean in our sample and the standard deviation. Let's start with the mean. Do so yourself but do not use the in-built function.

```
dice.sum <- dice[1] + dice[2] + dice[3] + dice[4] + dice[5] + dice[6] + dice[7] + dice[8] + dice[9] + di
dice.mean <- dice.sum / 10
dice.mean
```

```
[1] 4
```

We would have gotten the same result from the `mean()` function.

The sample standard deviation tells by how much an average roll of the dice differs from the estimated sample mean. Estimate the sample standard deviation on your own without using the `sd()` function. Do not copy. Type everything yourself.

```
numerator <- ( (dice[1] - dice.mean)^2 + (dice[2] - dice.mean)^2 + (dice[3] - dice.mean)^2  +
                 (dice[4] - dice.mean)^2 + (dice[5] - dice.mean)^2 + (dice[6] - dice.mean)^2 +
                 (dice[7] - dice.mean)^2 + (dice[8] - dice.mean)^2 + (dice[9] - dice.mean)^2 +
                 (dice[10] - dice.mean)^2 )
std.dev <- sqrt( (numerator / 9) )
std.dev
```

```
[1] 1.763834
```

An average deviation from the sample mean is 1.76.

5.1.2 T test for the sample mean

Our estimate of the mean is 4. The expected value is 3.5. Is this evidence that the die is loaded? The null hypothesis is that the die is fair. The alternative hypothesis is that the die is loaded.

Answer this question on your own by computing and interpreting the t value.

```
# standard error of the sample mean
std.err <- std.dev / sqrt(10)

# t value
t.value <- (4 - 3.5) / std.err
t.value
```

```
[1] 0.8964215
```

Clearly, we cannot reject the null hypothesis. The estimated difference between our sample mean and the population mean is 0.5. This value is 0.9 standard errors away from the population mean under the null hypothesis (3.5). We do not know the critical value for the t distribution here because we are in a small sample. However, it must be bigger than 1.96. Our t value is clearly smaller, therefore, we ***cannot*** reject the null hypothesis. That is not surprising, given that we have drawn from a uniform distribution, i.e., our sampling process was fair by definition.

Let's back up. The standard error quantifies the average difference between mean estimates that are due to sampling variability (chance). Put differently, the standard error approximates the average difference between the population mean and our sample estimate.

Now. Compute the variance of the mean on your own. Try yourself before you check the code.

```
# sample variance
variance <- numerator / 9

# variance of the mean
var.mean <- variance / 10
var.mean
```

```
[1] 0.3111111
```

The correct answer is 0.31. Now, compute the standard error of the mean from your variance of the mean. Try yourself.

$$s^2_{\bar{X}} = \frac{s^2_x}{n} = (\sqrt{\frac{s_x}{\sqrt{n}}})^2$$

```
# variance of the mean
sqrt(var.mean)
```

```
[1] 0.5577734
```

Let's estimate the p value from a t distribution with the correct amount of degrees of freedom.

```
p.value <- (1 - pt(t.value, df = 9))*2
p.value
```

```
[1] 0.3933733
```

The probability that we roll a fair die 10 times and get a sample mean of 4 is 39 percent. That is not unlikely at all. We are far from rejecting the null hypothesis.

We now construct the confidence interval around our mean estimate. To re-cap, we construct the confidence interval as:

$$\bar{Y} \pm \text{critical value} \times SE(\bar{Y})$$

First, we need the critical value (for an alpha level of 0.05). We get the critical value from the quantile function of the t distribution with $n-1$ degrees of freedom.

```
qt(p = 0.975, df = 9)
```

```
[1] 2.262157
```

where `p` is the cumulative probability and `df` is the degrees of freedom. The critical value in a t distribution with 9 degrees of freedom is 2.262157.

Construct the lower and upper bounds of the confidence interval yourself.

```r
# lower bound
lb <- dice.mean - qt(p = 0.975, df = 9) * std.err
# upper bound
ub <- dice.mean + qt(p = 0.975, df = 9) * std.err

# confidence interval
lb
```

```
[1] 2.738229
```

```r
ub
```

```
[1] 5.261771
```

With 95 percent probability (where the probability is the long-run relative frequency), the population mean is within the confidence interval.

Clearly, the population mean under the assumption that the null hypothesis is true (3.5), is within this interval.

Run a t test using the `t.test()` function yourself.

```r
t.test(dice,
       mu = 3.5,
       conf.level = .95)
```

```

    One Sample t-test

data:  dice
t = 0.89642, df = 9, p-value = 0.3934
alternative hypothesis: true mean is not equal to 3.5
95 percent confidence interval:
 2.738229 5.261771
sample estimates:
mean of x
        4
```

The `t.test()` function is more convenient than estimating everything by hand. The results are the same.

5.1.3 T test for the difference in means

We will now test the difference between two dice. First, remove everything from the workspace on your own.

```r
# remove everything from the workspace
rm(list = ls())
```

Set the random number generator to 1234.

```r
set.seed(1234)
```

Generate a vector of 100 rolls of a fair dice on your own and show the results in a frequency table.

```r
fair.die <- runif(n = 100, min = 1, max = 7)
fair.die <- as.integer(fair.die)
table(fair.die)
```

```
fair.die
 1  2  3  4  5  6
21 28  6 20 13 12
```

The absolute number are not that great to see the distribution of values at a glance. Create a table that shows the proportions of each outcome.

```
table(fair.die) / sum(table(fair.die))
```

```
fair.die
   1    2    3    4    5    6
0.21 0.28 0.06 0.20 0.13 0.12
```

Calculate the mean.

```
mean(fair.die)
```

```
[1] 3.12
```

Although, the proportions look like we rolled the lower numbers way too often, we are not too far away from the population mean (3.5).

Now, we create a loaded die. We draw randomly from a normal distribution with mean 4.5 and standard deviation 1.5.

```
loaded.die <- rnorm(n = 100, mean = 4.5, sd = 1.5)
loaded.die <- as.integer(loaded.die)
table(loaded.die)
```

```
loaded.die
 1  2  3  4  5  6  7  8
 2 13 19 28 23  8  6  1
```

Oops, we rolled 6 seven's and 1 eight. Let's change these to sixes. We use the square brackets to index the elements of the **loaded.die** vector. We use the `which()` function to identify the elements that are greater than 6. Finally, we change these values to 6.

```
loaded.die[ which(loaded.die > 6) ] <- 6
table(loaded.die)
```

```
loaded.die
 1  2  3  4  5  6
 2 13 19 28 23 15
```

Now, estimate the 2 means (of the fair die and the loaded die) yourself. Next, estimate the difference in means. You may use the `mean()` function.

```
# mean in the 2 groups
mean(fair.die)
```

```
[1] 3.12
```

```
mean(loaded.die)
```

```
[1] 4.02
```

```
# first difference (difference in means)
fd <- mean(fair.die) - mean(loaded.die)
fd
```

```
[1] -0.9
```

The difference suggests that we roll larger values with the loaded die. Is the difference statistically detectable? To find out, we carry out the t test for the difference in means. Our dependent interval scaled variable is the value of the roll of the dice. The our independent binary variable is whether the die is loaded or not.

Compute the standard error for the difference in means on your own.

$$SE(Y_{X=0} - Y_{X=1}) = \sqrt{\frac{s^2_{Y_{X=0}}}{n_{X=0}} + \frac{s^2_{Y_{X=1}}}{n_{X=1}}}$$

where $s^2_{Y_{X=0}}$ is the variance in the first group and $s^2_{Y_{X=1}}$ is the variance in the second group. The number of observations in the first group is $n_{X=0}$ and the number of observations in the second group is $n_{X=1}$.

The result is 0.2161462. Try until you get it. You may use the `var()` function.

```
# standard error of the first difference
se.fd <- sqrt( ((var(fair.die) / 100) + (var(loaded.die) / 100)) )
se.fd
```

```
[1] 0.2161462
```

Now, that we have the standard error of the difference in means, compute the t statistic on your own (without using the `t.test()` function).

```
# standard error of the first difference
t.value <- (mean(fair.die) - mean(loaded.die)) / se.fd
t.value
```

```
[1] -4.163848
```

The t value is large and our sample size is also large. We can take the critical from a normal distribution. For an alpha level of 0.05, the critical value is 1.96.

Construct the confidence interval for the difference in means yourself.

```
# lower bound
lb <- fd - 1.96 * se.fd
# upper bound
ub <- fd + 1.96 * se.fd

lb
```

```
[1] -1.323647
```

```
ub
```

```
[1] -0.4763534
```

The confidence interval for the difference in means ranges from -1.32 to -0.48. Clearly, the difference in means is smaller than 0. We can reject the null hypothesis that there is no difference between the loaded die and the fair die because 0 is not within the confidence interval.

Let's estimate the p value: i.e., the probability that we estimate a difference in means of -0.9 given that there really is no difference between the fair die and the loaded die.

The result is 3.129287e-05. Try until you get it. You can use the `pnrom()` function to get the cumulative probability from a standard normal distribution.

```
# 1st way
pnorm( t.value ) * 2
```

```
[1] 0.00003129287
```

```
# 2nd way
(1 - pnorm( abs(t.value) )) *2
```

```
[1] 0.00003129287
```

We can reject the null hypothesis. The loaded die is different from the fair die.

5.1.4 Exercises

1. Create a vector of a fair die and a vector of a loaded die with (25) observations such that you cannot distinguish the dice with a difference in means test. Carry out the t-test.
2. Re-create the dice but increase the number of observations to 1000. Does the result of the t test change?
3. Ordinary Economic Voting Behavior in the Extraordinary Election of Adolf Hitler Download and then load `who_voted_nazi_in_1932.csv`. who voted nazi in 1932 data Codebook:

Variable	Description
sharenazis	Percent of the vote Nazis received in the district
nazivote	Number of Nazi votes
nvoter	Total number of eligible voters
shareblue	Percent of blue-collar potential voters
sharewhite	Percent of white-collar potential voters
shareself	Percent of self-employed potential voters
sharedomestic	Percent of domestically employed potential voters
shareunemployed	Percent of unemployed potential voters

4. Estimate the means and illustrate the distribution of potential voter shares by class.
5. Estimate the mean vote shares for the Nazis in districts where the share of blue-collar voters was above the mean (30.82) and below it.
6. Construct confidence intervals around the means.
7. Are there differences between the groups? Use the appropriate statistical test to find out.
8. Calculate t values and p values on your own (without using the t test)
9. Interpret your results substantially.
10. Estimate the mean vote shares for the Nazis in districts where the share of white-collar voters was above the mean (11.423) and below it.
11. Construct confidence intervals around the means.
12. Are there differences between the groups? Use the appropriate statistical test to find out.
13. Calculate t values and p values on your own (without using the t test)
14. Interpret your results substantially.

5.2 Solutions

5.2.0.1 Exercise 1

Create a vector of a fair die and a vector of a loaded die with (25) observations such that you cannot distinguish the dice with a difference in means test. Carry out the t-test.

```
# set random number generator
set.seed(123456)

# fair die
die1 <- as.integer(runif(25, min = 1, max = 7))
# loaded die

die2 <- as.integer(rnorm(25, mean = 5, sd = 1.5))
die2[which(die2 < 0)] <- 0
die2[which(die2 > 6)] <- 6

# tables of proportions in each category for both dice
table(die1) / sum(table(die1))
```

```
die1
```

```
   1    2    3    4    5    6
0.16 0.16 0.20 0.08 0.16 0.24
```

```
table(die2) / sum(table(die2))
```

```
die2
   1    2    3    4    5    6
0.08 0.04 0.16 0.20 0.24 0.28
```

```
# check whether difference in means is detectable or not
t.test(die1, die2)
```

```
    Welch Two Sample t-test

data:  die1 and die2
t = -1.4118, df = 46.578, p-value = 0.1647
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.6492142  0.2892142
sample estimates:
mean of x mean of y
     3.64      4.32
```

Assuming that higher rolls of the die are better, the loaded die gives us better results than the fair die. The fair die has a mean of 3.64 and the loaded die as mean of 4.32. We cannot reject the null hypothesis that there is no difference between the fair die and the loaded die. The p value is 0.16 which is larger than our default alpha level of 0.05.

5.2.0.2 Exercise 2

Re-create the dice but increase the number of observations to 1000. Does the result of the t test change?

```
# set random number generator
set.seed(123456)

# fair die
die1 <- as.integer(runif(1000, min = 1, max = 7))
# loaded die

die2 <- as.integer(rnorm(1000, mean = 5, sd = 1.5))
die2[which(die2 < 0)] <- 0
die2[which(die2 > 6)] <- 6

# tables of proportions in each category for both dice
table(die1) / sum(table(die1))
```

```
die1
    1     2     3     4     5     6
0.172 0.155 0.168 0.176 0.161 0.168
```

```
table(die2) / sum(table(die2))
```

```
die2
    0     1     2     3     4     5     6
0.006 0.016 0.071 0.164 0.239 0.252 0.252
```

```r
# check whether difference in means is detectable or not
t.test(die1, die2)
```

```
    Welch Two Sample t-test

data:  die1 and die2
t = -12.72, df = 1892.4, p-value < 0.00000000000000022
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.009909 -0.740091
sample estimates:
mean of x mean of y
    3.503     4.378
```

The difference in means is clearly detected now. The p value is extremely small. Hence, we can reject the null hypothesis that there is no difference in means.

The difference in this exercise and the previous one is the sample size. When we increase the sample size, our standard error decreases. Therefore, we can detect a smaller effects (differences). The larger the sample size, the easier it is to detect significant differences. If the sample size is very very large, everything becomes significant (we can detect even minuscule differences).

5.2.0.3 Exercise 3

Ordinary Economic Voting Behavior in the Extraordinary Election of Adolf Hitler Download and then load who_voted_nazi_in_1932.csv.

```r
df <- read.csv("who_voted_nazi_in_1932.csv")
```

5.2.0.4 Exercise 4

Estimate the means and illustrate the distribution of potential voter shares by class.

```r
# self-employed
mean(df$shareself)
```

```
[1] 18.58871
```

```r
# blue-collar
mean(df$shareblue)
```

```
[1] 30.82347
```

```r
# white-collar
mean(df$sharewhite)
```

```
[1] 11.42254
```

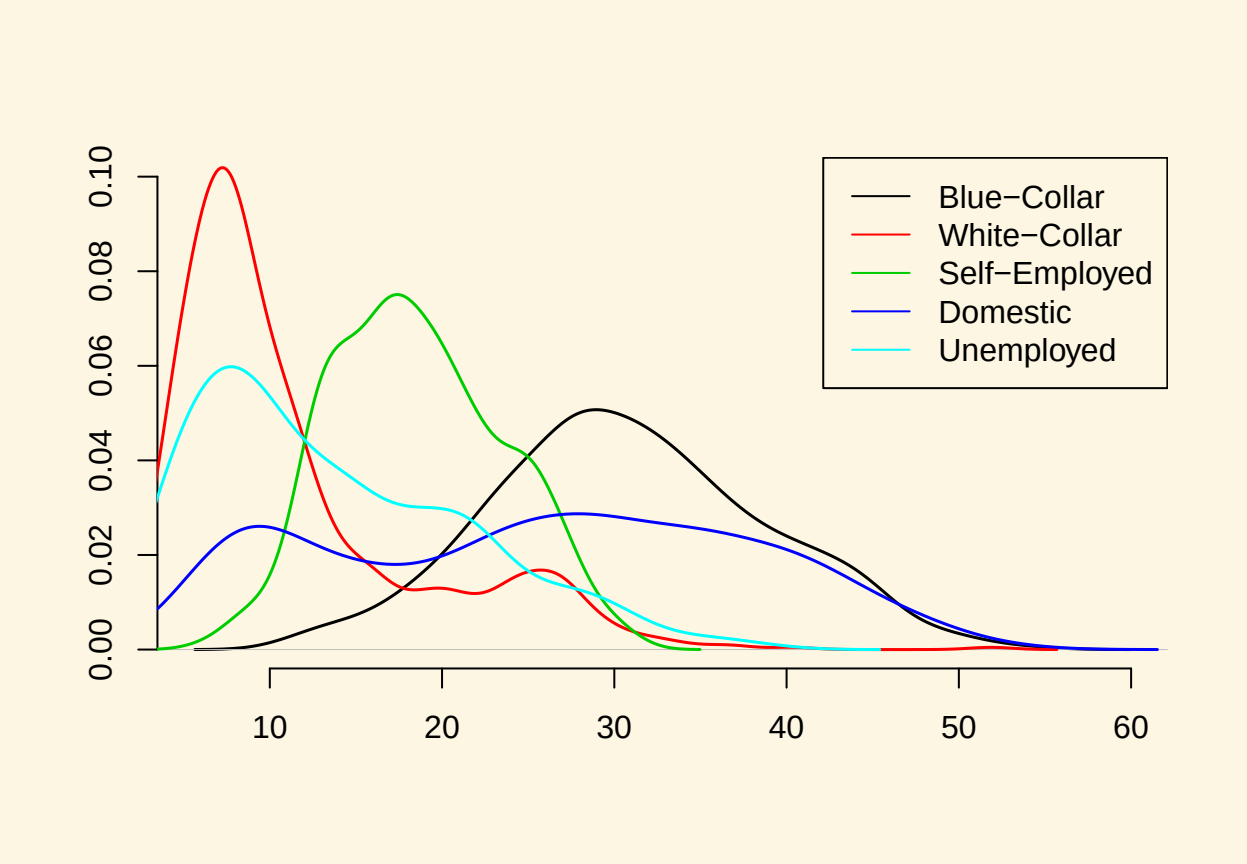
```r
# domestically employed
mean(df$sharedomestic)
```

```
[1] 25.44283
```

```r
# unemployed
mean(df$shareunemployed)
```

```
[1] 13.72245
```

```r
# illustrate distributions
plot(density(df$shareblue),
     main ="",
     xlab="",
     ylab="",
     ylim = c(0, 0.1),
     bty = "n",
     lwd = 1.5)
lines(density(df$sharewhite), col = 2, lwd = 1.5)
lines(density(df$shareself), col = 3, lwd = 1.5)
lines(density(df$sharedomestic), col = 4, lwd = 1.5)
lines(density(df$shareunemployed), col = 5, lwd = 1.5)
legend("topright", col = c(1,2,3,4,5), lty = "solid",
       c("Blue-Collar", "White-Collar", "Self-Employed",
         "Domestic", "Unemployed"))
```



5.2.0.5 Exercise 5

Estimate the mean vote shares for the Nazis in districts where the share of blue-collar voters was above the mean (30.82) and below it.

```r
# many blue-collar workers
share.in.blue.high <- mean(df$sharenazis[ df$shareblue > mean(df$shareblue) ])
share.in.blue.high
```

```
[1] 41.97132
```

```r
# fewer blue-collar workers
share.in.blue.low <- mean(df$sharenazis[ df$shareblue < mean(df$shareblue) ])
share.in.blue.low
```

```
[1] 41.19673
```

5.2.0.6 Exercise 6

Construct confidence intervals around the means.

```r
# ci blue-collar workers high
se.blue.high <- sd(df$sharenazis[ df$shareblue > mean(df$shareblue) ]) /
  sqrt( length(df$sharenazis[ df$shareblue > mean(df$shareblue) ]) )

# lower bound
share.in.blue.high - 1.96 * se.blue.high
```

```
[1] 40.90744
```

```r
# upper bound
share.in.blue.high + 1.96 * se.blue.high
```

```
[1] 43.03521
```

```r
# ci blue-collar workers low
se.blue.high <- sd(df$sharenazis[ df$shareblue < mean(df$shareblue) ]) /
  sqrt( length(df$sharenazis[ df$shareblue < mean(df$shareblue) ]) )

# lower bound
share.in.blue.high - 1.96 * se.blue.high
```

```
[1] 40.75962
```

```r
# upper bound
share.in.blue.high + 1.96 * se.blue.high
```

```
[1] 43.18303
```

5.2.0.7 Exercise 7

Are there differences between the groups? Use the appropriate statistical test to find out.

```r
# t-test for difference in means
t.test(df$sharenazis[ df$shareblue > mean(df$shareblue) ],
       df$sharenazis[ df$shareblue < mean(df$shareblue) ])
```

```
    Welch Two Sample t-test

data:  df$sharenazis[df$shareblue > mean(df$shareblue)] and df$sharenazis[df$shareblue < mean(df$shareb
t = 0.94153, df = 673.93, p-value = 0.3468
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.8407583  2.3899379
sample estimates:
mean of x mean of y
 41.97132  41.19673
```

We cannot reject the null hypothesis that there is no difference in means.

5.2.0.8 Exercise 8

Calculate t values and p values on your own (without using the t test)

```
# standard error of the difference in means
se.fd <- sqrt(  (var(df$sharenazis[ df$shareblue > mean(df$shareblue)]) /
                   length(df$sharenazis[ df$shareblue > mean(df$shareblue)])) +
                  (var(df$sharenazis[ df$shareblue < mean(df$shareblue)]) /
                     length(df$sharenazis[ df$shareblue < mean(df$shareblue)])))

# t value
t.val <- (share.in.blue.high - share.in.blue.low) / se.fd

# p value
(1- pnorm(t.val))*2
```

```
[1] 0.3464331
```

5.2.0.9 Exercise 9

Interpret your results substantially.

A common hypothesis is that it was blue-collar workers who voted en-masse for Hitler. However, when comparing districts where the share of blue-collar workers is above the mean to districts where the share is below the mean, we do not see any difference in the vote share of Nazis.

Based on this comparison, we would not conclude that a high share of blue-collar workers made the difference between a good and a bad result for the National Socialist Party.

5.2.0.10 Exercise 10

Estimate the mean vote shares for the Nazis in districts where the share of white-collar voters was above the mean (11.423) and below it.

```
# clear workspace
rm(list=ls())

# re-load data
df <- read.csv("who_voted_nazi_in_1932.csv")

# vector nazi vote share in places where white-collar workers was above the mean
n.share.high.wc <- df$sharenazis[ df$sharewhite > mean(df$sharewhite) ]

# vector nazi vote share in places where white-collar workers was below the mean
n.share.low.wc <- df$sharenazis[ df$sharewhite < mean(df$sharewhite) ]

# high white-collar group mean
mean(n.share.high.wc)
```

```
[1] 37.83325
```

```
# low white-collar group mean
mean(n.share.low.wc)
```

```
[1] 43.38576
```

5.2.0.11 Exercise 11

Construct confidence intervals around the means.

We do this first for the group with a share of white-collar workers.

```
# number of districts with high white-collar share
num.high.wc <- length(n.share.high.wc)

# standard error for high group
se.high.wc <- sd(n.share.high.wc) / sqrt(num.high.wc)

# lower bound
mean(n.share.high.wc) - 1.96 * se.high.wc
```

```
[1] 36.81174
```

```
# upper bound
mean(n.share.high.wc) + 1.96 * se.high.wc
```

```
[1] 38.85476
```

Now, we construct the confidence interval around the mean for the group with a low share of white-collar workers.

```
# number of districts with low white-collar share
num.low.wc <- length(n.share.low.wc)

# standard error for low group
se.low.wc <- sd(n.share.low.wc) / sqrt(num.low.wc)

# lower bound
mean(n.share.low.wc) - 1.96 * se.low.wc
```

```
[1] 42.32525
```

```
# upper bound
mean(n.share.low.wc) + 1.96 * se.low.wc
```

```
[1] 44.44627
```

5.2.0.12 Exercise 12

Are there differences between the groups? Use the appropriate statistical test to find out.

```
t.test(n.share.high.wc, n.share.low.wc)
```

```

    Welch Two Sample t-test

data:  n.share.high.wc and n.share.low.wc
t = -7.3909, df = 612.69, p-value = 0.0000000000004802
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -7.027862 -4.077152
sample estimates:
mean of x mean of y
 37.83325  43.38576
```

The t test shows that the difference in means is statistically significant. In districts with a high share of white-collar workers the share for the Nazis is lower.

5.2.0.13 Exercise 13

Calculate t values and p values on your own (without using the t test)

```
# variance in high white-collar group
var.high.wc <- var(n.share.high.wc)

# variance in low white-collar group
var.low.wc <- var(n.share.low.wc)


# standard error of the difference in means
se.fd <- sqrt( ((var.high.wc/num.high.wc) + (var.low.wc/num.low.wc))   )

# t value
t.val <- (mean(n.share.high.wc) - mean(n.share.low.wc)) / se.fd

# p value
pnorm(t.val) * 2
```

```
[1] 0.0000000000001457993
```

5.2.0.14 Exercise 14

Interpret your results substantially.

We reject the null hypothesis that there is no difference between districts with a low share of white-collar workers and districts with a high share of white-collar workers. In districts where the share of white-collar workers was high, the share for the Nazis was 5.6 percentage points lower.

6 Bivariate linear regression models

6.1 Seminar

```
rm(list = ls())
```

6.1.1 Packages

We will need to install a package in this week's seminar. Packages are like apps for your phone. R comes with some core functionality and allows users to add functionality. These add-ons are called packages. We first need to install a package (but only once). Every time we start R, we need to load the package.

To install a package, we write `install.packages("package.name")`. To load a package, we write `library(package.name)`.

This week's package is called `texreg` and it makes it easy to produce publication quality output from our regression models. We'll discuss this package in more detail as we go along. For now let's install the package and then load the package.

```
install.packages("texreg") # install only once
library(texreg) # load in the beginning of every R session
```

We will use a dataset collected by the US census bureau that contains several socioeconomic indicators. You can load the dataset directly from the internet.

```
dat <- read.csv("https://raw.githubusercontent.com/philippbroniecki/statistics1/master/data/communities
```

We will be exploring the relationship between the unemployment rate and low education. The variable names for these variables are not terribly clear and so first we will rename these variables using the `names()` function and the `which()` function from last week.

```
names(dat)
```

```
 [1] "state"           "county"          "community"
 [4] "communityname"   "fold"            "population"
 [7] "householdsize"   "racepctblack"    "racePctWhite"
[10] "racePctAsian"    "racePctHisp"     "agePct12t21"
[13] "agePct12t29"     "agePct16t24"     "agePct65up"
[16] "numbUrban"       "pctUrban"        "medIncome"
[19] "pctWWage"        "pctWFarmSelf"    "pctWInvInc"
[22] "pctWSocSec"      "pctWPubAsst"     "pctWRetire"
[25] "medFamInc"       "perCapInc"       "whitePerCap"
[28] "blackPerCap"     "indianPerCap"    "AsianPerCap"
[31] "OtherPerCap"     "HispPerCap"      "NumUnderPov"
[34] "PctPopUnderPov"  "PctLess9thGrade" "PctNotHSGrad"
[37] "PctBSorMore"     "PctUnemployed"
```

```
names(dat)[which(names(dat) == "PctUnemployed")] <- "UnemploymentRate"
names(dat)[which(names(dat) == "PctNotHSGrad")] <- "NoHighSchool"
```

The first variable (`UnemploymentRate`) measures the proportion of citizens in each community who are unemployed. The second variable (`NoHighSchool`) measures the proportion of citizens in each community who failed to finish high-school.

If we summarize these variables with the `summary()` function, we will see that they are both measured as proportions (they vary between 0 and 1):

```
summary(dat$UnemploymentRate)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.2200  0.3200  0.3635  0.4800  1.0000
```

```
summary(dat$NoHighSchool)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.2300  0.3600  0.3833  0.5100  1.0000
```

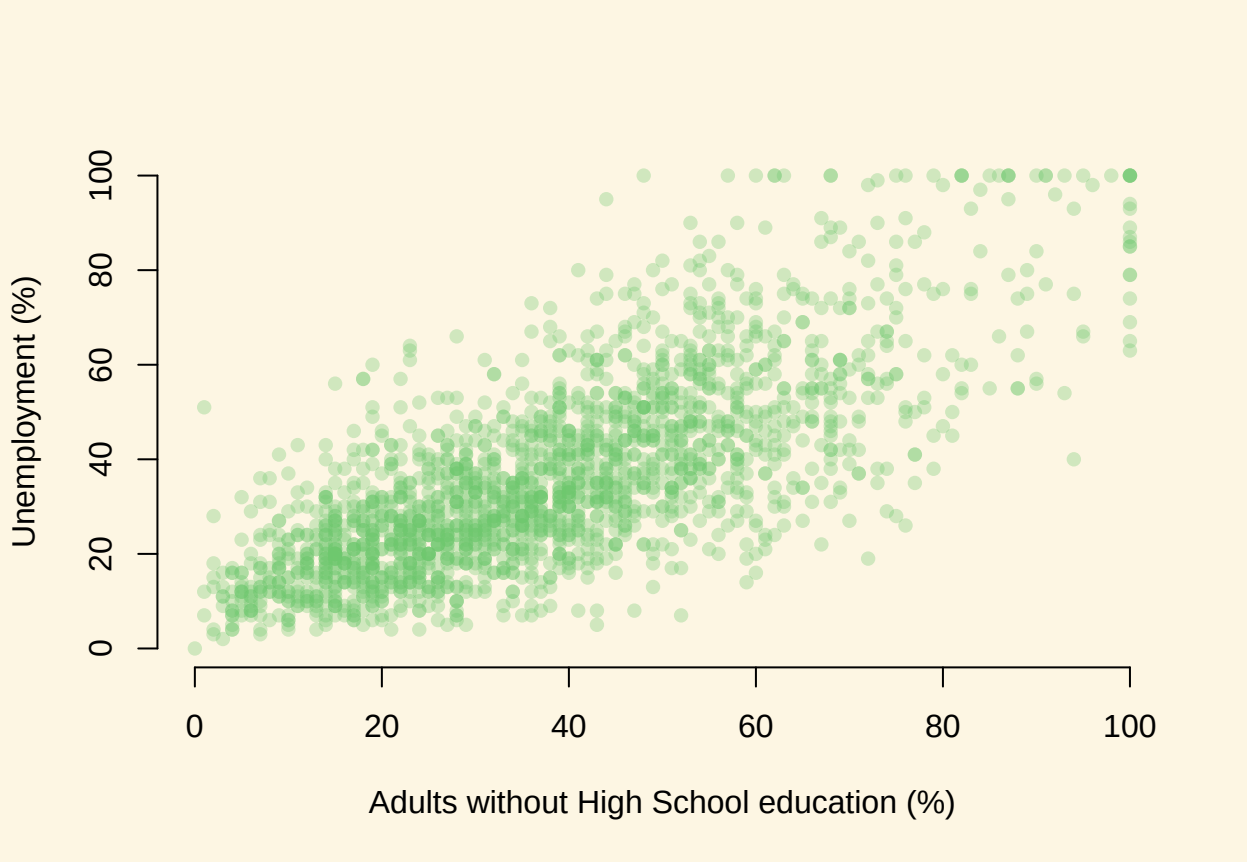
It will be a little easier to interpret the regression output if we convert these to percentages rather than proportions. We can do this with the following lines of code:

```
dat$UnemploymentRate <- dat$UnemploymentRate*100
dat$NoHighSchool <- dat$NoHighSchool*100
```

We can begin by drawing a scatterplot with the percentage of unemployed people on the y-axis and the percentage of adults without high-school education on the x-axis.

```
plot(
  y = dat$UnemploymentRate,
  x = dat$NoHighSchool,
```

```
  xlab = "Adults without High School education (%)",
  ylab = "Unemployment (%)",
  bty = "n",
  pch = 16,
  col = rgb(red = 110, green = 200, blue = 110, alpha = 80, maxColorValue = 255)
)
```



From looking at the plot, what is the association between the unemployment rate and lack of high-school level education?

In order to answer that question empirically, we will run a linear regression using the `lm()` function in R. The `lm()` function needs to know a) the relationship we're trying to model and b) the dataset for our observations. The two arguments we need to provide to the `lm()` function are described below.

Argument	Description
`formula`	The `formula` describes the relationship between the dependent and independent variables, for example `dependent.variable ~ independent.variable` In our case, we'd like to model the relationship using the formula: `UnemploymentRate ~ NoHighSchool`
`data`	This is simply the name of the dataset that contains the variable of interest. In our case, this is the merged dataset called `communities`.

For more information on how the `lm()` function works, type help(lm) in R.

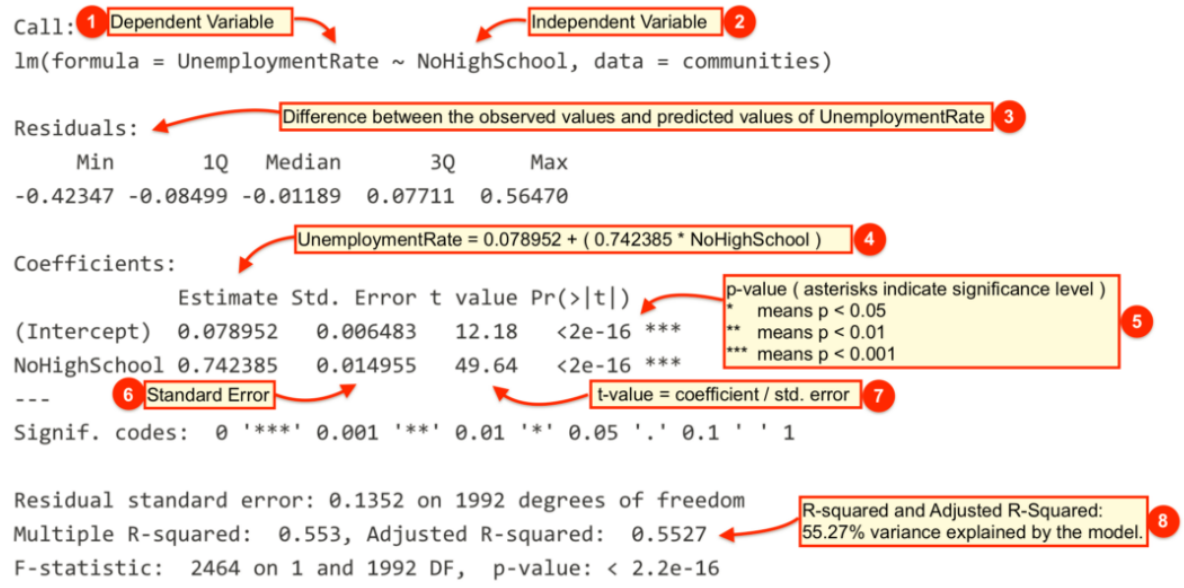


Figure 6:

```
model1 <- lm(UnemploymentRate ~ NoHighSchool, data = dat)
```

The `lm()` function has modeled the relationship between `PctUnemployed` and `NoHighSchool` and we've saved it in an object called `model1`. Let's use the `summary()` function to see what this linear model looks like.

```
summary(model1)
```

```
Call:
lm(formula = UnemploymentRate ~ NoHighSchool, data = dat)

Residuals:
    Min      1Q  Median      3Q     Max
-42.347  -8.499  -1.189   7.711  56.470

Coefficients:
             Estimate Std. Error t value            Pr(>|t|)
(Intercept)   7.89520    0.64833   12.18 <0.0000000000000002 ***
NoHighSchool  0.74239    0.01496   49.64 <0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 13.52 on 1992 degrees of freedom
Multiple R-squared:  0.553, Adjusted R-squared:  0.5527
F-statistic:  2464 on 1 and 1992 DF,  p-value: < 0.00000000000000022
```

6.1.1.1 Interpreting Regression Output

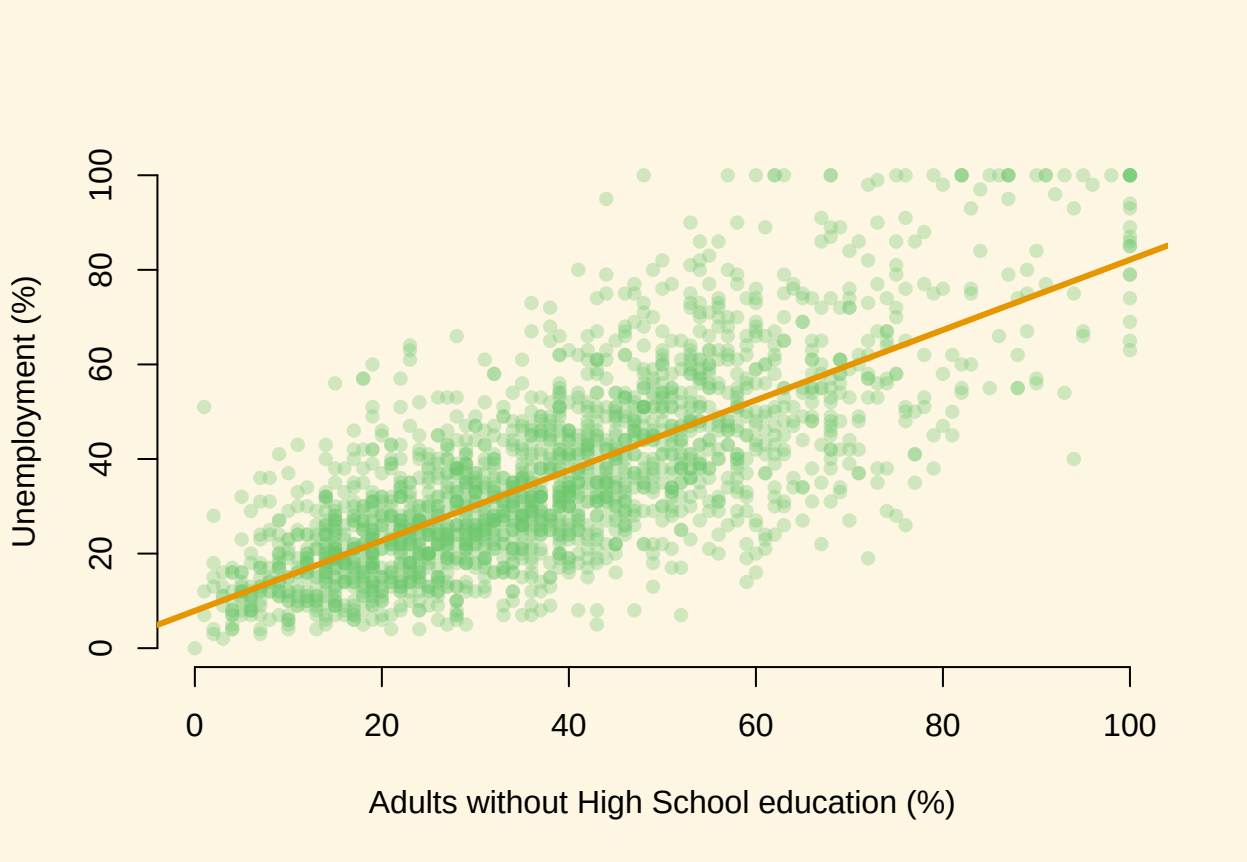
The output from `lm()` might seem overwhelming at first so let's break it down one item at a time.

#	Description
1	The *dependent* variable, also sometimes called the outcome variable. We are trying to model the effects of `NoHighSchool` on `UnemploymentRate` so `UnemploymentRate` is the *dependent* variable.
2	The *independent* variable or the predictor variable. In our example, `NoHighSchool` is the *independent* variable.
3	The differences between the observed values and the predicted values are called *residuals*. R produces a summary of the residuals.
4	The *coefficients* for the intercept and the *independent* variables. Using the *coefficients* we can write down the relationship between the *dependent* and the *independent* variables as: `UnemploymentRate` = 7.8952023 + (0.7423853 * `NoHighSchool`) This tells us that for each unit increase in the variable `NoHighSchool`, the `UnemploymentRate` increases by 0.7423853.
5	The *p-value* for each of the coefficients in the model. Recall that according to the null hypotheses, the value of the coefficient of interest is zero. The *p-value* tells us whether can can reject the null hypotheses or not.
6	The *standard error* estimates the standard deviation of the sampling distribution of the coefficients in our model. We can think of the *standard error* as the measure of precision for the estimated coefficients.
7	The *t statistic* is obtained by dividing the *coefficients* by the *standard error*.
8	The *R-squared* and *adjusted R-squared* tell us how much of the variance in our model is accounted for by the *independent* variable. The *adjusted R-squared* is always smaller than *R-squared* as it takes into account the number of *independent* variables and degrees of freedom.

Now let's add a regression line to the scatter plot using the `abline()` function.

```
## First we run the same "plot" function as before
plot(
  UnemploymentRate ~ NoHighSchool, data = dat,
  xlab = "Adults without High School education (%)",
  ylab = "Unemployment (%)",
  frame.plot = FALSE,
  pch = 16,
```

```
  col = rgb(red = 110, green = 200, blue = 110, alpha = 80, maxColorValue = 255)
)

## Then we use the "abline" function to plot the regression line from our saved model object
abline(model1, lwd = 3,
       col = rgb(red = 230, green = 150, blue = 0, alpha = 255, maxColorValue = 255))
```



We can see by looking at the regression line that it matches the coefficients we estimated above. For example, when `NoHighSchool` is equal to zero (i.e. where the line intersects the Y-axis), the predicted value for `UnemploymentRate` seems to be above 0 but below 10. This is good, as the *intercept* coefficient we estimated in the regression was `7.895`.

Similarly, the coefficient for the variable `NoHighSchool` was estimated to be `0.74239`, which implies that a one point increase in the percentage of citizens with no high-school education is associated with about .74 of a point increase in the percentage of citizens who are unemployed. The line in the plot seems to reflect this: it is upward sloping, so that higher levels of the no high-school variable are associated with higher levels of unemployment, but the relationship is not quite 1-to-1. That is, for each additional percentage point of citzens without high school education, the percentage of citizens who are unemployed increases by a little less than one point.

While the `summary()` function provides a slew of information about a fitted regression model, we often need to present our findings in easy to read tables similar to what you see in journal publications. The `texreg` package we installed earlier allows us to do just that.

Let's take a look at how to display the output of a regression model on the screen using the `screenreg()` function from `texreg`.

```
screenreg(model1)
```

```
========================
              Model 1
------------------------
(Intercept)      7.90 ***
                (0.65)
NoHighSchool     0.74 ***
                (0.01)
------------------------
R^2              0.55
Adj. R^2         0.55
Num. obs.     1994
RMSE            13.52
========================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

Here, the output includes some of the most salient details we need for interpretation. We can see the coefficient for the `NoHighSchool` variable, and the estimated coefficient for the intercept. Below these numbers, in brackets, we can see the standard errors. The table also reports the R^2, the adjusted R^2, the number of observations (n) and the root-mean-squared-error (RMSE).

One thing to note is that the table does not include either t-statistics or p-values for the estimated coefficents. Instead, the table employs a common device of using stars to denote whether a variable is statistically significant at a given alpha level.

- *** indicates that the coefficient is significant at the 99.9% confidence level (alpha = 0.001)
- ** indicates that the coefficient is significant at the 99% confidence level (alpha = 0.01)
- * indicates that the coefficient is significant at the 95% confidence level (alpha = 0.05)

Returning to our example, are there other variables that might affect the unemployment rate in our dataset? For example, is the unemployment rate higher in rural areas? To answer this question, we can swap `NoHighSchool` for a different independent variable. Let's use the variable `population`, which measures the proportion of adults who live in cities (rather than rural areas). Again, we can transform this proportion to a percentage with the following code:

```
dat$population <- dat$population*100
```

Let's fit a linear model using `population` as the independent variable:

```
model2 <- lm(UnemploymentRate ~ population, data = dat)
summary(model2)
```

```
Call:
lm(formula = UnemploymentRate ~ population, data = dat)

Residuals:
    Min      1Q  Median      3Q     Max
-35.252 -14.715  -3.946  11.054  64.980

Coefficients:
            Estimate Std. Error t value             Pr(>|t|)
(Intercept) 35.02042    0.49206  71.171 < 0.0000000000000002 ***
population   0.23139    0.03532   6.552       0.000000000072 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20.01 on 1992 degrees of freedom
Multiple R-squared:  0.0211,    Adjusted R-squared:  0.02061
F-statistic: 42.93 on 1 and 1992 DF,  p-value: 0.00000000007201
```

We can show regression line from the `model2` just like we did with our first model.

```
plot(
  UnemploymentRate ~ population, data = dat,
  xlab = "Adults living in cities (%)",
  ylab = "unemployment (%)",
  frame.plot = FALSE,
  pch = 16,
  col = rgb(red = 110, green = 200, blue = 110, alpha = 100, maxColorValue = 255)
  )
abline(model2, lwd = 2,
       col = rgb(red = 230, green = 150, blue = 0, alpha = 255, maxColorValue = 255))
```

So we now have two models! Often, we will want to compare two estimated models side-by-side. We might want to say how the coefficients for the independent variables we included differ in `model1` and `model2`, for example. Or we may want to ask: Does `model2` offer a better fit than `model1`?

It is often useful to print the salient details from the estimated models side-by-side. We can do this by using the `screenreg()` function.

```
screenreg(list(model1, model2))
```

```
=====================================
               Model 1      Model 2
-------------------------------------
(Intercept)      7.90 ***    35.02 ***
                (0.65)       (0.49)
NoHighSchool     0.74 ***
                (0.01)
population                    0.23 ***
                             (0.04)
-------------------------------------
R^2              0.55         0.02
Adj. R^2         0.55         0.02
Num. obs.     1994         1994
RMSE            13.52        20.01
=====================================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

What does this table tell us?

- The first column replicates the results from our first model. We can see that a one point increase in the percentage of citizens without high-school education is associated with an increase of 0.74 percentage points of unemployment, on average.
- The second column gives us the results from the second model. Here, a one point increase in the percentage of citizens who live in cities is associated with an increase of 0.23 percentage points of unemployment, on average
- We can also compare the R^2 values from the two models. The R^2 for `model1` is 0.55 and for `model2` is 0.02. This suggests that the model with `NoHighSchool` as the explanatory variable explains about 55% of the variation in unemployment. The model with `population` as the explanatory variable, on the other hand, explains just 2% of the variation in unemployment.

Finally, and this is something that might help with your coursework, let's save the same output as a Microsoft Word document using `htmlreg()`.

```
htmlreg(list(model1, model2), file = "Regressions_on_Unemployment.doc")
```

6.1.2 Fitted values

Once we have estimated a regression model, we can use that model to produce fitted values. Fitted values represent our "best guess" for the value of our dependent variable for a specific value of our independent variable.

To calculate fitted values we use the `predict()` function. Let's say that, on the basis of `model1` we would like to know what the unemployment rate is likely to be for a community where the percentage of adults without a high-school education is equal to 10%.

The predict function takes two main arguments.

Argument	Description
`object`	The `object` is the model object that we would like to use to produce fitted values. Here, we would like to base the analysis on `model1` and so specify `object = model1` here.

Argument	Description
`newdata`	This is an optional argument which we use to specify the values of our independent variable(s) that we would like fitted values for. If we leave this argument empty, R will automatically calculate fitted values for all of the observations in the data that we used to estimate the original model. If we include this argument, we need to provide a `data.frame` which has a variable with the same name as the independent variable in our model. Here, we specify `newdata = data.frame(NoHighSchool = 10)`, as we would like the fitted value for a community where 10% of adults did not complete high-school.

```
predict(model1, newdata = data.frame(NoHighSchool = 10))
```

```
       1
15.31906
```

Note that in this simple case, we can calculate the fitted value manually. The fitted value formula is:

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 * X_i$$

So, we can substitute in the relevant coefficients from `model1` and the number 10 for our X variable (as we want a fitted value for when X is equal to 10), and we get:

$$\hat{Y}_i = 7.9 + 0.74 * 10 = 15.3$$

which is the same as the result we obtained from the `predict()` function! The good thing about the `predict()` function, however, is that we will be able to use it for *all* the models we study on this course, and it can be useful for calculating many different fitted values. This will save a lot of time which might be wasted doing the calculations by hand.

6.1.3 Additional Resources

- Linear Regression - Interactive App

6.1.4 Exercises

1. Open a new script and save it as assignment6.
2. Clear your workspace.
3. Load the non-western foreigners dataset from week 2.
4. Estimate a model that explains subjective number of immigrants per 100 British citizens using only one independent variable. Justify your choice. (You do not have to pick the best variable but try to make a reasonable argument why more of x should lead to more/less of y).
5. Plot a scatterplot of the relationship and add the regression line to the plot.
6. Interpret the regression output and try to imagine that you are communicating your results to someone who does not know anything about statistics.
7. Estimate another model (i.e. choose a different independent variable) on the same dependent variable. Justify the choice.
8. Interpret the new regression output.
9. Compare the two models and explain which one you would choose.
10. Produce a table with both models next to each other in some text document. You can use `texreg` from the seminar, do it manually, or use something else.

11. Consider the following table. This analysis asks whether individuals who have spent longer in education have higher yearly earnings. The analysis is based on a sample of 300 individuals. The dependent variable in this analysis is the yearly income of the individual in UK pounds (`earnings`). The independent variable measures the number of years the individual spent in full-time education (`education`).

- Interpret the coefficient on the `education` variable.
- Using the values given in the table, calculate the test-statistic
- Can we reject the null hypothesis of no effect at the 95% confidence level? (Just looking at the stars is not sufficient here! How can we work out the result of the hypothesis test?)

```
=========================
             Model 1
-------------------------
(Intercept)   3663.85
             (2326.99)
education     1270.81 ***
              (160.97)
-------------------------
R^2              0.17
Adj. R^2         0.17
Num. obs.      300
RMSE         14018.52
=========================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

12. Save the script that includes all previous tasks.
13. Source your script, i.e. run the entire script all at once without error message.

6.2 Solutions

6.2.0.1 Exercise 3

Load the non-western foreigners dataset from week 2.

```
load("BSAS_manip.RData")
```

6.2.0.2 Exercise 4

Estimate a model that explains subjective number of immigrants per 100 British citizens using only one independent variable. Justify your choice. (You do not have to pick the best variable but try to make a reasonable argument why more of x should lead to more/less of y).

```
m1 <- lm(IMMBRIT ~ RAge, data = data2)
```

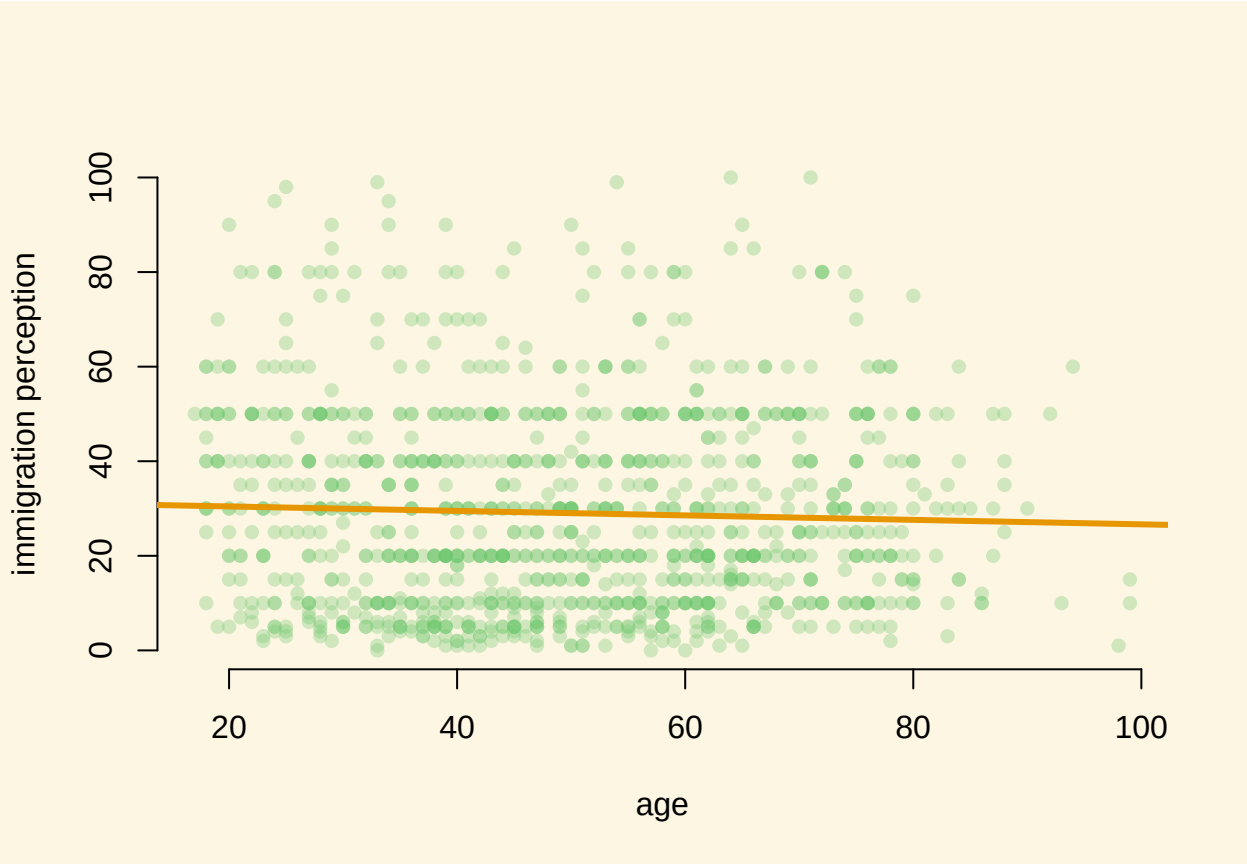
We use age as predictor variable. We argue that older people are less positive towards immigration and tend to overestimate the number of immigrants to overemphasize the problem. We, therefore, expect a positive relationship between age and the perception of immigration.

6.2.0.3 Exercise 5

Plot a scatterplot of the relationship and add the regression line to the plot.

```
plot(
  y = data2$IMMBRIT,
  x = data2$RAge,
```

```
  xlab = "age",
  ylab = "immigration perception",
  frame.plot = FALSE,
  pch = 16,
  col = rgb(red = 110, green = 200, blue = 110, alpha = 80, maxColorValue = 255)
)
abline(m1, lwd = 3,
       col = rgb(red = 230, green = 150, blue = 0, alpha = 255, maxColorValue = 255))
```



The regression line slopes downward, ever so slightly, pointing towards a tiny negative relationship. The residuals seem to be extraordinarily large. It seems, that the relation between age and immigration perception is weak at best. This is more consistent with no relationship.

6.2.0.4 Exercise 6

Interpret the regression output and try to imagine that you are communicating your results to someone who does not know anything about statistics.

```
library(texreg)
screenreg(m1)
```

```
========================
             Model 1
------------------------
(Intercept)    31.38 ***
               (1.95)
```

```
RAge           -0.05
               (0.04)
-----------------------
R^2             0.00
Adj. R^2        0.00
Num. obs.    1049
RMSE           21.06
=======================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

We cannot, with sufficient confidence, rule out that age and immigration perception are unrelated ($p > 0.05$).

As we will learn next week, R^2 indicates, that our model does a terrible job at predicting the perception of immigration. We could predict the outcome (perception of immigration) equally well without our model.

Suppose, our best guess of `IMMBRIT` for any age is just the mean of `IMMBRIT`. The quality of that prediction (in statistics jargon, the naive guess) would be as good as the predictions we get from our model.

6.2.0.5 Exercise 7

Estimate another model (i.e. choose a different independent variable) on the same dependent variable. Justify the choice.

```
m2 <- lm(IMMBRIT ~ HHInc, data = data2)
```

We choose income as the predictor in our second model. We conjecture that on average, wealthier people are more educated and hence have a more realistic view of immigration. Furthermore, they tend to face less competition from immigration and, therefore, tend not to exaggerate the level of immigration. We expect that the wealthier the respondent, the lower the respondent's estimate of immigration.

6.2.0.6 Exercise 8

Interpret the new regression output.

```
screenreg(m2)
```

```
========================
             Model 1
------------------------
(Intercept)    43.12 ***
               (1.41)
HHInc          -1.47 ***
               (0.13)
------------------------
R^2             0.10
Adj. R^2        0.10
Num. obs.    1049
RMSE           19.94
========================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

In line with our expectation, wealthier people perceive immigration to be lower than poorer people. The relationship is significant at the five percent level. We explain a tenth of the variance in the perception of immigration with our model (you will learn this next week). Considering that this model is very small (we use only one predictor variable), we do quite well at predicting the outcome.

6.2.0.7 Exercise 9

Compare the two models and explain which one you would choose.

```
screenreg( list(m1, m2))
```

```
====================================
             Model 1      Model 2
------------------------------------
(Intercept)    31.38 ***    43.12 ***
               (1.95)       (1.41)
RAge           -0.05
               (0.04)
HHInc                       -1.47 ***
                            (0.13)
------------------------------------
R^2             0.00         0.10
Adj. R^2        0.00         0.10
Num. obs.    1049         1049
RMSE           21.06        19.94
====================================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

From what we have learned so far, we would look at the coefficients in our models. In the first, Age was insignificant. In the second, income is. Therefore, the second model is better at explaining perception on immigration. We learn nothing about potential causes of overestimating immigration from model one, whereas from model two, we do.

6.2.0.8 Exercise 10

```
htmlreg( list(m1, m2), file = "regressions_on_immigration_perception.doc")
```

6.2.0.9 Exercise 11

Consider the following table. This analysis asks whether individuals who have spent longer in education have higher yearly earnings. The analysis is based on a sample of 300 individuals. The dependent variable in this analysis is the yearly income of the individual in UK pounds (`earnings`). The independent variable measures the number of years the individual spent in full-time education (`education`).

- Interpret the coefficient on the `education` variable.

For each additional year of education we expect earnings to go up by 1270.81 pounds on average.

- Using the values given in the table, calculate the test-statistic

We compute the t value using the formula:

$$\frac{\bar{Y_{HA}} - \mu_{H_0}}{\sigma_{\bar{Y_{HA}}}}$$

So, we take the alternative hypothesis (our estimate of the effect of education) minus the mean under the null hypothesis and divide the result by the standard error of our estimate. Unless stated otherwise, the null hypothesis is that there is no effect of education on income, i.e. the null is zero.

The coefficient estimate here is 1270.81 and its standard error is 160.97.

```
1270.81 / 160.97
```

```
[1] 7.894701
```

The t value is 7.89.

- Can we reject the null hypothesis of no effect at the 95% confidence level? (Just looking at the stars is not sufficient here! How can we work out the result of the hypothesis test?)

We have 300 observations in our sample and because we estimate two parameters, 298 degrees of freedom. A t distribution with 298 degrees of freedom is well approximated by the standard normal distribution. Under the normal distribution 95% are within 1.96 standard deviations from the mean. Our t value is more extreme than that. Our estimate is 7.89 standard deviations from the mean. It is unlikely to observe such an extreme value by chance (assuming the null hypothesis, there is no relation between education and income, is true). We therefore, reject the null hypothesis at 5 percent level.

7 Multiple linear regression models (I)

7.1 Seminar

```
library(foreign)
library(texreg)
```

```
rm(list = ls())
```

7.1.1 Loading, Understanding and Cleaning our Data

Today, we load the full standard (cross-sectional) dataset from the Quality of Government Institute (this is a newer version that the one we used in week 3). This is a great data source for comparativist political science research. The codebook is available from their main website. You can also find time-series and cross-section data sets on this page.

The dataset is in stata format (`.dta`). Loading it requires the `foreign` library and the `read.dta()` function which operates similar to `read.csv()`. Download the data here

Let's load the data set

```
# load dataset in Stata format from online source
world_data <- read.dta("qog_std_cs_jan15.dta")

# check the dimensions of the dataset
dim(world_data)
```

```
[1]  193 2037
```

The dataset contains many variables. We will select a subset of variables that we want to work with. We are interested in political stability. Specifically, we want to find out what predicts the level of political stability. Therefore, `political_stability` is our dependent variable (also called response variable, left-hand-side variable, explained/predicted variable). We will also select a variable that identifies each row (observation) in the dataset uniquely: `cname` which is the name of the country. Potential predictors (independent variables, right-hand-side variables, covariates) are:

1. `lp_lat_abst` is the distance to the equator which we rename into `latitude`
2. `dr_ing` is an index for the level of globalization which we rename to `globalization`
3. `ti_cpi` is Transparency International's Corruptions Perceptions Index, renamed to `institutions_quality` (larger values mean better quality institutions, i.e. less corruption)

4. `br_dem` is a factor variable stating whether the relevant country is a democracy or not (with labels `"1. Democracy"` and `"0. Dictatorship"`)

Our dependent variable:

- `wbgi_pse` which we rename into `political_stability` (larger values mean more stability)

One approach of selecting a subset of variables we're interested in, is to use the square bracket `[ ]` operator. But first, we rename the variables we care about like we did last week.

```
names(world_data)[which(names(world_data) == "cname")] <- "country"
names(world_data)[which(names(world_data) == "wbgi_pse")] <- "political_stability"
names(world_data)[which(names(world_data) == "lp_lat_abst")] <- "latitude"
names(world_data)[which(names(world_data) == "dr_ig")] <- "globalization"
names(world_data)[which(names(world_data) == "chga_demo")] <- "democracy"
names(world_data)[which(names(world_data) == "ti_cpi")] <- "institutions_quality"
```

Now, we take our subset. We will create an object named `keep` which is a vector of the variable names that we would like to keep for the analysis. We can then use this vector to subset our `world_data` object by including it within the square parentheses:

```
keep <- c("country", "political_stability", "latitude", "globalization","democracy", "institutions_quali
world_data <- world_data[, keep]
```

Let's make sure we've got everything we need

```
head(world_data)
```

```
              country political_stability  latitude globalization
1         Afghanistan          -2.5498192 0.3666667      31.46042
2             Albania          -0.1913142 0.4555556      58.32265
3             Algeria          -1.2624909 0.3111111      52.37114
4             Andorra           1.3064846 0.4700000            NA
5              Angola          -0.2163249 0.1366667      44.73296
6 Antigua and Barbuda           0.9319394 0.1892222      48.15911
        democracy institutions_quality
1 0. Dictatorship                  1.4
2    1. Democracy                  3.3
3 0. Dictatorship                  2.9
4    1. Democracy                   NA
5 0. Dictatorship                  1.9
6    1. Democracy                   NA
```

The function `summary()` lets you summarize data sets. We will look at the dataset now. When the dataset is small in the sense that you have few variables (columns) then this is a very good way to get a good overview. It gives you an idea about the level of measurement of the variables and the scale. `country`, for example, is a character variable as opposed to a number. Countries do not have any order, so the level of measurement is categorical.

If you think about the next variable, political stability, and how one could measure it you know there is an order implicit in the measurement: more or less stability. From there, what you need to know is whether the more or less is ordinal or interval scaled. Checking `political_stability` you see a range from roughly -3 to 1.5. The variable is numerical and has decimal places. This tells you that the variable is at least interval scaled. You will not see ordinally scaled variables with decimal places. Examine the summaries of the other variables and determine their level of measurement.

```
summary(world_data)
```

```
   country          political_stability    latitude      globalization
```

```
 Length:193         Min.   :-3.10637    Min.   :0.0000   Min.   :24.35
 Class :character   1st Qu.:-0.72686    1st Qu.:0.1444   1st Qu.:45.22
 Mode  :character   Median :-0.01900    Median :0.2444   Median :54.99
                    Mean   :-0.06079    Mean   :0.2865   Mean   :57.15
                    3rd Qu.: 0.78486    3rd Qu.:0.4444   3rd Qu.:68.34
                    Max.   : 1.57240    Max.   :0.7222   Max.   :92.30
                                        NA's   :12       NA's   :12
          democracy   institutions_quality
 0. Dictatorship: 74   Min.   :1.010
 1. Democracy   :118   1st Qu.:2.400
 NA's           :  1   Median :3.300
                       Mean   :3.988
                       3rd Qu.:5.100
                       Max.   :9.300
                       NA's   :12
```

The variables `latitude`, `globalization` and `inst_quality` have 12 missing values each marked as `NA`. `democracy` has 1 missing value. Missing values could cause trouble because operations including an `NA` will produce `NA` as a result (e.g.: `1 + NA = NA`). We will drop these missing values from our data set using the `is.na()` function and square brackets. The exlamation mark in front of `is.na()` means "not". So, we keep all rows that are not NA's on the variable `latitude`.

```
world_data <- world_data[ !is.na(world_data$latitude) ,]
```

Generally, we want to make sure we drop missing values only from variables that we care about. Now that you have seen how to do this, drop missings from `globalization`, `institutions_quality`, and `democracy` yourself.

```
world_data <- world_data[ !is.na(world_data$globalization) ,]
world_data <- world_data[ !is.na(world_data$institutions_quality) ,]
world_data <- world_data[ !is.na(world_data$democracy) ,]
```

```
summary(world_data)
```

```
   country          political_stability    latitude      globalization
 Length:170         Min.   :-2.67338    Min.   :0.0000   Min.   :25.46
 Class :character   1st Qu.:-0.79223    1st Qu.:0.1386   1st Qu.:46.05
 Mode  :character   Median :-0.03174    Median :0.2500   Median :55.87
                    Mean   :-0.12018    Mean   :0.2865   Mean   :57.93
                    3rd Qu.: 0.66968    3rd Qu.:0.4444   3rd Qu.:69.02
                    Max.   : 1.48047    Max.   :0.7222   Max.   :92.30
          democracy   institutions_quality
 0. Dictatorship: 67   Min.   :1.400
 1. Democracy   :103   1st Qu.:2.500
                       Median :3.300
                       Mean   :4.050
                       3rd Qu.:5.175
                       Max.   :9.300
```

Let's look at the output of `summary(world_data)` again and check the range of the variable `latitude`. It is between `0` and `1`. The codebook clarifies that the latitude of a country's capital has been divided by `90` to get a variable that ranges from `0` to `1`. This would make interpretation difficult. When interpreting the effect of such a variable a unit change (a change of `1`) covers the entire range or put differently, it is a change from a country at the equator to a country at one of the poles.

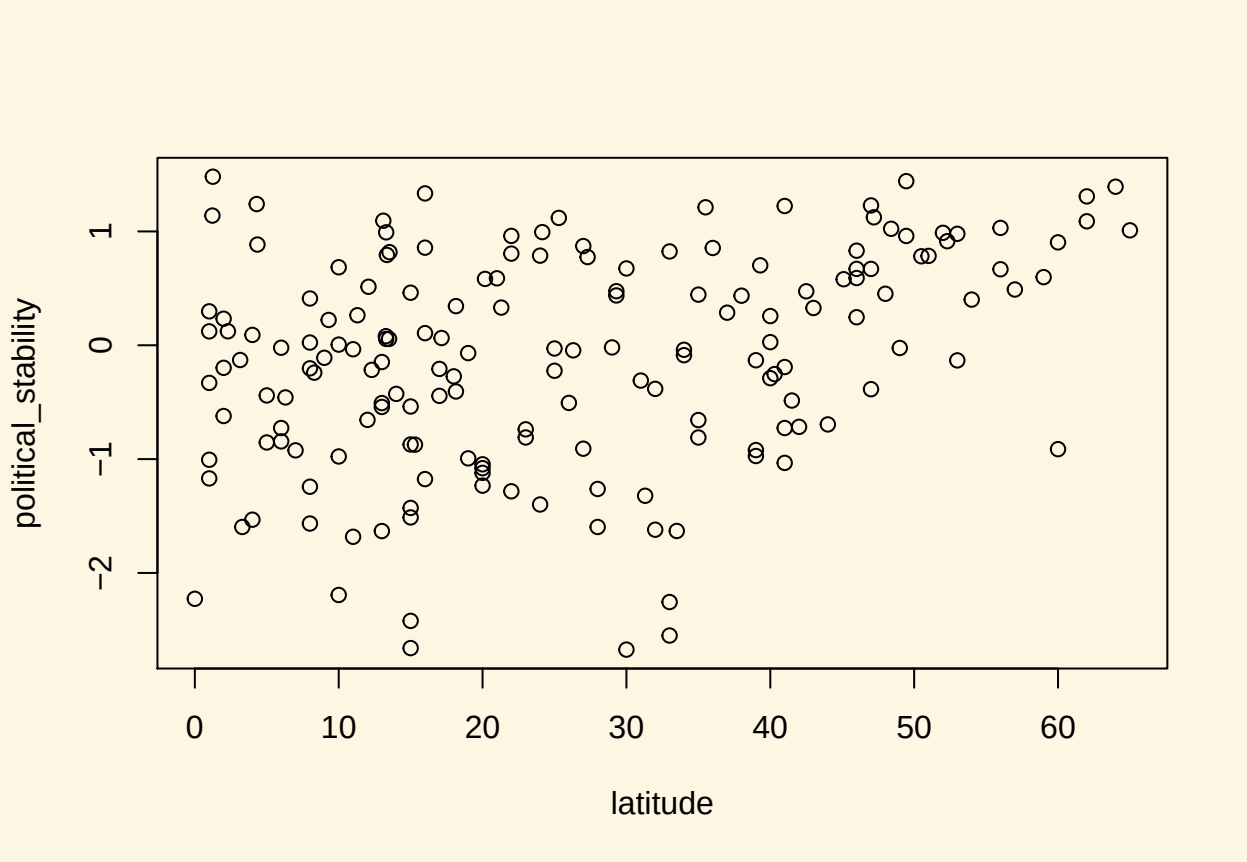
We therefore multiply by `90` again. This will turn the units of the `latitude` variable into degrees again which makes interpretation easier.

```r
# transform latitude variable
world_data$latitude <- world_data$latitude * 90
```

7.1.2 Estimating a Bivariate Regression

Is there a correlation between the distance of a country to the equator and the level of political stability? Both political stability (dependent variable) and distance to the equator (independent variable) are continuous. Therefore, we will get an idea about the relationship using a scatter plot.

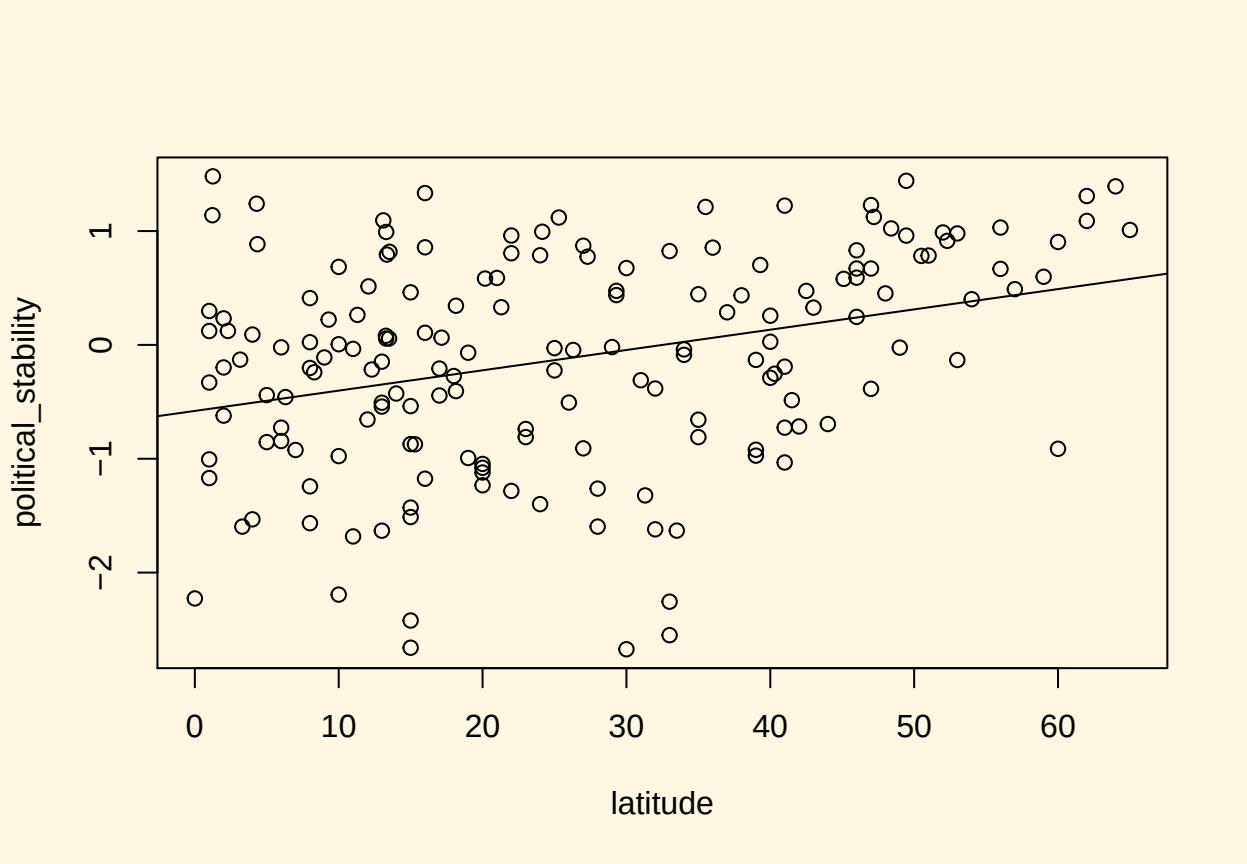
```r
plot(political_stability ~ latitude, data = world_data)
```



Looking at the cloud of points suggests that there might be a positive relationship: increases in our independent variable `latitude` appear to be associated with increases in the dependent variable `political_stability` (the further from the equator, the more stable).

We can fit a line of best fit through the points. To do this we must estimate the bivariate regression model with the `lm()` function and then plot the line using the `abline()` function.

```r
latitude_model <- lm(political_stability ~ latitude, data = world_data)

# add the line
plot(political_stability ~ latitude, data = world_data)
abline(latitude_model)
```



We can also view a simple summary of the regression by using the `screenreg` function:

```
# regression output
screenreg(latitude_model)
```

```
========================
              Model 1
------------------------
(Intercept)    -0.58 ***
               (0.12)
latitude        0.02 ***
               (0.00)
------------------------
R^2             0.11
Adj. R^2        0.10
Num. obs.     170
RMSE            0.89
========================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

Thinking back to last week, how can we interpret this regression ouput?

- The coefficient for the variable `latitude` (β_1) indicates that a one-unit increase in a country's latitude is associated with a `0.02` increase in the measure of political stability, on average. Question: Is this association statistically significant at the 95% confidence level?
- The coefficient for the `(intercept)` term (β_0) indicates that the average level of political stability for a country with a latitude of 0 is -0.58 (where `latitude = 0` is a country positioned at the equator)

- The R^2 of the model is 0.11. This implies that 11% of the variation in the dependent variable (political stability) is explained by the independent variable (latitude) in the model.

7.1.3 Multivariate Regression

The regression above suggests that there is a significant association between these variables However, as good social scientistis, we probably do not think that the distance of a country from the equator is a theoretically relevant variable for explaining political stability. This is because there is no plausible causal link between the two. We should therefore consider other variables to include in our model.

We will include the index of globalization (higher values mean more integration with the rest of the world), the quality of institutions, and the indicator for whether the country is a democracy. For all of these variables we can come up with a theoretical story for their effect on political stability.

To specify a *multiple* linear regression model, the only thing we need to change is what we pass to the `formula` argument of the `lm()` function. In particular, if we wish to add additional explanatory variables, the formula argument will take the following form:

```
dependent.variable ~ independent.variable.1 + independent.variable.2 ... independent.variable.k
```

where `k` indicates the total number of independent variables we would like to include in the model. In the example here, our model would therefore look like the following:

```
# model with more explanatory variables
inst_model <- lm(political_stability ~ latitude + globalization + institutions_quality + democracy,
                 data = world_data)
```

Remember, `political_stability` is our dependent variable, as before, and now we have four independent variables: `latitude`, `globalization`, `democracy` and `institutions_quality`. Again, just as with the bivariate model, we can view the summarised output of the regression by using `screenreg()`. As we now have two models (a simple regression model, and a multiple regression model), we can join them together using the `list()` function, and then put all of that inside `screenreg()`.

```
screenreg(list(latitude_model, inst_model))
```

```
================================================
                        Model 1     Model 2
------------------------------------------------
(Intercept)              -0.58 ***   -1.25 ***
                         (0.12)      (0.20)
latitude                  0.02 ***    0.00
                         (0.00)      (0.00)
globalization                        -0.00
                                     (0.01)
institutions_quality                  0.34 ***
                                     (0.04)
democracy1. Democracy                 0.04
                                     (0.11)
------------------------------------------------
R^2                       0.11        0.50
Adj. R^2                  0.10        0.49
Num. obs.                170         170
RMSE                      0.89        0.67
================================================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

Including the two new predictors leads to substantial changes.

- First, we now explain 50% of the variance of our dependent variable instead of just 11%.
- Second, the effect of the distance to the equator is no longer significant.
- Third, better quality institutions are associated with more political stability. In particular, a one-unit increase in the measure of instituion quality (which ranges from 1 to 10) is associated with a 0.34 increase in the measure for political stability.
- Fourth, there is no significant relationship between globalization and political stability in this data.
- Fifth, there is no significant relationship between democracy and political stability in this data.

7.1.4 Joint Significance Test (F-statistic)

Whenever you add variables to your model, you will explain more of the variance in the dependent variable. That means, using your data, your model will better predict outcomes. We would like to know whether the difference (the added explanatory power) is statistically significant. The null hypothesis is that the added explanatory power is zero and the p-value gives us the probability of observing such a difference as the one we actually computed assuming that null hypothesis (no difference) is true.

The F-test is a joint hypothesis test that lets us compute that p-value. Two conditions must be fulfilled to run an F-test:

Conditions for F-test model comparison
Both models must be estimated from the same sample! If your added variables contain lots of missing values and therefore your `n` (number of observations) are reduced substantially, you are not estimating from the same sample. The models must be nested. That means, the model with more variables must contain all of the variables that are also in the model with fewer variables.

We specify two models: a restricted model and an unrestricted model. The restricted model is the one with fewer variables. The unrestricted model is the one including the extra variables. We say restricted model because we are "restricting" it to NOT depend on the extra variables. Once we estimated those two models we compare the residual sum of squares (RSS). The RSS is the sum over the squared deviations from the regression line and that is the unexplained error. The restricted model (fewer variables) is always expected to have a larger RSS than the unrestricted model. Notice that this is same as saying: the restricted model (fewer variables) has less explanatory power.

We test whether the reduction in the RSS is statistically significant using a distribution called "F distribution". If it is, the added variables are jointly (but not necessarily individually) significant. You do not need to know how to calculate p-values from the F distribution, as we can use the `anova()` function in R to do this for us.

```
anova(latitude_model, inst_model)
```

```
Analysis of Variance Table

Model 1: political_stability ~ latitude
Model 2: political_stability ~ latitude + globalization + institutions_quality +
    democracy
  Res.Df     RSS Df Sum of Sq      F                Pr(>F)
1    168 133.121
2    165  74.229  3    58.892 43.636 < 0.00000000000000022 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As we can see from the output, the p-value here is *very* small, which means that we can reject the null hypothesis that the unrestricted model has no more explanatory power than the restricted model.

7.1.5 Predicting outcome conditional on institutional quality

Just as we did with the simple regression model last week, we can use the fitted model object to calculate the fitted values of our dependent variable for different values of our explanatory variables. To do so, we again use the `predict()` function.

We proceed in three steps.

1. We set the values of the covariates for which we would like to produce fitted values.
    - You will need to set covariate values for *every* explanatory variable that you included in your model.
    - As only one of our variables has a significant relationship with the outcome in the multiple regression model that we estimated above, we are really only interested in that variable (`institutions_quality`).
    - Therefore, we will calculate fitted values over the range of `institutions_quality`, while setting the values of `latitude` and `globalization` to their mean values.
    - As `democracy` is a factor variable, we cannot use the mean value. Instead, we will set `democracy` to be equal to `"1. Democracy"` which is the label for democratic countries
2. We calculate the fitted values.
3. We report the results (here we will produce a plot).

For step one, the following code produces a `data.frame` of new covariate values for which we would like to calculate a fitted value from our model:

```
## Set the values for the explanatory variables
data_for_fitted_values <- data.frame(institutions_quality = seq(from = 1.4, to = 9.3, by = 1),
                                     globalization = mean(world_data$globalization),
                                     latitude = mean(world_data$latitude),
                                     democracy = "1. Democracy"
                                     )
```

Here, we have set the `institutions_quality` variable to vary between 1.4 and 9.3, with increments of 1 unit. We have set `globalization` to be equal to the mean value of `globalization` in the `world_data` object, and `latitude` to be equal to the mean value of `latitude` in the `world_data` object. Finally, we have set `democracy` to be equal to `"1. Democracy"` (the value for democratic countries). We have then put all of these values into a new `data.frame` called `data_for_fitted_values` which we will pass to the `predict()` function.

Before we do that, let's just take a quick look at the `data_for_fitted_values` object:

```
head(data_for_fitted_values)
```

```
  institutions_quality globalization latitude    democracy
1                  1.4      57.93053 25.78218 1. Democracy
2                  2.4      57.93053 25.78218 1. Democracy
3                  3.4      57.93053 25.78218 1. Democracy
4                  4.4      57.93053 25.78218 1. Democracy
5                  5.4      57.93053 25.78218 1. Democracy
6                  6.4      57.93053 25.78218 1. Democracy
```

As you can see, this has produced a data.frame in which every observation has a different value of `institutions_quality` but the same value for `latitude`, `globalization`, and `democracy`.

We can now calculate the fitted values for each of these combinations of our explanatory variables by passing the `data_for_fitted_values` object to the `newdata` argument of the `predict()` function.

```
# Calculate the fitted values
fitted_values <- predict(inst_model, newdata = data_for_fitted_values)
```

```
## Save the fitted values as a new variable in the data_for_fitted_values object

data_for_fitted_values$fitted_values <- fitted_values
```

We can now look again at the `data_for_fitted_values` object:

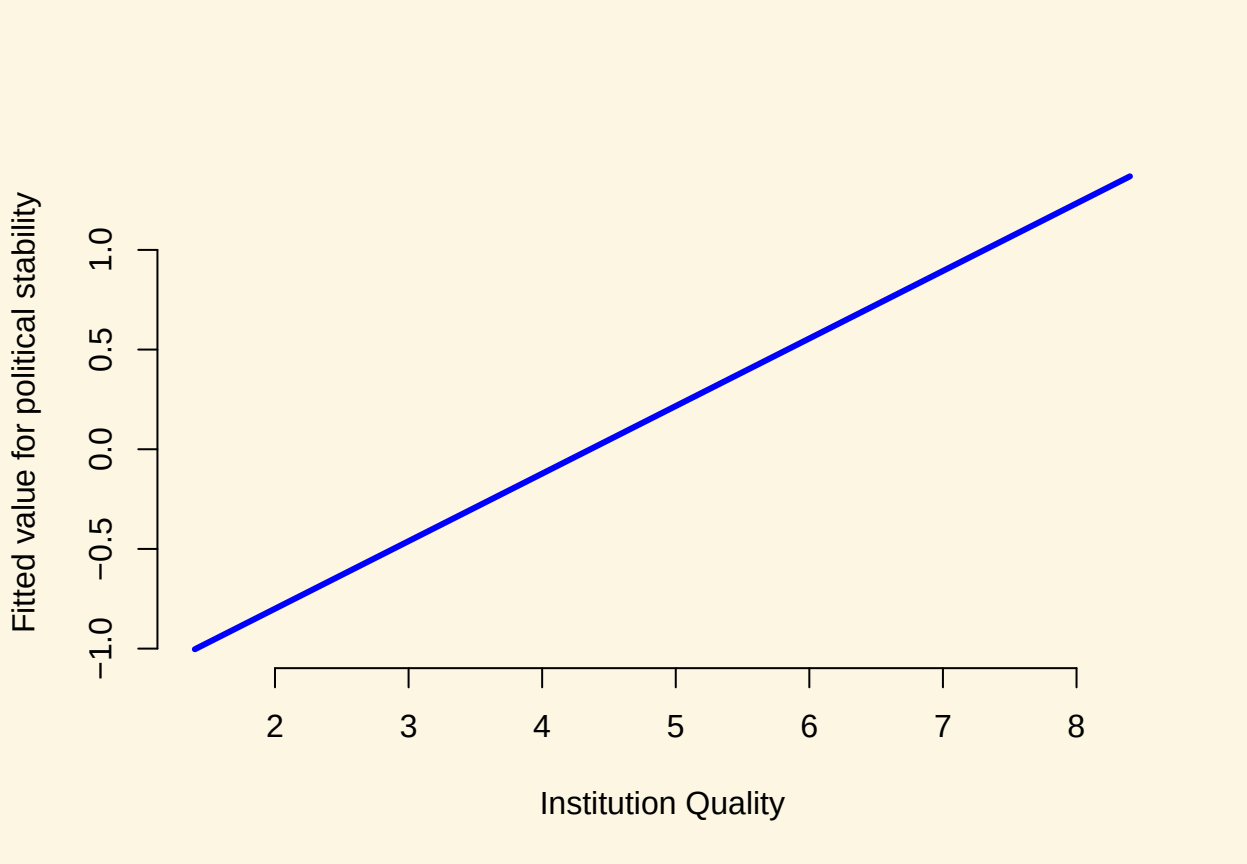
```
head(data_for_fitted_values)
```

```
  institutions_quality globalization latitude    democracy fitted_values
1                  1.4      57.93053 25.78218 1. Democracy   -1.00319887
2                  2.4      57.93053 25.78218 1. Democracy   -0.66431685
3                  3.4      57.93053 25.78218 1. Democracy   -0.32543483
4                  4.4      57.93053 25.78218 1. Democracy    0.01344719
5                  5.4      57.93053 25.78218 1. Democracy    0.35232921
6                  6.4      57.93053 25.78218 1. Democracy    0.69121123
```

Hey presto! Now, for each of our explanatory variable combinations, we have the corresponding fitted values as calculated from our estimated regression.

Finally, we can plot these values:

```
plot(
  fitted_values ~ institutions_quality, # Specify the formula for the plot (dependent.variable ~ indepe
  data = data_for_fitted_values, # Specify the data to use for the plot
  xlab = "Institution Quality", # Specify the X-axis title
  ylab = "Fitted value for political stability", # Specify the Y-axis title
  frame.plot = FALSE, # The frame.plot = FALSE argument removes the box from around the plot
  col = "blue", # The col argument specifies the color
  type = "l", # type = "l" will produce a line plot, rather than the default scatter plot
  lwd = 3 # lwd = 3 will increase the thinkness of the line on the plot
)
```



We could also use the output from our model to plot two separate lines of fitted values: one for democracies, and one for dictatorships. We have already done this for democracies, so the following code constructs a `data.frame` of fitted values for dictatorships:

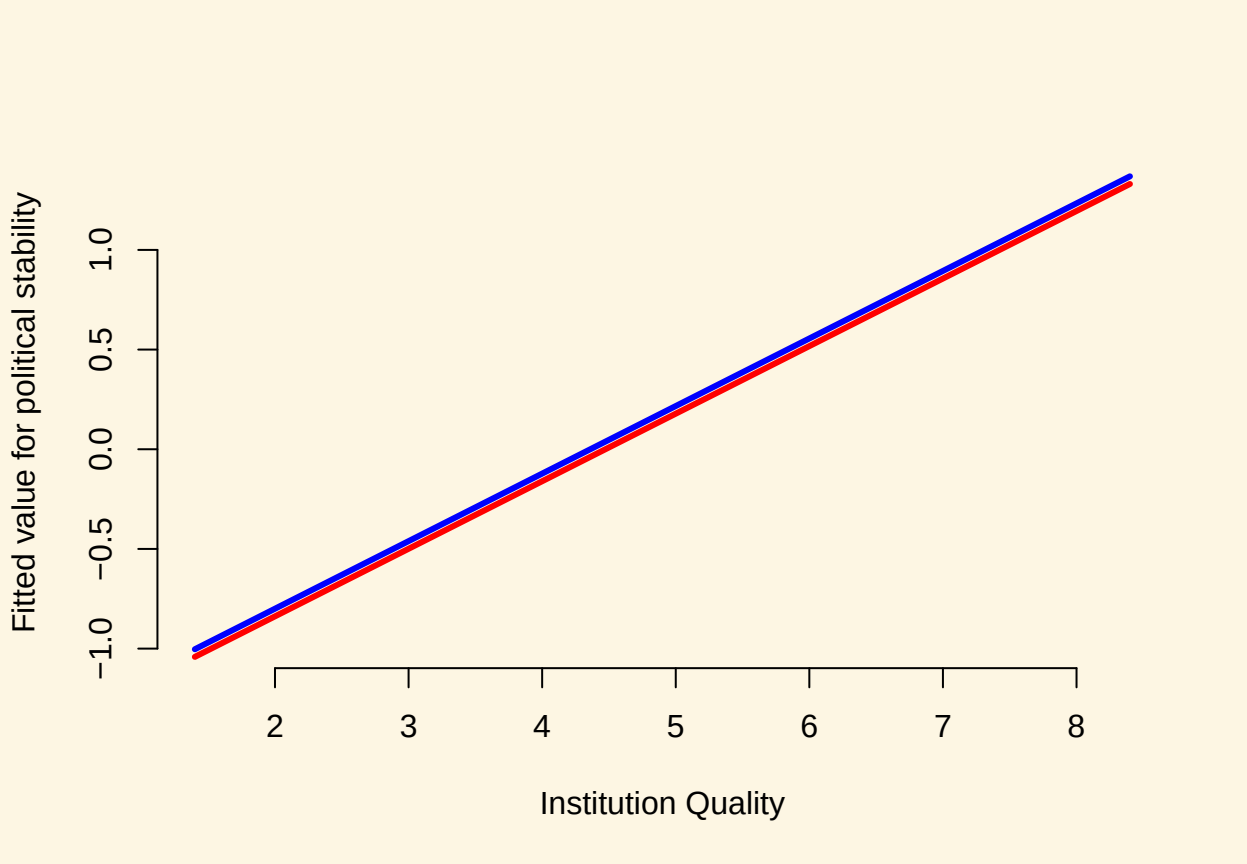
```
## Set the values for the explanatory variables
data_for_fitted_values_dictator <- data.frame(institutions_quality = seq(from = 1.4, to = 9.3, by = 1),
                                      globalization = mean(world_data$globalization),
                                      latitude = mean(world_data$latitude),
                                      democracy = "0. Dictatorship"
                                      )

# Calculate the fitted values
fitted_values_dictator <- predict(inst_model, newdata = data_for_fitted_values_dictator)

## Save the fitted values as a new variable in the data_for_fitted_values object

data_for_fitted_values_dictator$fitted_values <- fitted_values_dictator

## Take a look at the output
head(data_for_fitted_values_dictator)
```

```
  institutions_quality globalization latitude       democracy
1                  1.4      57.93053 25.78218 0. Dictatorship
2                  2.4      57.93053 25.78218 0. Dictatorship
3                  3.4      57.93053 25.78218 0. Dictatorship
4                  4.4      57.93053 25.78218 0. Dictatorship
5                  5.4      57.93053 25.78218 0. Dictatorship
```

```
6                 6.4      57.93053 25.78218 0. Dictatorship
  fitted_values
1   -1.04148517
2   -0.70260315
3   -0.36372113
4   -0.02483911
5    0.31404291
6    0.65292493
```

Now that we have calculated these fitted values, we can add the line for dictatorships to the plot we created above using the `lines()` function:

```
## Create the same plot as above for fitted values over the range of institution quality for democracies
plot(
  fitted_values ~ institutions_quality, # Specify the formula for the plot (dependent.variable ~ indepe
  data = data_for_fitted_values, # Specify the data to use for the plot
  xlab = "Institution Quality", # Specify the X-axis title
  ylab = "Fitted value for political stability", # Specify the Y-axis title
  frame.plot = FALSE, # The frame.plot = FALSE argument removes the box from around the plot
  col = "blue", # The col argument specifies the color
  type = "l", # type = "l" will produce a line plot, rather than the default scatter plot
  lwd = 3 # lwd = 3 will increase the thinkness of the line on the plot
)

## Add an additional line of fitted values over the range of institution quality for dictatorships
lines(x = data_for_fitted_values_dictator$institutions_quality,
      y = data_for_fitted_values_dictator$fitted_values,
      col = "red",
      lwd = 3)
```



We can see from the plot that the fitted values for democracies (blue line) are almost exactly the same as those for dictatorships (red line). This is reassuring, as the estimated coefficient on the democracy variable was very small (0.04) and was not statistically significantly different from 0. Often, however, it can be very illuminating to construct plots like this where we construct a line to indicate how our predicted values for Y vary across one of our explanatory variables (here, institution quality), and we create different lines for different values of another explanatory variable (here, democracy/dictatorship).

7.1.6 Additional Resources

Visualizing Distributions

7.1.7 Exercises

1. Open a new script and save it as assignment5.
2. Go to the QoG website and download the standard data and the accompanying codebook.
3. Pretend that you are writing a research paper. Select one dependent variable that you want to explain with a statistical model.
4. Run a simple regression (with one explanatory variable) on that dependent variable and justify your choice of explanatory variable. This will require you to think about the outcome that you are trying to explain, and what you think will be a good predictor for that outcome.
5. Add some additional explanatory variables (no more than 5) to your model and justify the choice again.
6. Carry out the F-test to check whether the new model explains more than the old model.
7. Produce a regression table of the better model.
8. Interpret the output of the better model. State your expectations and whether they were met.
9. Calculate fitted values whilst varying at least one of your continuous explanatory variables.

10. Plot the result.
11. Interpret the plot.
12. Source your entire script. If it produces errors, clean your script.

7.1.8 Midterm Preparation

We wrote a midterm preparation exercise for you. It envolves working a little bit with data (in section 3). You will obviously not get questions where you work with R, so you can skip that part of section 3 if you want.

Midterm Preparatory Questions Prepartory Questions Dataset Midterm Preparatory Response

8 Multiple linear regression models (II)

8.1 Seminar

In the first part of this seminar, we cover R^2 and adjusted R^2. In the second, we cover interactions. First, an interaction between continuous and binary independent variable and second, between two continuous variables. We also show the F test.

To start with, load the `foreign` library and the `texreg` library.

```
library(foreign) # to load non-native file formats
library(texreg) # to create better looking regression tables
```

In case `library(texreg)` throws an error message, you have to install the package first. A package is like an app on your mobile. It adds additional functionality. You need to install it only once. To install the package run `install.packages("texreg")`. Then load the library `library(texreg)`.

```
rm(list = ls())
```

8.1.1 Loading Data

We will use the small version of the Quality of Government data from 2012 again (`QoG2012.csv`) with four variables:

Variable	Description
`former_col`	0 = not a former colony 1 = former colony
`undp_hdi`	UNDP Human Development Index. Higher values mean better quality of life
`wbgi_cce`	Control of corruption. Higher values mean better control of corruption
`wdi_gdpc`	GDP per capita in US dollars

```
a <- read.csv("QoG2012.csv")
names(a)
```

```
[1] "h_j"         "wdi_gdpc"    "undp_hdi"    "wbgi_cce"    "wbgi_pse"
[6] "former_col"  "lp_lat_abst"
```

```
summary(a)
```

```
      h_j            wdi_gdpc          undp_hdi         wbgi_cce
 Min.   :0.0000   Min.   :  226.2   Min.   :0.2730   Min.   :-1.69953
 1st Qu.:0.0000   1st Qu.: 1768.0   1st Qu.:0.5390   1st Qu.:-0.81965
```

```
 Median :0.0000   Median : 5326.1   Median :0.7510   Median :-0.30476
 Mean   :0.3787   Mean   :10184.1   Mean   :0.6982   Mean   :-0.05072
 3rd Qu.:1.0000   3rd Qu.:12976.5   3rd Qu.:0.8335   3rd Qu.: 0.50649
 Max.   :1.0000   Max.   :63686.7   Max.   :0.9560   Max.   : 2.44565
 NA's   :25       NA's   :16        NA's   :19       NA's   :2
    wbgi_pse          former_col      lp_lat_abst
 Min.   :-2.46746   Min.   :0.0000   Min.   :0.0000
 1st Qu.:-0.72900   1st Qu.:0.0000   1st Qu.:0.1343
 Median : 0.02772   Median :1.0000   Median :0.2444
 Mean   :-0.03957   Mean   :0.6289   Mean   :0.2829
 3rd Qu.: 0.79847   3rd Qu.:1.0000   3rd Qu.:0.4444
 Max.   : 1.67561   Max.   :1.0000   Max.   :0.7222
                                     NA's   :7
```

Let's create a copy of the dataset and then remove all missing values.

```
# copy of original dataset
a.full <- a
```

To remove all missing values at once, we use the `apply()` function. It is very useful to repeat the same operations on all rows or columns of a dataset. The `apply()` function takes the following arguments:

argument	description
`X`	the name of the dataset on which we want to repeat the operation on
`MARGIN`	1 = rows, 2 = columns
`FUN`	operation that we want to repeat

So for example, `apply(X = a, MARGIN = 1, FUN = mean )` would return the mean value for every row. We will define our own operation. First, `!is.na()` means "is not missing". The function returns true or false for every cell. We want to keep rows where none of the observations are missings. Therefore, we use the `all()` function which is true if all cells are not missing.

```
# drop all missing values
a <- a[apply(a, 1, function(x) all(!is.na(x)) ), ]
```

If you do not fully understand this code, don't worry. It deletes rows from a dataset whenever a value on any variable is missing.

8.1.2 R Squared

Let's say, we want to predict the quality of life. Our variable that approximates this is called *undp_hdi* (the United Nations human development index). The variable is continuous and has a theoretical range from 0 to 1. Larger values correspond to better life quality.

If we did not have any information on a country, our best prediction for every country would be the mean of *undp_hdi*. We would make some mistakes. But on average, this would be our best prediction. Let's confirm that this is the case.

```
# mean of undp_hdi
y_bar <- mean(a$undp_hdi)
round(y_bar, digits = 2)
```

```
[1] 0.69
```

Our mean is 0.69. If we predict the mean for every country, we will make some mistakes. These mistakes are the differences between the actual values of *undp_hdi* in every country and the mean of *undp_hdi*.

```
# deviations from the mean
deviations.from.ybar <- (a$undp_hdi - y_bar)
```

It's always going to be true that the sum of the average deviations is 0. This is a property of the mean.

```
# sum of deviations from ybar is always zero (or extremely close to 0)
sum(deviations.from.ybar)
```

```
[1] 0.00000000000002553513
```

We will now square the deviations from the mean.

```
# squared deviations from the mean
sq.deviations.from.ybar <- deviations.from.ybar^2
```

The squared deviations from the mean capture the overall variability of our variable *undp_hdi*. At this point you should see that what we did so far, is the first step to getting variance. Dividing the sum of the squared deviations by $n - 1$ would give us the variance of *undp_hdi*. The variance quantifies the variability of a variable.

Let's do this:

```
total.variance <- sum(sq.deviations.from.ybar) / (length(a$undp_hdi) - 1)
total.variance
```

```
[1] 0.03434449
```

The overall variance of *undp_hdi* is 0.0343445.

Let's say, we have additional information about a country like for example its wealth. Wealth is measured by the variable *wdi_gdpc*. We now run a linear model where we predict *undp_hdi* using the information in *wdi_gdpc*. If the regression coefficient of *wdi_gdpc* is significant, that means that both variables are related. This in turn means that we can explain some of the variability in *undp_hdi* using variability in *wdi_gdpc*.

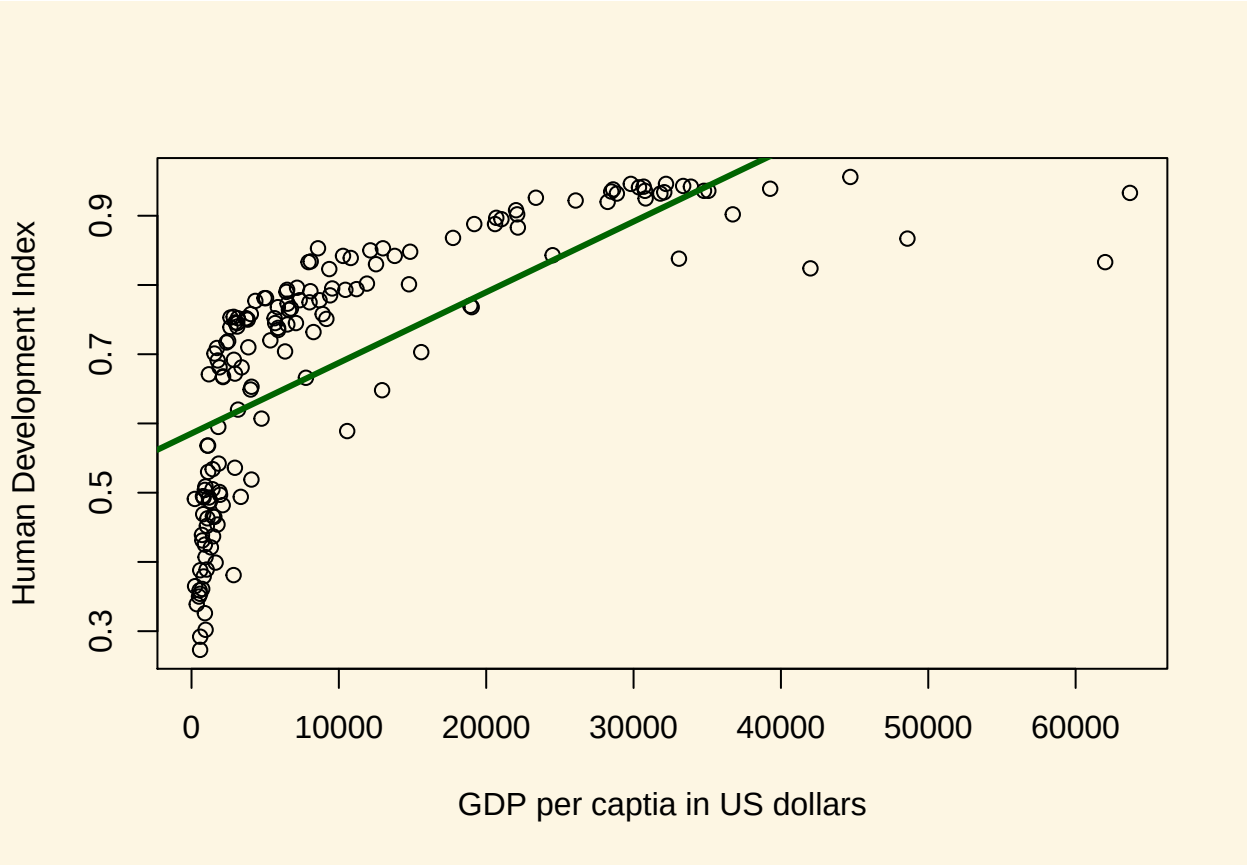
```
# we regress undp_hdi on wdi_gdpc
m1 <- lm( undp_hdi ~ wdi_gdpc, data = a )
screenreg(m1)
```

```
========================
             Model 1
------------------------
(Intercept)    0.59 ***
              (0.01)
wdi_gdpc       0.00 ***
              (0.00)
------------------------
R^2            0.50
Adj. R^2       0.50
Num. obs.    158
RMSE           0.13
========================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

The coefficient of *wdi_gdpc* is indeed significant. Let's plot the relationship in a scatterplot and draw the regression line.

```
# the scatterplot
plot(x = a$wdi_gdpc,
     y = a$undp_hdi,
```

```
     xlab = "GDP per captia in US dollars",
     ylab = "Human Development Index")
# the regression line
abline(m1, col = "darkgreen", lwd = 3)
```



This model looks awful. We over-predict quality of live for poor countries. We under-predict for medium wealth levels and over-predict for rich countries. We will return to this problem later.

For now, you should see that using the variability in *wdi_gdpc* we actually explain some of the variability in *undp_hdi*. We do not explain all of the variability. There are still differences between our predictions (the points on the line) and actual outcomes. Were we to explain all of the variability, all of the points would be on the line. They are not. They never will be in real-life applications. The social world is complex. However, we can ask: "How much of the variability in *undp_hdi* do we explain using the variability in *wdi_gdpc*.

To answer this question, we first extract a fitted value for every observation in the dataset. Recall that a fitted value is a point on the line. Hence, we take the regression equation $\hat{Y} = 0.5855759 + 0.000102 \times \text{GDP per capita}$ and plug in the value of *wdi_gdpc* for every row in the dataset.

Actually, R has already done that for us. We can access the fitted values from our model object `m1` like so.

```
fitted.vals <- m1$fitted.values
```

Now, we take the same steps as we did earlier. We take the deviations between the actual outcome of *undp_hdi* and our model predictions (the fitted values). These differences are the mistakes, we make they are called residuals.

```
# residuals
resids <- (a$undp_hdi - fitted.vals)
```

Actually, R did that for us as well. We could have accessed the residuals as `m1$residuals`. We again square the deviations between the model predictions and the actual outcomes.

```
# squared residuals
sq.resids <- resids^2
```

The squared residuals are the variability in *undp_hdi* which we have *NOT* explained with the variability in *wdi_gdpc*.

R^2 is then: 1 - unexplained variability / total variability. R^2, therefore, answers the question "how much of the total variability in *undp_hdi* do we explain using variability in *wdi_gdpc*."

```
R.sq <- 1 - sum(sq.resids) / sum(sq.deviations.from.ybar)
R.sq
```

```
[1] 0.4990749
```

We have successfully estimated R^2. There are other ways to get there. For instance, we could compare the variances instead of sums of squared deviations.

Note: The difference between our estimate of R^2 and the one from the regression table is due to rounding error. We get the same value if we use the `round()` function to round to 2 digits like so:

```
round(R.sq, digits = 2)
```

```
[1] 0.5
```

8.1.2.1 R Squared - Approach 2

Let's take the unexplained variance of *undp_hdi* instead of the unexplained sum of squares.

```
# unexplained variance
unexplained.variance <- sum(sq.resids) / (length(sq.resids) - 1)
```

R^2 is then 1 - unexplained variance over total variance.

```
R.sq <- 1- unexplained.variance / total.variance
R.sq
```

```
[1] 0.4990749
```

8.1.2.2 R Squared - Approach 3

Let's use the explained sum of squares instead of the unexplained sum of squares, i.e.,

$$R^2 = \frac{ESS}{TSS}$$

.

```
# explained sum of squares
ESS <- sum((fitted.vals - y_bar)^2)

# R^2
R.sq <- ESS/ sum(sq.deviations.from.ybar)
R.sq
```

```
[1] 0.4990749
```

8.1.2.3 R Squared - Approach 4

You may have already noticed that R^2 looks very similar to the correlation coefficient. That's right. In fact, we can quickly estimate R^2 by taking the squared correlation between our fitted values and the actual outcomes.

```
R.sq <- cor(m1$fitted.values, a$undp_hdi)^2
R.sq
```

```
[1] 0.4990749
```

8.1.2.4 Adjusted R^2

R^2 always weakly increases if we include more X variables into our model. The reason is that the correlation between two variables is never exactly zero. That means in any sample, two variables are always related to some degree. They may not be related in the same way in the population, however. So, the relationship between two variables that we see in our sample may just be noise.

Yet, R^2 increases whenever two variables are correlated. R^2 never decreases. Adjusted R^2 accounts for the number of predictors that we have added two our model by adding a penalty that increases as we increase the number of X variables in our model. The penalty looks like this:

We multiply R^2 by $\frac{n-1}{n-k-1}$, where n is the number of observations and k is the number of X variables. So, assuming our sample size is 158 and we add 1 predictor to our model that so far only included political stability, we would multiply R^2 by

```
# number of observations
n <- length(sq.resids)
# number of X variables
k <- length(m1$coefficients) - 1
# penalty
penalty <- (n-1) / (n-k-1)
adj.R.sq <- 1 - penalty * (sum(sq.resids) / sum(sq.deviations.from.ybar))
adj.R.sq
```

```
[1] 0.4958638
```

Since the formula for adjusted R^2 is:

$$1 - \frac{n-1}{n-k-1} \times \frac{\text{Sum of unexplained variability}}{\text{Sum of total variability}}$$

we can estimate adjusted R^2 by rearranging the formula to:

$$1 - (1 - R^2) \times \frac{n-1}{n-k-1}$$

Let's compute adjusted R^2 directly from R^2:

```
adj.R.sq <- 1 - (1 - R.sq) * penalty
adj.R.sq
```

```
[1] 0.4958638
```

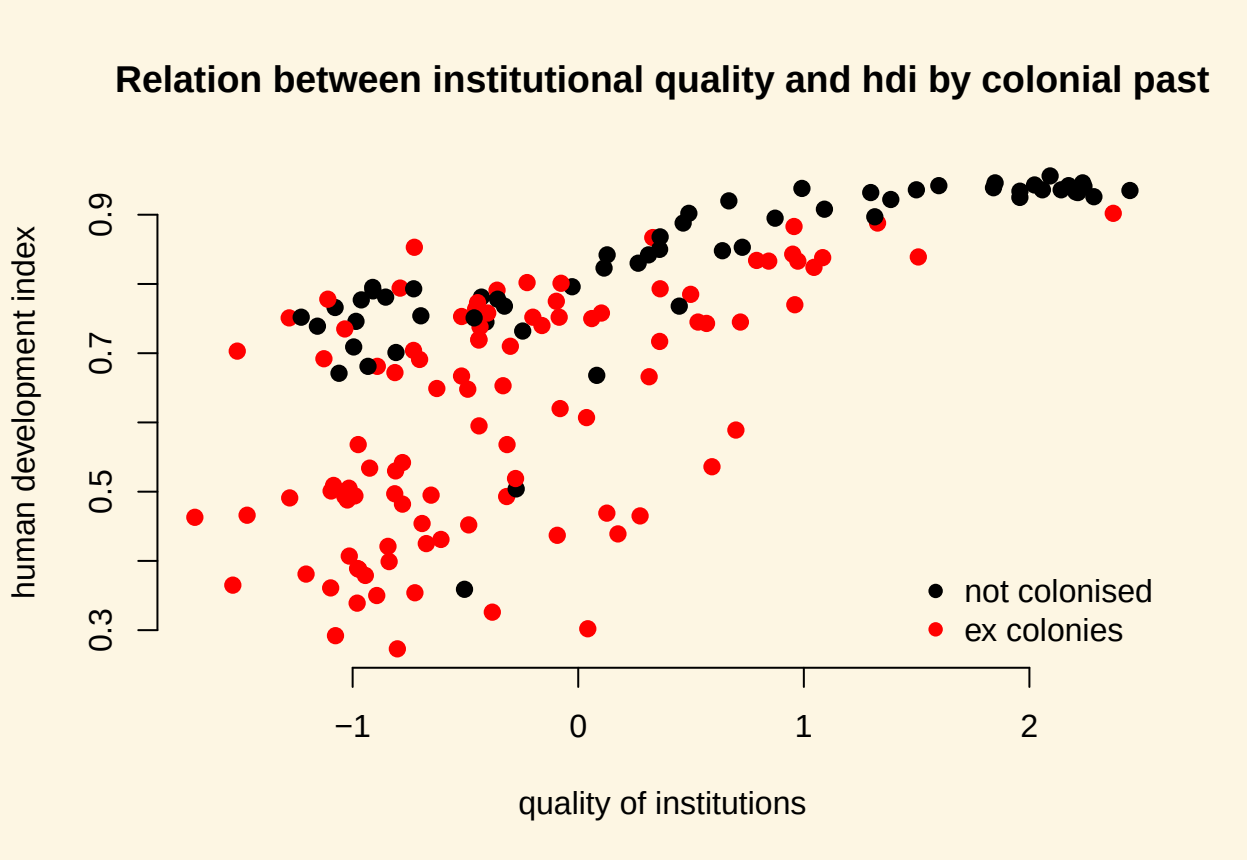
8.1.3 The Relationship between Institutional Quality and Quality of Life by Colonial Past

Let's create a scatterplot between `wbgi_cce` and `undp_hdi` and color the points based on the value of `former_col`.

NOTE: We're using `pch = 16` to plot solid circles. You can see other available styles by typing `?points` or `help(points)` at the console.

Copy the plot command in the seminar, you can go over it at home.

```
# main plot
plot(
  x = a$wbgi_cce,
  y = a$undp_hdi,
  col = factor(a$former_col),
  pch = 16,
  cex = 1.2,
  bty = "n",
  main = "Relation between institutional quality and hdi by colonial past",
  xlab = "quality of institutions",
  ylab = "human development index"
  )
# add a legend
legend(
  "bottomright",  # position fo legend
  legend = c("not colonised", "ex colonies"), # what to seperate by
  col = factor(a$former_col), # colors of legend labels
  pch = 16, # dot type
  bty = "n" # no box around the legend
  )
```



To explain the level of development with quality of institutions is intuitive. We could add the colonial past dummy, to control for potential confounders. Including a dummy gives us the difference between former

colonies and not former colonies. It therefore shifts the regression line parallelly. We have looked at binary variables in the last weeks. To see the effect of a dummy again, refer to the extra info at the bottom of page.

8.1.4 Interactions: Continuous and Binary

From the plot above, we can tell that the slope of the line (the effect of institutional quality) is probably different in countries that were colonies and those that were not. We say: the effect of institutional quality is conditional on colonial past.

To specify an interaction term, we use the asterisk (`*`)

	Example
*	`A*B` - In addition to the interaction term (A*B), both the constituents (A and B) are automatically included.

```
m2 <- lm(undp_hdi ~ wbgi_cce * former_col, data = a)
screenreg( m2 )
```

```
================================
                      Model 1
--------------------------------
(Intercept)             0.79 ***
                       (0.02)
wbgi_cce                0.07 ***
                       (0.01)
former_col             -0.13 ***
                       (0.02)
wbgi_cce:former_col     0.06 **
                       (0.02)
--------------------------------
R^2                     0.59
Adj. R^2                0.58
Num. obs.             158
RMSE                    0.12
================================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

We set our covariate `former_col` to countries that weren't colonized and then second, to ex colonies. We vary the quality of institutions from `-1.7` to `2.5` which is roughly the minimum to the maximum of the variable.

NOTE: We know the range of values for `wbgi_cce` from the summary statistics we obtained after loading the dataset at the beginning of the seminar. You can also use the `range()` function.

```
# minimum and maximum of the quality of institutions
range(a$wbgi_cce)
```

```
[1] -1.699529  2.445654
```

We now illustrate what the interaction effect does. To anticipate, the effect of the quality of institutions is now conditional on colonial past. That means, the two regression lines will have different slopes.

We make use of the `predict()` function to draw both regression lines into our plot. First, we need to vary the institutional quality variable from its minimum to its maximum. We use the `seq()` (sequence) function to create 10 different institutional quality values. Second, we create two separate covariate datasets. In the

first, x1, we set the `former_col` variable to never colonies. In the second, x2, we set the same variable to ex colonies. We then predict the fitted values `y_hat1`, not colonised countries, and `y_hat2`, ex colonies.

```
# sequence of 10 institutional quality values
institutions_seq <- seq(from = -1.7, to = 2.5, length.out = 10)

# covariates for not colonies
x1 <- data.frame(former_col = 0, wbgi_cce = institutions_seq)
# look at our covariates
head(x1)
```

```
  former_col   wbgi_cce
1          0 -1.7000000
2          0 -1.2333333
3          0 -0.7666667
4          0 -0.3000000
5          0  0.1666667
6          0  0.6333333
```

```
# covariates for colonies
x2 <- data.frame(former_col = 1, wbgi_cce = institutions_seq)
# look at our covariates
head(x2)
```

```
  former_col   wbgi_cce
1          1 -1.7000000
2          1 -1.2333333
3          1 -0.7666667
4          1 -0.3000000
5          1  0.1666667
6          1  0.6333333
```

```
# predict fitted values for countries that weren't colonised
yhat1 <- predict(m2, newdata = x1)

# predict fitted values for countries that were colonised
yhat2 <- predict(m2, newdata = x2)
```

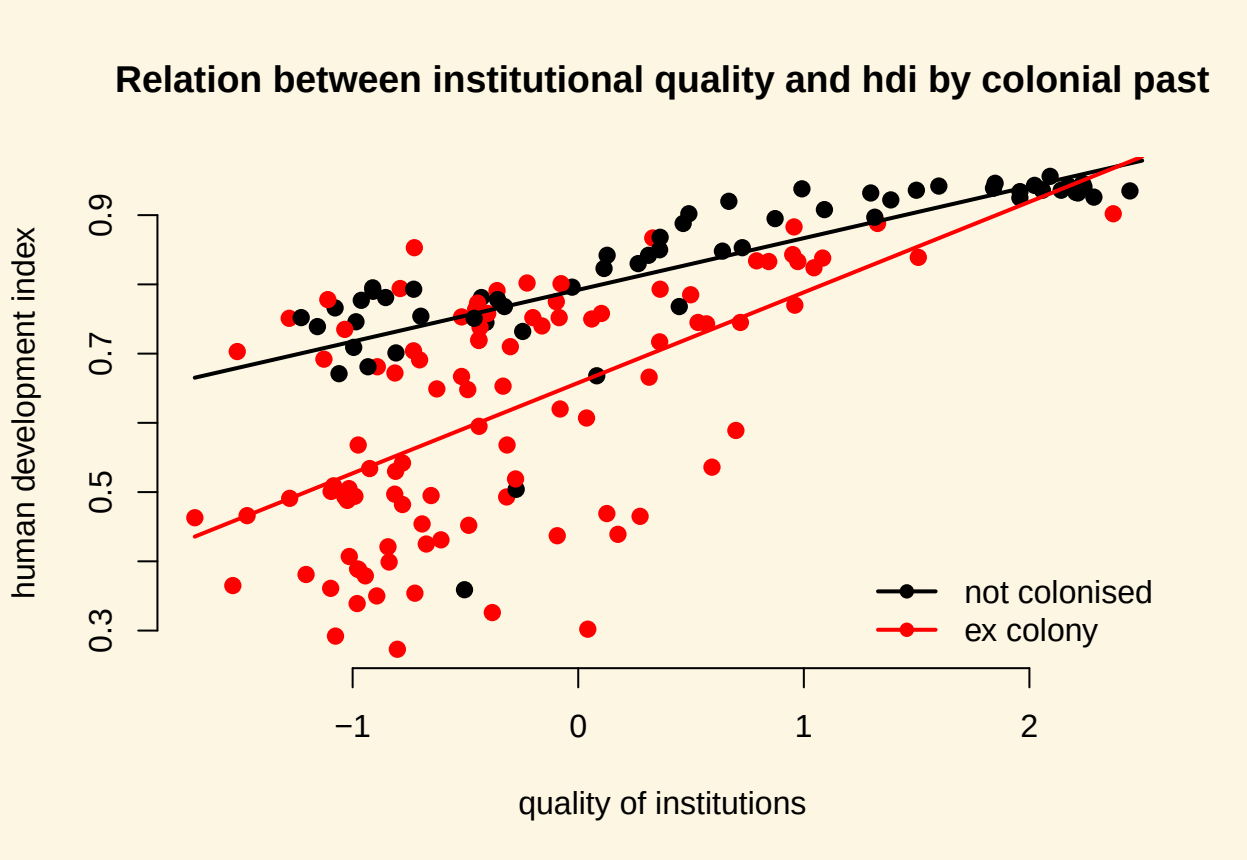
We now have the predicted outcomes for varying institutional quality. Once for the countries that were former colonies and once for the countries that were not.

We will re-draw our earlier plot. In addition, right below the `plot()` function, we use the `lines()` function to add the two regression lines. The function needs to arguments `x` and `y` which represent the coordinates on the respective axes. On the x axis we vary our independent variable quality of institutions. On the y axis, we vary the predicted outcomes.

We add two more arguments to our `lines()` function. The line width is controlled with `lwd` and we set the colour is controlled with `col` which we set to the first and second colours in the colour palette respectively.

```
# main plot
plot(
  y = a$undp_hdi,
  x = a$wbgi_cce,
  frame.plot = FALSE,
  col = factor(a$former_col),
  pch = 16,
  cex = 1.2,
  bty = "n",
```

```
  main = "Relation between institutional quality and hdi by colonial past",
  xlab = "quality of institutions",
  ylab = "human development index"
  )

# add the regression line for the countries that weren't colonised
lines(x = institutions_seq, y = yhat1, lwd = 2, col = 1)

# add the regression line for the ex colony countries
lines(x = institutions_seq, y = yhat2, lwd = 2, col = 2)

# add a legend
legend(
  "bottomright",  # position fo legend
  legend = c("not colonised", "ex colony"), # what to seperate by
  col = factor(a$former_col), # colors of legend labels
  pch = 16, # dot type
  lwd = 2, # line width in legend
  bty = "n" # no box around the legend
  )
```



As you can see, the line is steeper for ex colonies than for countries that were never colonised. That means the effect of institutional quality on human development is conditional on colonial past. Institutional quality matters more in ex colonies.

Let's examine the effect sizes of institutional quality conditional on colonial past.

$$\hat{y} = \beta_0 + \beta_1 \times \text{wbgi}_\text{c}\text{ce} + \beta_2 \times \text{former\_col} + \beta_3 \times \text{wbgi}_\text{c}\text{ce} \times \text{former\_col} \quad (1)$$

$$\hat{y} = 0.79 + 0.08 \times \text{wbgi}_\text{c}\text{ce} + -0.12 \times \text{former\_col} + 0.05 \times \text{wbgi}_\text{c}\text{ce} \times \text{former\_col} \quad (2)$$

There are now two scenarios. First, we look at never coloines or second, we look at ex colonies. Let's look at never colonies first.

If a country was never a colony, all terms that are multiplied with `former_col` drop out.

$$\hat{y} = 0.79 + 0.08 \times \text{wbgi}_\text{c}\text{ce} + -0.12 \times 0 + 0.05 \times 0 \quad (3)$$

$$\hat{y} = 0.79 + 0.08 \times \text{wbgi}_\text{c}\text{ce} \quad (4)$$

Therefore, the effect of the quality of institutions (measured by *wbgi_cce*) in never colonies is just the coefficient of `wbgi_cce` $\beta_1 = 0.08$.

In the second scenario, we are looking at ex colonies. In this case none of the terms drop out. From our original equation:

$$\hat{y} = 0.79 + 0.08 \times \text{wbgi}_\text{c}\text{ce} + -0.12 \times \text{former\_col} + 0.05 \times \text{wbgi}_\text{c}\text{ce} \times \text{former\_col} \quad (5)$$

$$\hat{y} = 0.79 + 0.08 \times \text{wbgi}_\text{c}\text{ce} + -0.12 \times 1 + 0.05 \times \text{wbgi}_\text{c}\text{ce} \times 1 \quad (6)$$

$$\hat{y} = 0.79 - 0.12 + 0.08 \times \text{wbgi}_\text{c}\text{ce} + 0.05 \times \text{wbgi}_\text{c}\text{ce} \quad (7)$$

$$\hat{y} = 0.67 + 0.08 \times \text{wbgi}_\text{c}\text{ce} + 0.05 \times \text{wbgi}_\text{c}\text{ce} \quad (8)$$

The effect of the quality of institutions is then: $\beta_1 + \beta_3 = 0.08 + 0.05 = 0.13$.

The numbers also tell us that the effect of the quality of institutions is bigger in ex colonies. For never colonies the effect is 0.08 for every unit-increase in institutional quality. For ex colonies, the corresponding effect is 0.13.

The table below summarises the interaction of a continuous variable with a binary variable in the context of our regression model.

Ex Colony	Intercept	Slope
0 = never colony	$\beta_0 = 0.79$	$\beta_1 = 0.08$
1 = ex colony	$\beta_0 + \beta_2 = 0.79 + -0.12 = 0.67$	$\beta_1 + \beta_3 = 0.08 + 0.05 = 0.13$

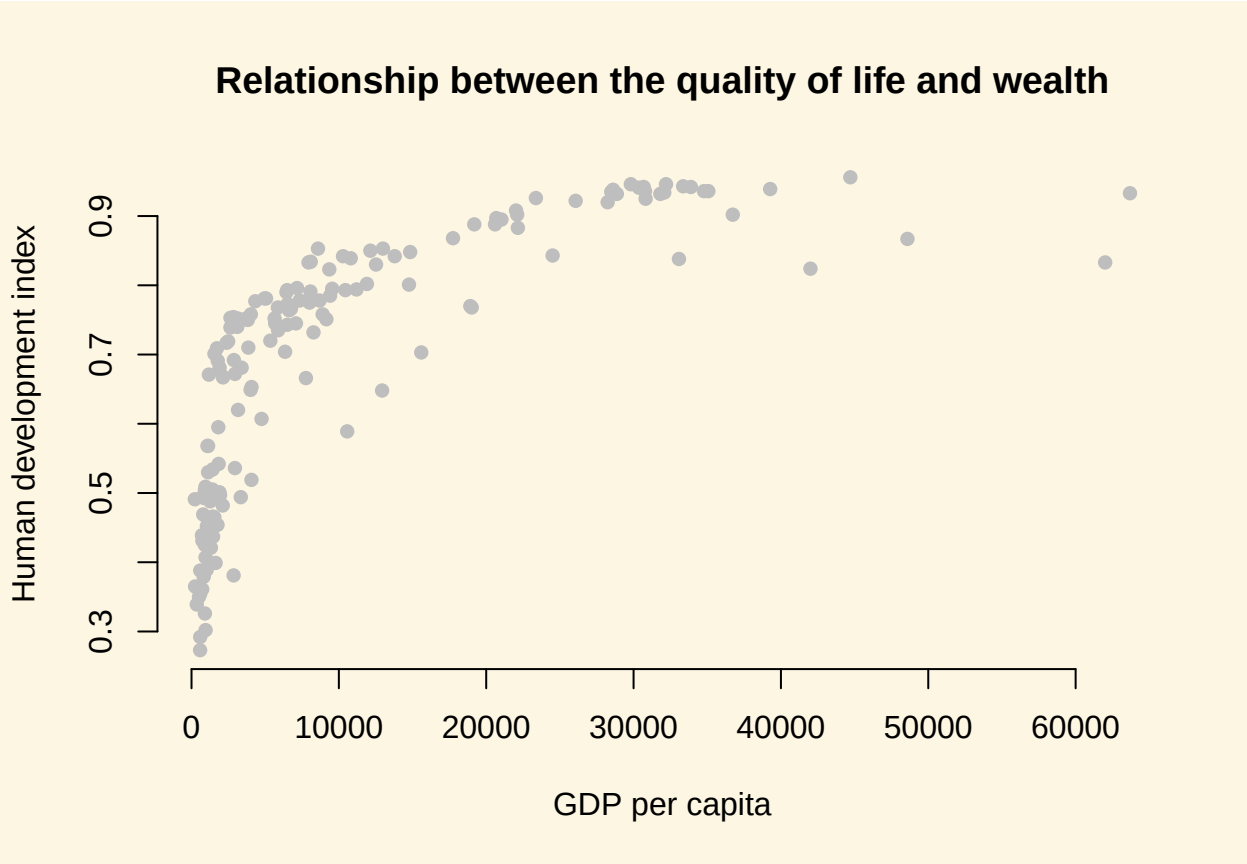
8.1.5 Non-Linearities

We can use interactions to model non-linearities. Let's suppose we want to illustrate the relationship between GDP per capita and the human development index.

We draw a scatter plot to investigate the relationship between the quality of life (hdi) and wealth (gdp/captia).

```
plot(
  y = a$undp_hdi,
  x = a$wdi_gdpc,
  pch = 16,
  frame.plot = FALSE,
  col = "grey",
```

```
  main = "Relationship between the quality of life and wealth",
  ylab = "Human development index",
  xlab = "GDP per capita"
  )
```



It's easy to see, that the relationship between GDP per captia and the Human Development Index is not linear. Increases in wealth rapidly increase the quality of life in poor societies. The richer the country, the less pronounced the effect of additional wealth. We would mis-specify our model if we do not take the non-linear relationship into account.

Let's go ahead and mis-specify our model :-)

```
# a mis-specified model
bad.model <- lm(undp_hdi ~ wdi_gdpc, data = a)
screenreg( bad.model )
```

```
========================
              Model 1
------------------------
(Intercept)     0.59 ***
               (0.01)
wdi_gdpc        0.00 ***
               (0.00)
------------------------
R^2             0.50
Adj. R^2        0.50
```

```
Num. obs.    158
RMSE           0.13
=======================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

We detect a significant linear relationship. The effect may look small because the coefficient rounded to two digits is zero. But remember, this is the effect of increasing GDP/capita by 1 US dollar on the quality of life. That effect is naturally small but it is probably not small when we increase wealth by 1000 US dollars.

However, our model would also entail that for every increase in GDP/capita, the quality of life increases on average by the same amount. We saw from our plot that this is not the case. The effect of GDP/capita on the quality of life is conditional on the level of GDP/capita. If that sounds like an interaction to you, then that is great because, we will model the non-linearity by raising the GDP/capita to a higher power. That is in effect an interaction of the variable with itself. GDP/capita raised to the second power, e.g. is GDP/capita * GDP/capita.

8.1.5.1 Polynomials

We know from school that polynomials like x^2, x^3 and so on are not linear. In fact, x^2 can make one bend, x^3 can make two bends and so on.

Our plot looks like the relationship is quadratic. So, we use the `poly()` function in our linear model to raise GDP/capita to the second power like so: `poly(wdi_gdpc, 2)`.

```
better.model <- lm(undp_hdi ~ poly(wdi_gdpc, 2), data = a)
screenreg( list(bad.model, better.model),
           custom.model.names = c("bad model", "better model"))
```

```
=============================================
                     bad model   better model
---------------------------------------------
(Intercept)            0.59 ***    0.69 ***
                      (0.01)      (0.01)
wdi_gdpc               0.00 ***
                      (0.00)
poly(wdi_gdpc, 2)1                 1.64 ***
                                  (0.11)
poly(wdi_gdpc, 2)2                -0.98 ***
                                  (0.11)
---------------------------------------------
R^2                    0.50        0.68
Adj. R^2               0.50        0.67
Num. obs.            158         158
RMSE                   0.13        0.11
=============================================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

It is important to note, that in the better model the effect of GDP/capita is no longer easy to interpret. We cannot say for every increase in GDP/capita by one dollar, the quality of life increases on average by this much. No, the effect of GDP/capita depends on how rich a country was to begin with.

It looks like our model that includes the quadratic term has a much better fit. The adjusted R^2 increases by a lot. Furthermore, the quadratic term, `poly(gdp_capita, 2)2` is significant. That indicates that newly added variable improves model fit. We can run an F-test with `anova()` function which will return the same result. The F-test would be useful when we add more than one new variable, e.g. we could have raised GDP_captia to the power of 5 which would have added four new variables.

```r
# f test
anova(bad.model, better.model)
```

```
Analysis of Variance Table

Model 1: undp_hdi ~ wdi_gdpc
Model 2: undp_hdi ~ poly(wdi_gdpc, 2)
  Res.Df    RSS Df Sum of Sq      F                Pr(>F)
1    156 2.7010
2    155 1.7384  1   0.96263 85.831 < 0.00000000000000022 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can interpret the effect of wealth (GDP/capita) on the quality of life (human development index) by predicting the fitted values of the human development index given a certain level of GDP/capita. We will vary GDP/captia from its minimum in the data to its maximum and the plot the results which is a good way to illustrate a non-linear relationship.

Step 1: We find the minimum and maximum values of GDP/capita.

```r
# find minimum and maximum of per capita gdp
range(a$wdi_gdpc)
```

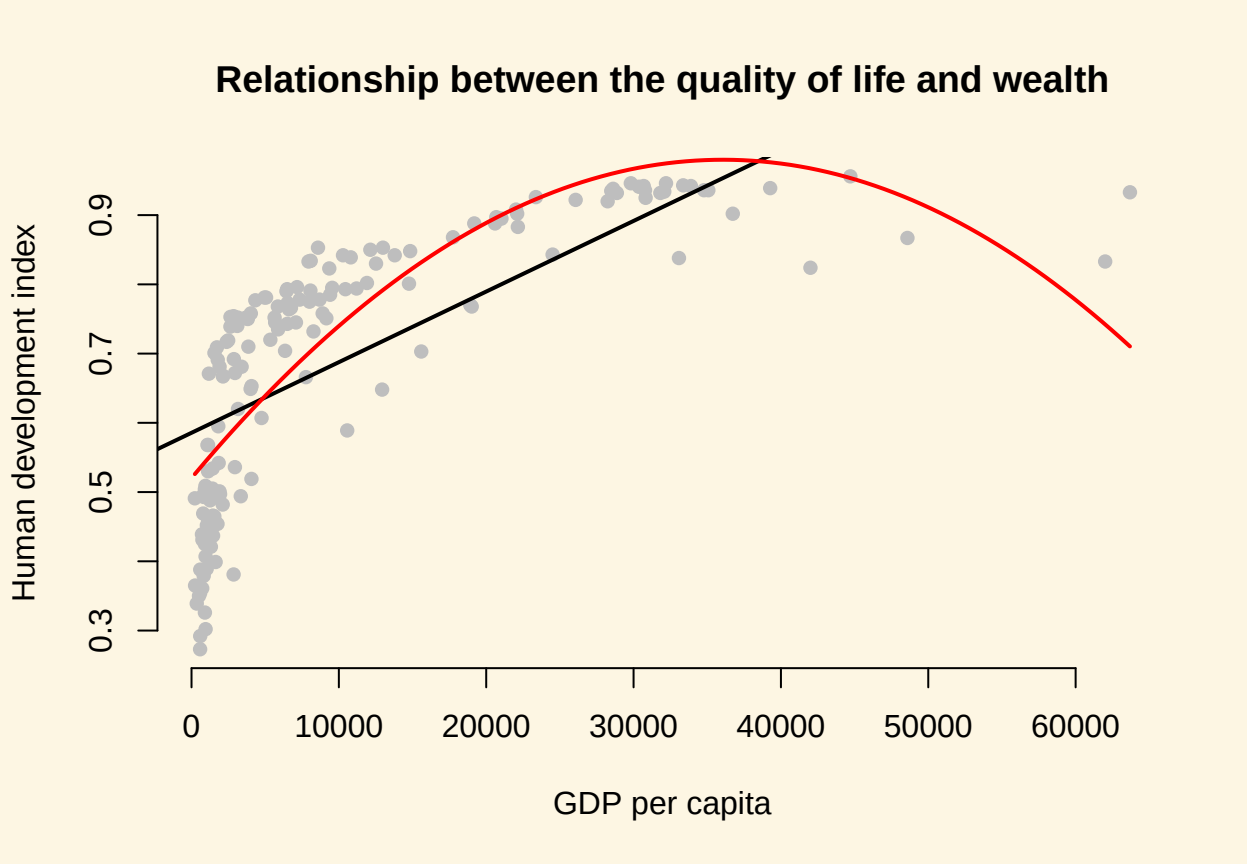
```
[1]   226.235 63686.676
```

Step 2: We predict fitted values for varying levels of GDP/captia (let's create 100 predictions).

```r
# our sequence of 100 GDP/capita values
gdp_seq <- seq(from = 226, to = 63686, length.out = 100)

# we set our covarite values (here we only have one covariate: GDP/captia)
x <- data.frame(wdi_gdpc = gdp_seq)

# we predict the outcome (human development index) for each of the 100 GDP levels
y_hat <- predict(better.model, newdata = x)
```

Step 3: Now that we have created our predictions. We plot again and then we add the `bad.model` using `abline` and we add our non-linear version `better.model` using the `lines()` function.

```r
plot(
  y = a$undp_hdi,
  x = a$wdi_gdpc,
  pch = 16,
  frame.plot = FALSE,
  col = "grey",
  main = "Relationship between the quality of life and wealth",
  ylab = "Human development index",
  xlab = "GDP per capita"
  )

# the bad model
abline(bad.model, col = 1, lwd = 2)

# better model
lines(x = gdp_seq, y = y_hat, col = 2, lwd = 2)
```



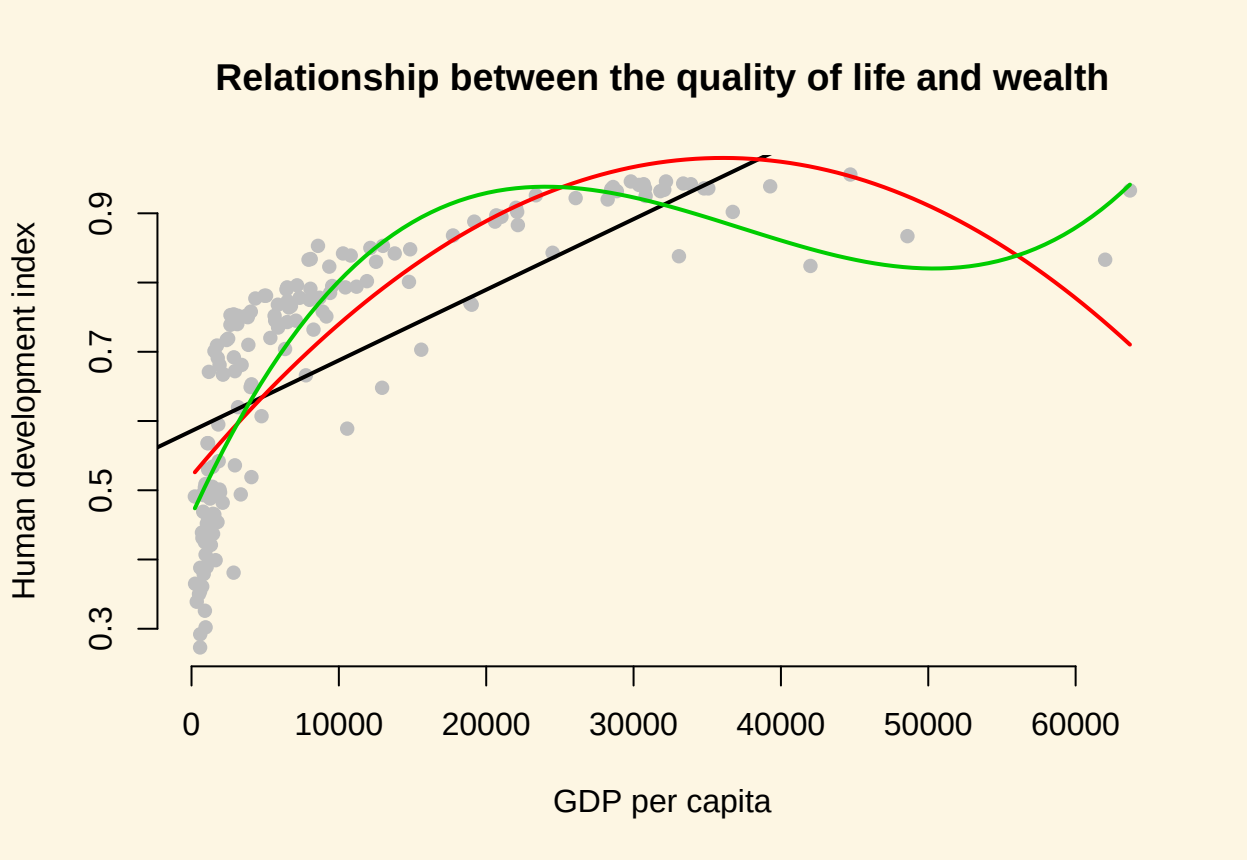
At home, we want you to estimate `even.better.model` with GDP/capita raised to the power of three to determine whether the data fit improves. Show this visually and with an F test.

```
# estimate even better model with gdp/capita^3
even.better.model <- lm(undp_hdi ~ poly(wdi_gdpc, 3), data = a)

# f test
anova(better.model, even.better.model)
```

```
Analysis of Variance Table

Model 1: undp_hdi ~ poly(wdi_gdpc, 2)
Model 2: undp_hdi ~ poly(wdi_gdpc, 3)
  Res.Df    RSS Df Sum of Sq      F          Pr(>F)
1    155 1.7384
2    154 1.3346  1    0.4038 46.595 0.0000000001888 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# so, our even better.model is statistically significantly even better

# we predict the outcome (human development index) for each of the 100 GDP levels
y_hat2 <- predict(even.better.model, newdata = x)

plot(
  y = a$undp_hdi,
  x = a$wdi_gdpc,
```

```
  pch = 16,
  frame.plot = FALSE,
  col = "grey",
  main = "Relationship between the quality of life and wealth",
  ylab = "Human development index",
  xlab = "GDP per capita"
  )

# the bad model
abline(bad.model, col = 1, lwd = 2)

# better model
lines(x = gdp_seq, y = y_hat, col = 2, lwd = 2)

# even better model
lines(x = gdp_seq, y = y_hat2, col = 3, lwd = 2)
```



We generate an even better fit with the cubic, however it still looks somewhat strange. The cubic is being wagged around by its tail. The few extreme values cause the strange shape. This is a common problem with polynomials. We move on to an alternative.

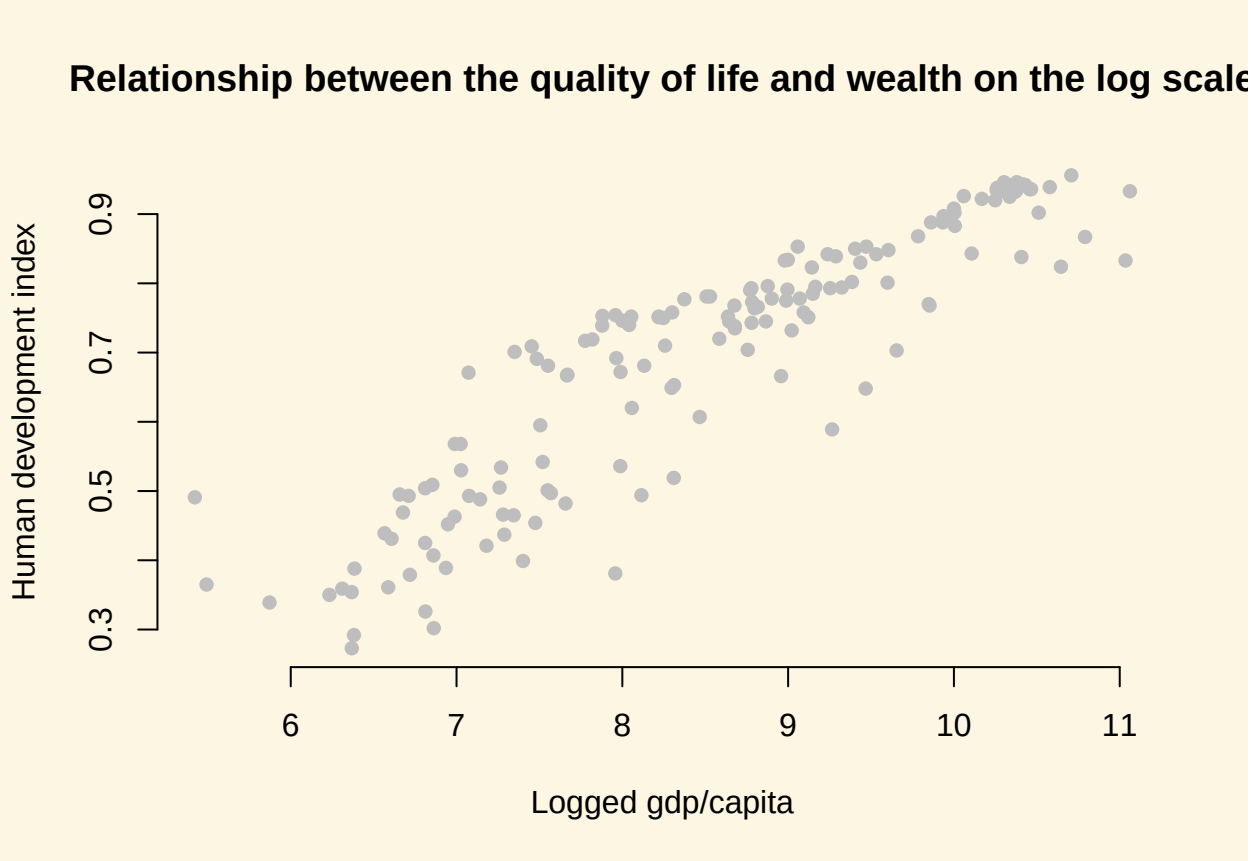
8.1.5.2 Log-transformations

Many non-linear relationships actually do look linear on the log scale. We can illustrate this by taking the natural logarithm of GDP/captia and plot the relationship between quality of life and our transformed GDP variable.

Note: Some of you will remember from your school calculators that you have an ln button and a log button where ln takes the natural logarithm and log takes the logarithm with base 10. The natural logarithm represents relations that occur frequently in the world and R takes the natural logarithm with the `log()` function by default.

Below, we plot the same plot from before but we wrap `gdp_capita` in the `log()` function which log-transforms the variable.

```
plot(
  y = a$undp_hdi,
  x = log(a$wdi_gdpc),
  pch = 16,
  frame.plot = FALSE,
  col = "grey",
  main = "Relationship between the quality of life and wealth on the log scale",
  ylab = "Human development index",
  xlab = "Logged gdp/capita"
  )
```



As you can see, the relationship now looks linear and we get the best fit to the data if we run our model with log-transformed gdp.

```
# run model with log-transformed gdp
best.model <- lm(undp_hdi ~ log(wdi_gdpc), data = a)

# let's check our model
screenreg( list(bad.model, better.model, even.better.model, best.model),
           custom.model.names = c("Bad Model", "Better Model", "Even Better Model", "Best Model"))
```

```
=============================================================================
                    Bad Model   Better Model  Even Better Model  Best Model
-----------------------------------------------------------------------------
(Intercept)          0.59 ***    0.69 ***      0.69 ***          -0.37 ***
                    (0.01)      (0.01)        (0.01)             (0.04)
wdi_gdpc             0.00 ***
                    (0.00)
poly(wdi_gdpc, 2)1               1.64 ***
                                (0.11)
poly(wdi_gdpc, 2)2              -0.98 ***
                                (0.11)
poly(wdi_gdpc, 3)1                            1.64 ***
                                             (0.09)
poly(wdi_gdpc, 3)2                           -0.98 ***
                                             (0.09)
poly(wdi_gdpc, 3)3                            0.64 ***
                                             (0.09)
log(wdi_gdpc)                                                    0.13 ***
                                                                (0.00)
-----------------------------------------------------------------------------
R^2                  0.50        0.68          0.75               0.82
Adj. R^2             0.50        0.67          0.75               0.82
Num. obs.          158         158           158                158
RMSE                 0.13        0.11          0.09               0.08
=============================================================================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

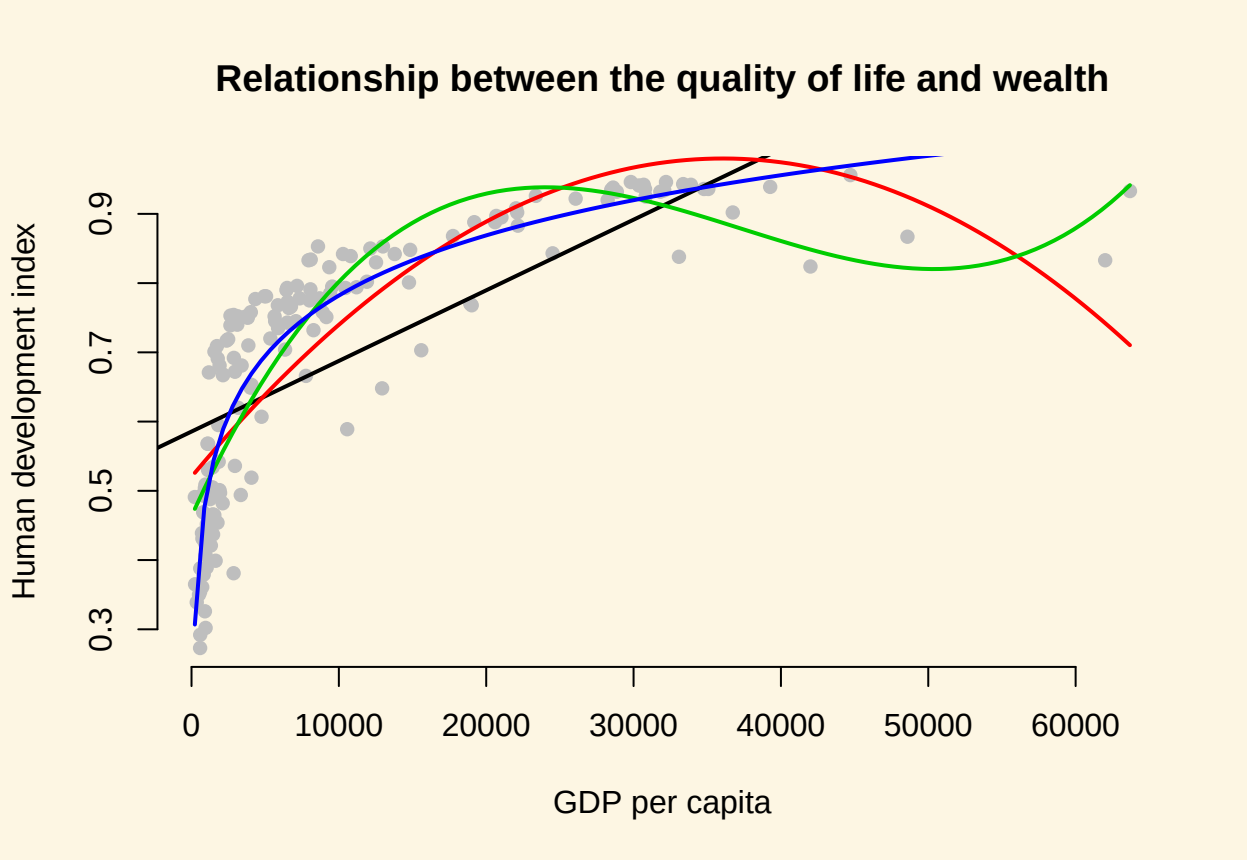
Polynomials can be useful for modelling non-linearities. However, for each power we add an additional parameter that needs to be estimated. This reduces the degrees of freedom. If we can get a linear relationship on the log scale, one advantage is that we lose only one degree of freedom. Furthermore, we gain interpretability. The relationship is linear on the log scale of gdp/capita. This means we can interpret the effect of gdp/captia as: For an increase of gdp/captia by one percent, the quality of life increases by $\frac{0.12}{100}$ points on average. The effect is very large because `human_development` only varies from 0 to 1.

To assess model fit, the f test is not very helpful here because, the initial model and the log-transformed model estimate the same number of parameters (the difference in the degrees of freedom is 0). Therefore, we rely on adjusted R^2 for interpretation of model fit. It penalises for additional parameters. According to our adjusted R^2, the log-transformed model provides the best model fit.

To illustrate that this is the case, we return to our plot and show the model fit graphically.

```
# fitted values for the log model (best model)
y_hat3 <- predict(best.model, newdata = x)

# plot showing the fits
plot(
  y = a$undp_hdi,
  x = a$wdi_gdpc,
  pch = 16,
  frame.plot = FALSE,
  col = "grey",
  main = "Relationship between the quality of life and wealth",
  ylab = "Human development index",
  xlab = "GDP per capita"
  )
```

```r
# the bad model
abline(bad.model, col = 1, lwd = 2)

# better model
lines(x = gdp_seq, y = y_hat, col = 2, lwd = 2)

# even better model
lines(x = gdp_seq, y = y_hat2, col = 3, lwd = 2)

# best model
lines(x = gdp_seq, y = y_hat3, col = 4, lwd = 2)
```



The dark purple line shows the log-transformed model. It clearly fits the data best.

8.1.6 Exercises

1. Using `better model`, where we included the square of GDP/capita, what is the effect of:
   a. an increase of GDP/capita from 5000 to 15000?
   b. an increase of GDP/capita from 25000 to 35000?
2. You can see that the curve in our quadratic plot curves down when countries become very rich. Speculate whether that results make sense and what the reason for this might be.
3. Raise GDP/captia to the highest power using the `poly()` that significantly improves model fit.
   a. Does your new model solve the potentially artefical down-curve for rich countries?
   b. Does the new model improve upon the old model?
   c. Plot the new model.

4. Estimate a model where `wbgi_pse` (political stability) is the response variable and `h_j` and `former_col` are the explanatory variables. Include an interaction between your explanatory variables. What is the marginal effect of:
   a. An independent judiciary when the country is a former colony?
   b. An independent judiciary when the country was not colonized?
   c. Does the interaction between `h_j` and `former_col` improve model fit?
5. Run a model on the human development index (`hdi`), interacting an independent judiciary (`h_j`) and `institutions_quality`. What is the effect of quality of institutions:
   a. In countries without an independent judiciary?
   b. When there is an independent judiciary?
   c. Illustrate your results.
   d. Does the interaction improve model fit?
6. Clear your workspace and download the California Test Score Data used by Stock and Watson.
   a. Download 'caschool.dta' Dataset
   b. Draw a scatterplot between `avginc` and `testscr` variables.
   c. Run two regressions with `testscr` as the dependent variable. c.a. In the first model use `avginc` as the independent variable. c.b. In the second model use quadratic `avginc` as the independent variable.
   d. Test whether the quadratic model fits the data better.

8.1.7 Extra Info: Dummy Variables Repetition

The variable `ex_colony` is binary. It takes on two values: "never colonies" or "ex colonies". The first value, "never colonies" is the baseline. When the variable takes on the value "ex colonies", we end up with an intercept shift. Consequently, we get a second parallel regression line.

```
model1 <- lm(human_development ~ institutions_quality + ex_colony, data = world_data)
screenreg(model1)
```

```
================================
                        Model 1
--------------------------------
(Intercept)              0.77 ***
                        (0.02)
institutions_quality     0.10 ***
                        (0.01)
ex_colonyex colonies    -0.11 ***
                        (0.02)
--------------------------------
R^2                      0.54
Adj. R^2                 0.54
Num. obs.              172
RMSE                     0.12
================================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

8.1.7.1 The Effect of a Dummy Variable

Ex Colony	Intercept	Slope
0 = not a former colony	`Intercept` + (`ex_colonyex colonies` * 0)= 0.77 + (-0.11 * 0)= 0.77	`institutions_quality` = 0.10

Ex Colony	Intercept	Slope
1 = former colony	`Intercept` + (`ex_colonyex colonies` * 1)= 0.77 + (-0.11 * 1)= 0.66	`institutions_quality` = 0.10

To illustrate the effect of a dummy we can access the coefficients of our `lm` model directly with the `coefficients()` function:

```
model_coef <- coefficients(model1)
model_coef
```

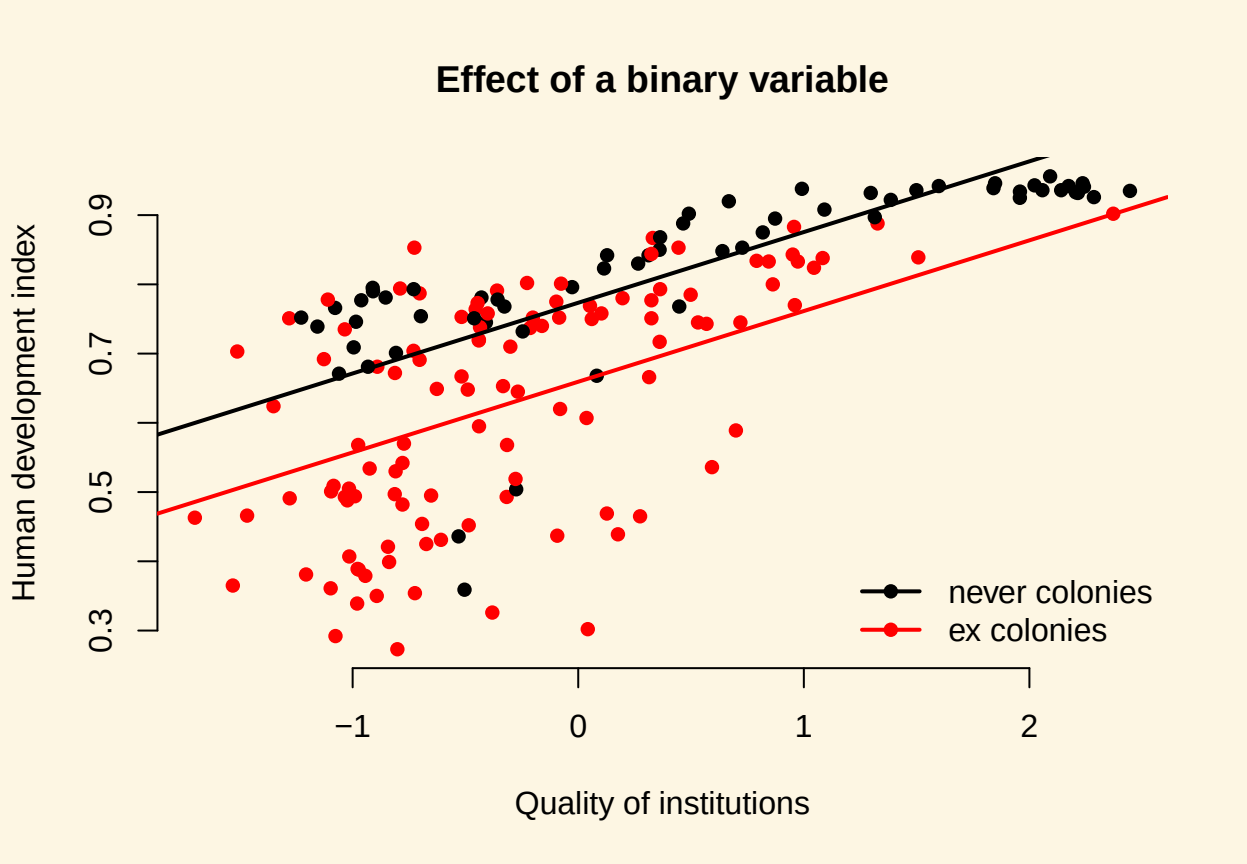
```
          (Intercept) institutions_quality ex_colonyex colonies
            0.7734319            0.1021146           -0.1140925
```

We can use the square brackets `[ ]` to access individual coefficients.

To illustrate the effect of a dummy we draw a scatterplot and then use `abline()` to draw two regression lines, one with `ex_colonyex colonies == "never colony"` and another with `ex_colony == "former colony"`.

Instead of passing the model as the first argument to `abline()`, we can just pass the intercept and slope as two separate arguments.

```
plot(
  human_development ~ institutions_quality,
  data = world_data,
  frame.plot = FALSE,
  col = ex_colony,
  pch = 16,
  xlab = "Quality of institutions",
  ylab = "Human development index",
  main = "Effect of a binary variable"
  )

# the regression line when ex_colony = never colony
abline(model_coef[1], model_coef[2], col = 1, lwd = 2)

# the regression line when ex_colony = ex colony
abline(model_coef[1] + model_coef[3], model_coef[2], col = 2, lwd = 2)

# add a legend to the plot
legend(
  "bottomright",
  legend = levels(world_data$ex_colony),
  col = world_data$ex_colony,
  lwd = 2,   # line width = 1 for adding a line to the legend
  pch = 16,
  bty = "n"
  )
```



9 Regression Assumptions

9.1 Seminar

Clear the environment and set the working directory.

```
rm(list = ls())
setwd("Your directory")
```

9.1.1 Required Packages

Today, we need two new packages: `lmtest` and `sandwich`. We'll explain the functionality offered by each package as we go along. We load the the libraries up front and you need to install `lmtest` and `sandwich` once.

```
install.packages("lmtest")
install.packages("sandwich")
```

Now load the libraries.

```
library(lmtest)
library(sandwich)
library(texreg)
```

9.1.2 Omitted Variable Bias

- Spurious Correlations

We have already dealt with OVB at length, but it is worth going over another (silly) example here for revision purposes (we talked about this example in class as well).

9.1.2.1 [Ice Cream and Shark Attacks] Ice Cream and Shark Attacks

Does ice cream consumption increase the likelihood of being attacked by a shark? Let's explore a dataset of shark attacks, ice cream sales and monthly temperatures collected over 7 years to find out.

```
shark_attacks <- read.csv("shark_attacks.csv")
head(shark_attacks)
```

```
  Year Month SharkAttacks Temperature IceCreamSales
1 2008     1           25        11.9            76
2 2008     2           28        15.2            79
3 2008     3           32        17.2            91
4 2008     4           35        18.5            95
5 2008     5           38        19.4           103
6 2008     6           41        22.1           108
```

Run a linear model to see the effects of ice cream consumption on shark attacks.

```
model1 <- lm(SharkAttacks ~ IceCreamSales, data = shark_attacks)
screenreg(model1)
```

```
========================
                Model 1
------------------------
(Intercept)      4.35
                (5.24)
IceCreamSales    0.34 ***
                (0.06)
------------------------
R^2              0.29
Adj. R^2         0.28
Num. obs.       84
RMSE             7.17
========================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

Let's check pairwise correlation between ice cream consumption, average monthly temperatures and shark attacks.

```
cor(shark_attacks[, c("SharkAttacks", "Temperature", "IceCreamSales")])
```

```
              SharkAttacks Temperature IceCreamSales
SharkAttacks     1.0000000   0.7169660     0.5343576
Temperature      0.7169660   1.0000000     0.5957694
IceCreamSales    0.5343576   0.5957694     1.0000000
```

It looks like there's high correlation between average monthly temperatures and shark attacks, and also between ice cream consumption and monthly temperatures, so let's add the `Temperature` variable to our model.

```
model2 <- lm(SharkAttacks ~ IceCreamSales + Temperature, data = shark_attacks)
```

Now compare it to the first model.

```
screenreg(list(model1, model2))
```

```
====================================
               Model 1    Model 2
------------------------------------
(Intercept)     4.35       0.57
               (5.24)     (4.31)
IceCreamSales   0.34 ***   0.10
               (0.06)     (0.06)
Temperature                1.29 ***
                          (0.20)
------------------------------------
R^2             0.29       0.53
Adj. R^2        0.28       0.52
Num. obs.      84         84
RMSE            7.17       5.84
====================================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

The second model shows that once we control for monthly temperatures, ice cream consumption has no effect on the number of shark attacks.

What was the bias in the coefficient of `IceCreamSales` in the model where we did not control for `Temperature`?

```
# coefficient in new model minus coefficient in old model
0.10 -  0.34
```

```
[1] -0.24
```

We overestimated the effect by 0.24, more than two thirds of the effect. We know that the real effect of selling ice cream on shark attacks is exactly 0 (unless for some reason sharks really like icecream). The two phenomena are unrelated. Our estimate, although insignificant, is still an over-estimate.

9.1.3 Detecting non-linearity

For this part of the problem set, we will be using data on the results of the German Bundestag elections in 2017. You can load this data directly from GitHub using the following command:

```
german_results <- read.csv("https://raw.githubusercontent.com/UCLSPP/datasets/master/data/german_electi
```

Note, the data contains weird looking German characters. If you want them properly displayed, load the data using its character encoding utf-8 like so: `german_results <- read.csv("https://raw.githubusercontent.com/UCLSPP/d fileEncoding = "utf-8")`. It's usually not a problem because it only messes up the names of the districts.

We should again quickly see the variables that are included in the data using the `head()` function (or any other way you prefer):

```
head(german_results)
```

```
                          district      CDU      SPD      AfD Die.Linke
```

```
1             Flensburg â\200" Schleswig 39.99930 28.01454 6.214686  7.129180
2 Nordfriesland â\200" Dithmarschen Nord 45.14833 25.15210 5.881031  5.150071
3     Steinburg â\200" Dithmarschen SÃ¼d 41.87862 26.14625 7.642704  5.481995
4              Rendsburg-EckernfÃ¶rde 42.68043 28.87215 6.826306  5.172259
5                                Kiel 30.50011 31.05149 6.136603  7.376207
6               PlÃ¶n â\200" NeumÃ¼nster 40.69998 28.91278 7.834377  5.371334
      Grune       FDP unemploymentrate migrantfraction catholicfraction
1 10.512869  6.544773             7.16            10.0              5.1
2  9.453158  8.051428             7.05             8.0              5.2
3  6.716686 11.033518             6.30             9.2              4.2
4  8.954402  6.455395             5.18             8.5              4.9
5 14.403295  7.462318             9.65            18.0              7.3
6  8.993862  7.187579             6.72            10.8              5.6
```

For this problem, we are going to be interested in modelling the share of the vote won by the SPD in each of the 299 electoral districts. This variable is named `SPD` in the `german_results` data.frame. Let's build a linear regression model with two predictors: `migrantfraction` and `catholicfraction`, both of which are measured on a range from 0 to 100 percent.

```
# model on SPD
spd_linear_model <- lm(SPD ~ migrantfraction + catholicfraction , data = german_results)

# show model
screenreg(spd_linear_model)
```

```
============================
                  Model 1
----------------------------
(Intercept)        19.47 ***
                   (1.07)
migrantfraction     0.38 ***
                   (0.05)
catholicfraction   -0.06 **
                   (0.02)
----------------------------
R^2                 0.16
Adj. R^2            0.16
Num. obs.         299
RMSE                7.62
============================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

The model suggests that the fraction of migrants in a district is positively related to the popularity of the SPD in that district, and that there is a negative relationship between the fraction of catholics in a district and the SPD vote share. More precisely, the model tells us the for each additional percentage point of migrants, the SPD's vote share increases by 0.38 points, holding constant the fraction of Catholics. Conversely, for each additional percentage point of Catholics, the SPD's share of the vote increases by -0.06 points, holding constant the fraction of migrants.

As discussed in lecture, it is often useful to examine plots of the *model residuals* to determine whether there is evidence of a non-linear relationship between our dependent variable and our independent variables. Recall that the sample residuals are the difference between the observed values of Y and the predicted values of Y given our model: $u_i = Y_i - \hat{Y}_i$

We can calculate the residuals for each observation in our data that we used to estimate the model by applying the `residuals()` function to the model object. Let's do this now, and assign the output to be a new variable

in the `german_results` data frame.

```
# get the residuals
german_results$residuals <- spd_linear_model$residuals
```

Assume you wanted to estimate the residuals from the fitted values. Do so on your own. Hint: You can get fitted values (model predictions for each observation in the data) as `spd_linear_model$fitted.values`.

```
german_results$residuals2 <- german_results$SPD - spd_linear_model$fitted.values
```

Assess whehter both methods lead to the same result. Hint: There is rounding error, so both methods may not produce similar results numericaly. Confirm that the results are equivalent nonetheless.

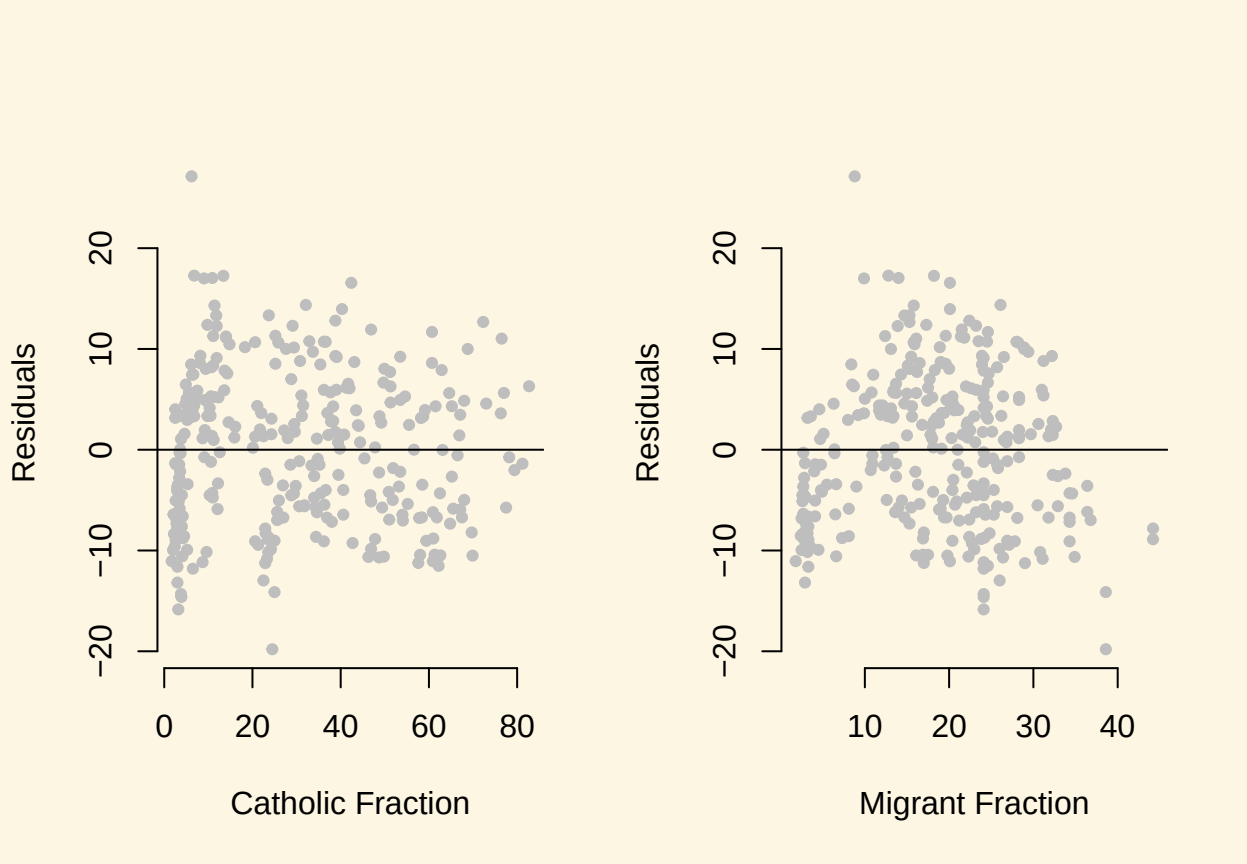
```
# correlation coefficient
cor(german_results$residuals, german_results$residuals2)
```

```
[1] 1
```

To assess whether there is evidence for non-linearity, it can be helpful to plot the residuals against the continuous explanatory variables in our model. Remember, the assumptions behind OLS models suggest that the residuals should be randomly distributed around zero for all values of X: $E(u_i|X_i) = 0$. So, any pattern that deviates from this randomness suggests that we may have misspecified our model.

Let's plot the residuals from `spd_linear_model` against the two independent variables in our data.

```
par(mfrow = c(1,2))
plot(y = german_results$residuals,
     x = german_results$catholicfraction,
     xlab = "Catholic Fraction",
     ylab = "Residuals",
     frame.plot = F,
     pch = 19,
     cex = .7,
     col = "grey")
abline(h = 0)

plot(y = german_results$residuals,
     x = german_results$migrantfraction,
     xlab = "Migrant Fraction",
     ylab = "Residuals",
     frame.plot = F,
     pch = 19,
     cex = .7,
     col = "grey")
abline(h = 0)
```



What can we conclude from these plots? In the left-hand plot, the residuals seem to be distributed fairly randomly around zero for all values of `catholicfraction`: there does not appear to be evidence of non-linearity for this variable.

In the right-hand plot, however, it seems that the residuals are mostly negative for both low and high levels of the `migrantfraction` variable, but evenly spread around zero for the more moderate range of this variable. This pattern of residuals does seem to deviate from the OLS assumptions to a certain degree, and therefore suggests that we might want to try to account for this possible non-linearity in our model.

One way of doing this is to include a polynomial term (we will use X^2 here) in our model, as we did last week. Let's do that now, and then `screenreg()` the results. Try on your own:

```
spd_non_linear_model <- lm(SPD ~ poly(migrantfraction,2) + catholicfraction , data = german_results)
screenreg(spd_non_linear_model)
```

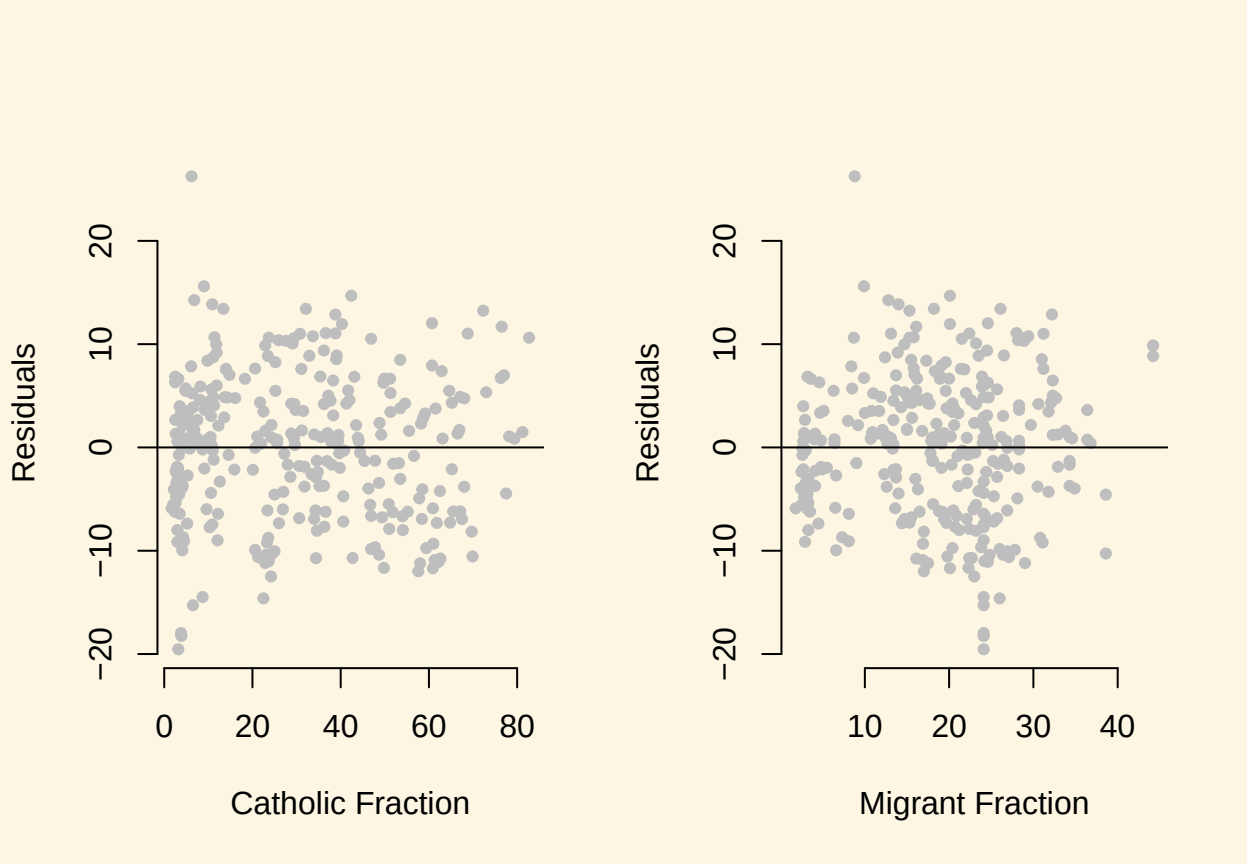
```
=====================================
                            Model 1
-------------------------------------
(Intercept)                  28.57 ***
                             (0.75)
poly(migrantfraction, 2)1    66.31 ***
                             (7.28)
poly(migrantfraction, 2)2   -58.19 ***
                             (7.71)
catholicfraction             -0.13 ***
                             (0.02)
-------------------------------------
```

```
R^2                          0.30
Adj. R^2                     0.29
Num. obs.                  299
RMSE                         6.99
=====================================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

The coefficient on the squared term (`poly(migrantfraction, 2)2`) is significant, which indicates that there is evidence of a non-linear relationship between `migrantfraction` and SPD vote share in our data. This suggests that our interpretation of the residual plot above was correct, and that including the squared term is an important modification to our model. (You might also note that the Adjusted R^2 has nearly doubled with the inclusion of X^2, again suggesting that the non-linear adjustment is important.)

We can now repeat the same residual-plotting exersise with the residuals from our new model, to see if we are closer to meeting the OLS assumption that the error term is randomly distributed around zero for all values of X. Try on your own:

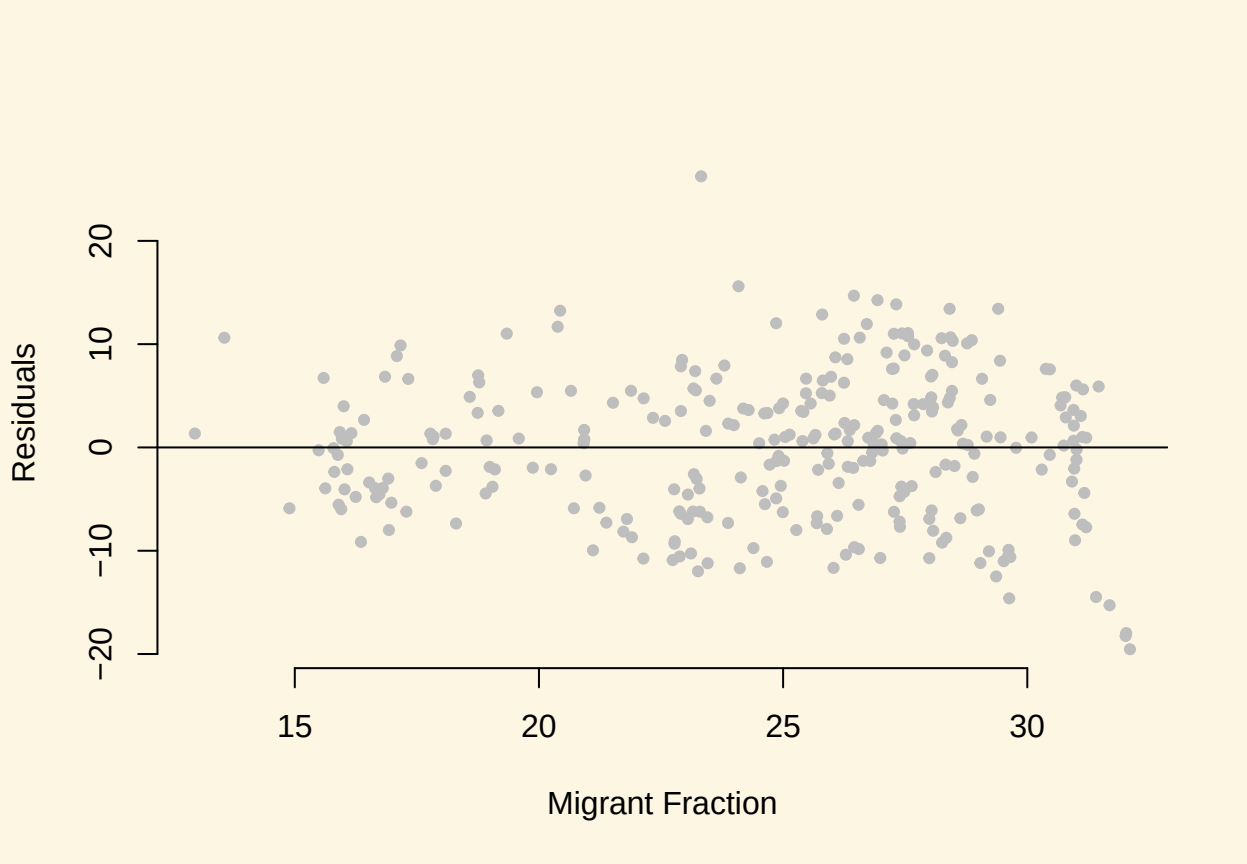
```
german_results$residuals_model_two <- residuals(spd_non_linear_model)

par(mfrow = c(1,2))
plot(y = german_results$residuals_model_two,
     x = german_results$catholicfraction,
     xlab = "Catholic Fraction",
     ylab = "Residuals",
     frame.plot = F,
     pch = 19,
     cex = .7,
     col = "grey")
abline(h = 0)

plot(y = german_results$residuals_model_two,
     x = german_results$migrantfraction,
     xlab = "Migrant Fraction",
     ylab = "Residuals",
     frame.plot = F,
     pch = 19,
     cex = .7,
     col = "grey")
abline(h = 0)
```



The right-hand plot now looks much better: the residuals seem randomly distributed around zero for all values of the `migrantfraction` variable, and there is no clearly distinguishable pattern anymore.

You can also inspect how the model does overall. Repeat the previous exercise but plot the fitted values on the x-axis. What can we conclude?

```
par(mfrow = c(1,1))
plot(y = german_results$residuals_model_two,
     x = spd_non_linear_model$fitted.values,
     xlab = "Migrant Fraction",
     ylab = "Residuals",
     frame.plot = F,
     pch = 19,
     cex = .7,
     col = "grey")
abline(h = 0)
```



Overall, it looks like conditional mean independence is satisfied.

9.1.4 Heteroskedasticity

In order to understand heteroskedasticity, let's start by loading a sample of the U.S. Current Population Survey (CPS) from 2013. The dataset contains 2989 observations of full-time workers with variables including age, gender, years of education and income reported in hourly earnings.

```
cps <- read.csv("cps2013.csv")
```

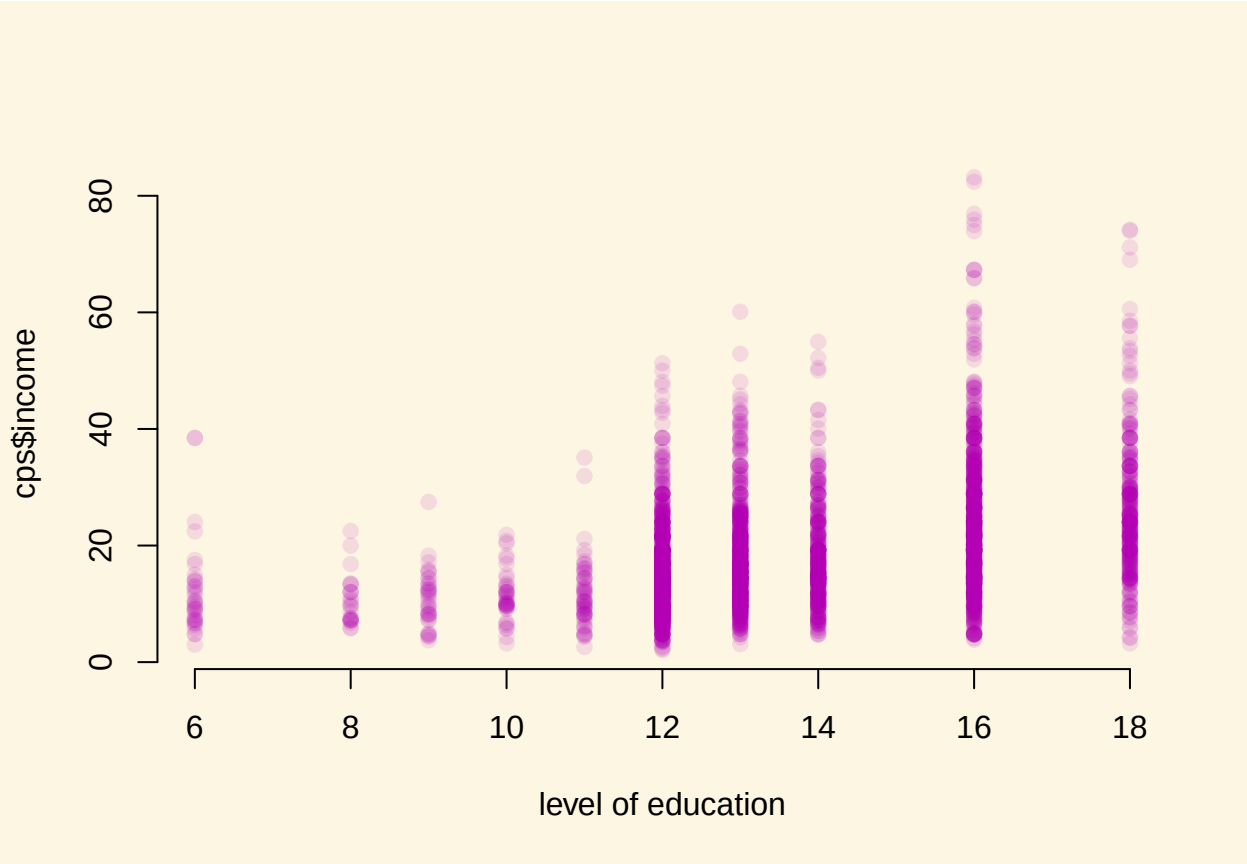
```
head(cps)
```

```
  age gender education    income
1  30   Male        13 12.019231
2  30   Male        12  9.857142
3  30   Male        12  8.241758
4  29 Female        12 10.096154
5  30   Male        14 24.038462
6  30   Male        16 23.668638
```

We plot income by years of education.

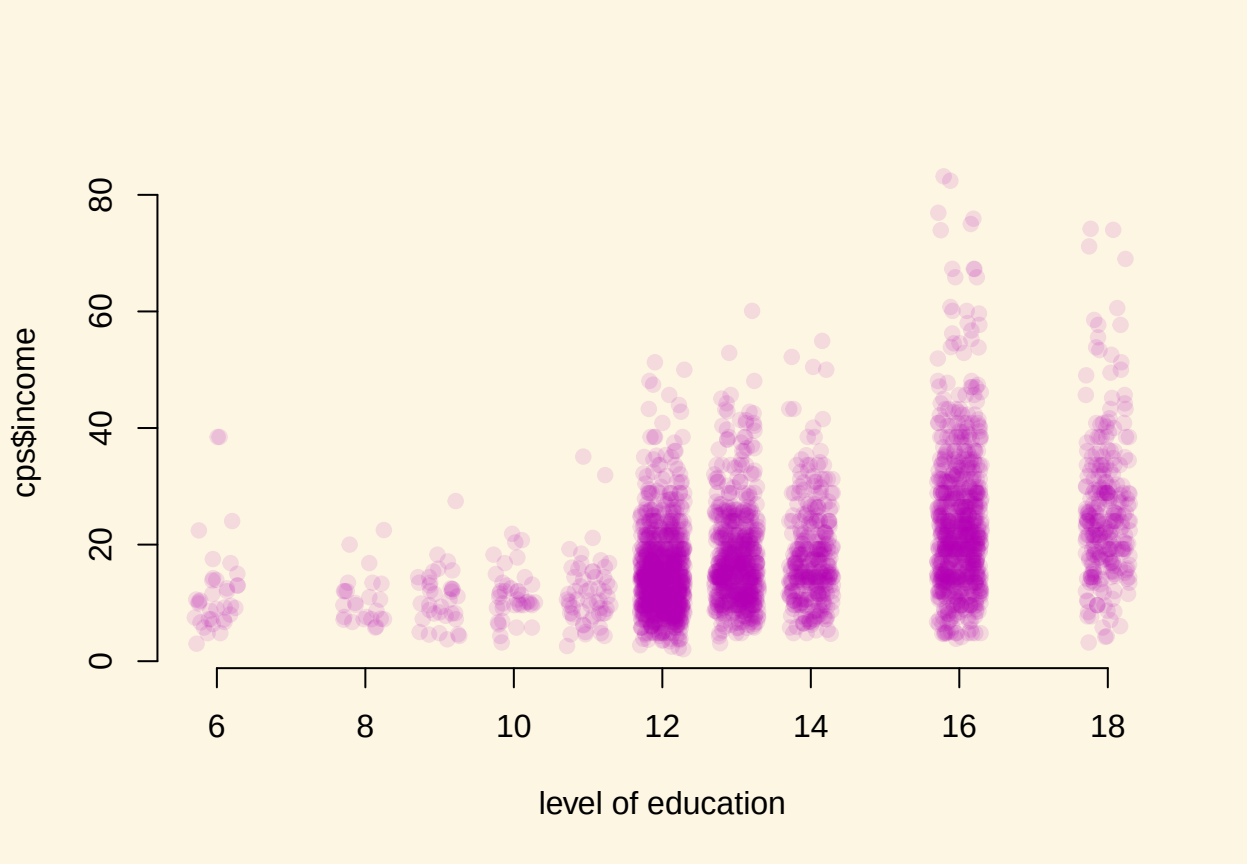
```
plot(
  y = cps$income,
  x = cps$education,
  xlab = "level of education",
  pch = 19,
```

```
  cex = 1,
  bty = "n",
  col = rgb(180,0,180,30, maxColorValue = 255)
  )
```



The dots in this plot largely overlap because education is only collected for full years. It's therefore a bit hard to see the amount of observations that have 12 years of education and income 20. You can make it easier to see the amount of points by adding sine random noise to the education variable. So for example, for everyone with twelve years of eductation, we add or subtract a small amount at random. We will still be able to tell the education categories apart. We use this `jitter()` function to do this. We choose a small number of noise to add such that we can see the number of observations better and education categories are still visible. You have to play around to find a good number. Try on your own.

```
plot(
  y = cps$income,
  x = jitter(cps$education, 1.5),
  xlab = "level of education",
  pch = 19,
  cex = 1,
  bty = "n",
  col = rgb(180,0,180,30, maxColorValue = 255)
  )
```



We can see that the range of values for income have a larger spread as the level of education increases. Intuitively this makes sense as people with more education tend to have more opportunities and are employed in a wider range of professions. But how does this larger spread affect the regression model?

Let's run linear regression and take a look at the results to find out the answer.

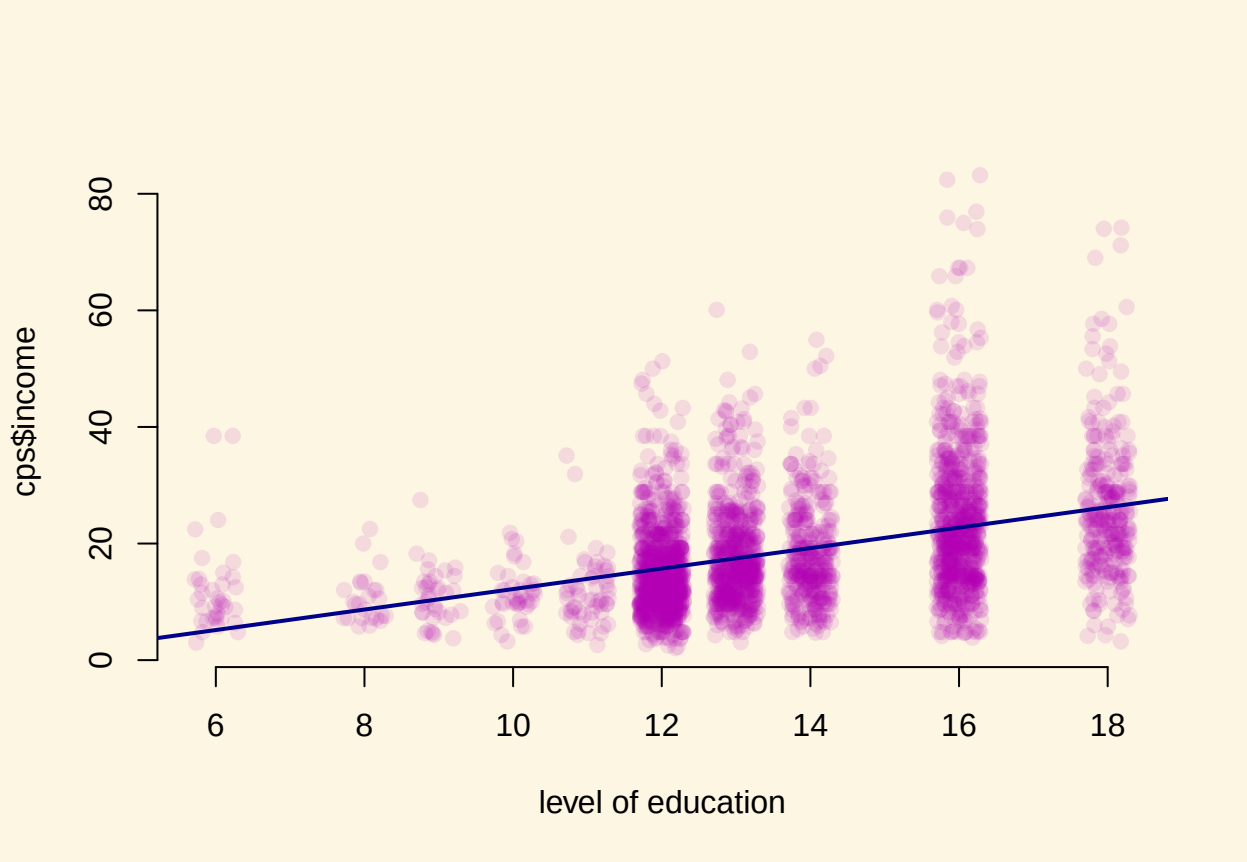
```
model1 <- lm(income ~ education, data = cps)
summary(model1)
```

```
Call:
lm(formula = income ~ education, data = cps)

Residuals:
    Min      1Q  Median      3Q     Max
-23.035  -6.086  -1.573   3.900  60.446

Coefficients:
            Estimate Std. Error t value             Pr(>|t|)
(Intercept) -5.37508    1.03258  -5.206          0.000000207 ***
education    1.75638    0.07397  23.745 < 0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.496 on 2987 degrees of freedom
Multiple R-squared:  0.1588,    Adjusted R-squared:  0.1585
F-statistic: 563.8 on 1 and 2987 DF,  p-value: < 0.00000000000000022
```

The model tells us the for each additional year of education, the income increases by 1.756.

Now let's plot the fitted model on our scatter plot of observations.

```
plot(
  y = cps$income,
  x = jitter(cps$education,1.5),
  xlab = "level of education",
  pch = 19,
  cex = 1,
  bty = "n",
  col = rgb(180,0,180,30, maxColorValue = 255)
  )
abline(model1, col = "darkblue", lwd = 2)
```

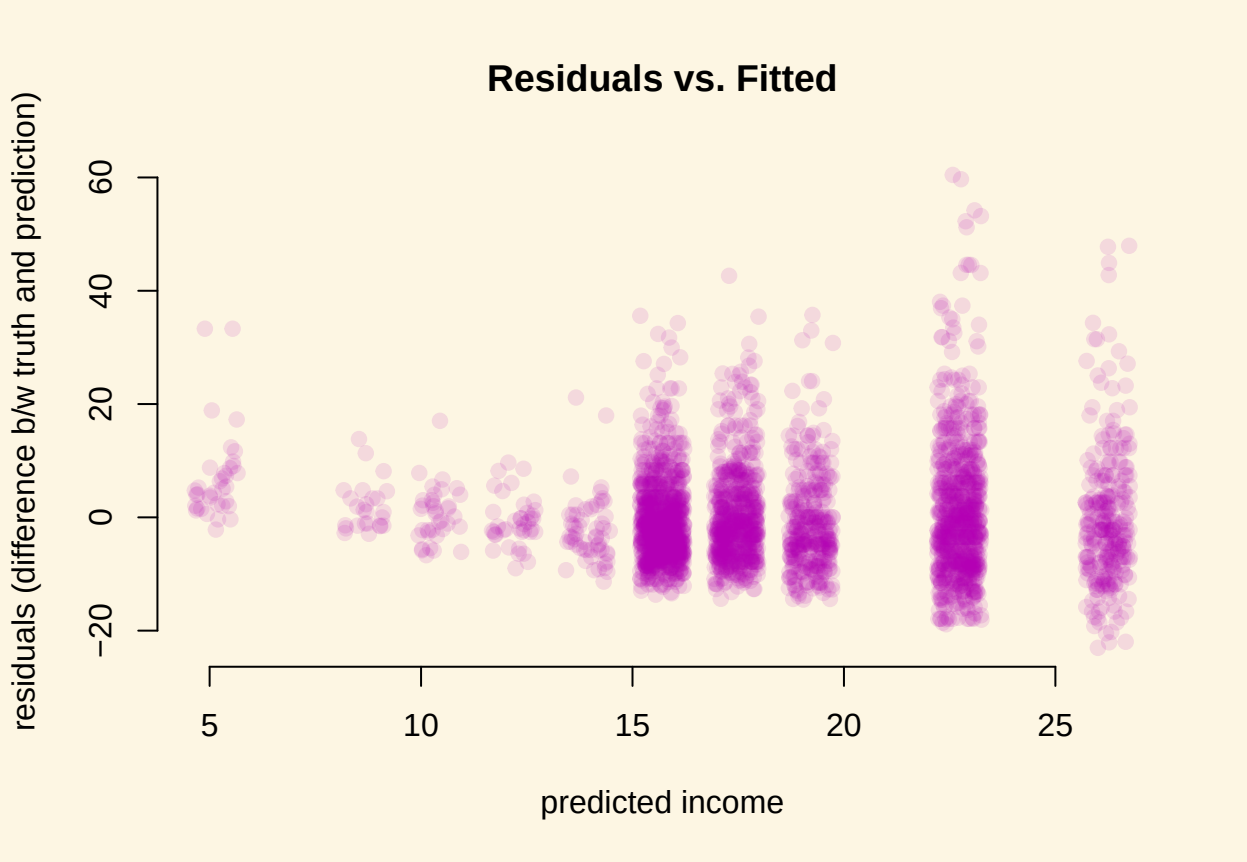


Looking at the fitted model, we can see that the errors (i.e. the differences between the predicted value and the observed value) increase as the level of education increases. This is what is known as *heteroskedasticity* or *heteroskedastic errors*. In plain language it simply means that the variability of the error term is not constant across the entire range of our observations.

Another way to visualize this is by plotting residuals against the fitted values.

```
plot(
  y = model1$residuals,
  x = jitter(model1$fitted.values, 1.5),
  pch = 19,
  bty = "n",
  xlab = "predicted income",
```

```
  ylab = "residuals (difference b/w truth and prediction)",
  col = rgb(180,0,180,30, maxColorValue = 255),
  main = "Residuals vs. Fitted")
```



Again, we can see that residuals are not constant and increase as the fitted values increase. In addition to visual inspection, we can also test for heteroskedasticity using the Breusch-Pagan Test from the `lmtest` package we loaded at the beginning.

```
bptest(model1)
```

```
    studentized Breusch-Pagan test

data:  model1
BP = 76.679, df = 1, p-value < 0.00000000000000022
```

The null hypothesis for the Breusch-Pagan test is that the variance of the error term is constant, or in other words, the errors are homoskedestic. By looking at the *p-value* from Breusch-Pagan test we can determine whether we have heteroskedastic errors or not.

The *p-value* tells us that we can reject the null hypothesis of homoskedestic errors. Once we've determined that we're dealing with heteroskedastic errors, we can correct them using the `coeftest()` function from the `lmtest` package.

Here is a list of arguments for the `coeftest()` function:

```
coeftest(model, vcov)
```

Argument	Description
`model`	This is the estimated model obtained from `lm()`.
`vcov`	Covariance matrix. The simplest way to obtain heteroskedasticity-consistent covariance matrix is by using the `vcovHC()` function from the `sandwich` package. The result from `vcovHC()` can be directly passed to `coeftest()` as the second argument.

```
screenreg(coeftest(model1, vcov = vcovHC(model1)))
```

```
======================
             Model 1
----------------------
(Intercept)  -5.38 ***
             (1.05)
education     1.76 ***
             (0.08)
======================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

After correcting for heteroskedasticity we can see that the standard error for the independent variable `education` have increased from 0.074 to 0.08. Even though this increase might seem very small, remember that it is relative to the scale of the independent variable. Since the standard error is the measure of precision for the esitmated coefficients, an increase in standard error means that our original estiamte wasn't as good as we thought it was before we corrected for heteroskedasticity.

Now that we can get heteroskedastic corrected standard errors, how would you present them in a publication (or in your dissertation or the final exam)? Fortunately, all `textreg` functions such as `screenreg()` or `htmlreg()` allow us to easily override the *standard errors* and *p-value* in the resulting output.

We first need to save the result from `coeftest()` to an object and then override the output from `screenreg()` by extracting the two columns of interest.

The corrected *standard errors* are in column `2` of the object returned by `coeftest()` and the associated *p-values* are in column `4`:

```
corrected_errors <- coeftest(model1, vcov = vcovHC(model1))

screenreg(model1,
          override.se = corrected_errors[, 2],
          override.pval = corrected_errors[, 4])
```

```
========================
             Model 1
------------------------
(Intercept)    -5.38 ***
               (1.05)
education       1.76 ***
               (0.08)
------------------------
R^2             0.16
Adj. R^2        0.16
Num. obs.    2989
RMSE            9.50
========================
*** p < 0.001, ** p < 0.01, * p < 0.05
```

9.1.5 Exercises

Does Indiscriminate Violence Incite Insurgent Attacks? Download and then load `chechen.csv`.

chechen data

Jason Lyall:

Does a state's use of indiscriminate violence incite insurgent attacks? To date, most existing theories and empirical studies have concluded that such violence is highly counterproductive because it creates new grievances while forcing victims to seek security, if not safety, in rebel arms. This proposition is tested using Russian artillery fire in Chechnya (2000 to 2005) to estimate indiscriminate violence's effect on subsequent patterns of insurgent attacks [. . .].

Codebook:

Variable	Description
vilno	unique village number
disno	unique distric number
treat	Russian artillery shelling of cechan village
poverty	estimated need (3 is highest)
pop2000	population in 2000
lpop2000	logged population in 2000
pret	pre-treatment insurgent attacks
post	post-treatment insurgent attacks
diff	post minus pret
groznyy	1 if in groznyy
elevation	in meters
lelev	logged elevation
tariq	Tariqa (1=Naq.)
nn	distance to nearest neighbour (km)
lnn	logged distance to nearest neighbour
reb	rebel (1 = Basayev)
garrison	(1 = Russian base present)
vildays	days to first insurgent attack after treatment
deaths	estimated number of individuals killed during Russian artillery shelling
deathsd	1 indicates that individuals were killed during shelling
wounded	estimated number of individuals wounded during Russian artillery shelling
woundedd	1 indicates that individuals were wounded during Russian artillery shelling
fail	1 if insurgent attack is observed in 90 days after artillery shelling
swept	1 indicates village was swept in 90 days prior to artillery shelling
prop	estimated number of structures destroyed during shelling
propd	1 indicates that property was destroyed during shelling
lprop	logged number of estimated structures destroyed or damaged during shelling
history	number of times village had been shelled in the past
historyd	1 indicates that village had been shelled in the past
lhistory	logged number of times village had been shelled
change	1 if DD is increase: 0 if no change, -1 if DD decreases
decrease	1 if decrease recorded, 0 if not

Variable	Description
increase	1 if increase in DD recorded, 0 if not
abandon	number of neighbours within 5km^2 radius

The dataset comes from an article in the Journal of Conflict Resolution. Use the codebook above for the tasks and if anything is unclear, refer to the article that we have linked to above.

1. Load `chechen.csv`.
2. Run a bivariate regression model with `diff` as the dependent variable and `treat` as the predictor.
3. Control for additional variables. Discuss your choice of variables and refer to omitted variable bias.
4. Discuss the improvement in model fit and use the appropriate test statistic.
5. Check for non-linearity and if there is solve the problem.
6. Check for heteroskedasticity and if there is solve the problem.
7. Thoroughly compare the models, and interpret the outcome in substantial as well statistical terms - do not forget to talk about model fit. What are the implications of this?

Ordinary Economic Voting Behavior in the Extraordinary Election of Adolf Hitler Download and then load `who_voted_nazi_in_1932.csv`.

who voted nazi in 1932 data

The goal of analysis is to investigate which types of voters (based on their occupation category) cast ballots for the Nazis. One hypothesis says that the Nazis received much support from blue-collar workers. Since the data do not directly tell us how many blue-collar workers voted for the Nazis, we must infer this information using a statistical analysis [. . .].

The data contains aggregate voting behaviour from 681 voting districts.

Codebook:

Variable	Description
sharenazis	Percent of the vote Nazis received in the district
nazivote	Number of Nazi votes
nvoter	Total number of eligible voters
shareblue	Percent of blue-collar potential voters
sharewhite	Percent of white-collar potential voters
shareself	Percent of self-employed potential voters
sharedomestic	Percent of domestically employed potential voters
shareunemployed	Percent of unemployed potential voters

8. Load `who_voted_nazi_in_1932.csv`.
9. Run a bivariate model to test whether blue-collar voters voted for the Nazis.
10. Control for additional variables. Discuss your choice of variables and refer to omitted variable bias.
11. Can you fit a model using `shareblue`, `sharewhite`, `shareself`, `sharedomestic`, and `shareunemployed` as predictors? Why or why not?
12. Check for heteroskedasticity and if there is solve the problem.
13. Interpret your models.

Load the Massachusetts Test Scores dataset `MA_Schools.csv` from your PUBLG100 folder.

The dataset contains `16` variables, but we're only interested in the following:

Variable	Description
score8	8th grade scores
stratio	Student-teacher ratio

Variable	Description
english	% English learners
lunch	% Receiving lunch subsidy
income	Average district income

14. Fit a model to explain the relationship between percentage of English learners and average 8th grade scores.
15. Plot the model and the regression line.
16. Check for correlation between the variables listed above to see if there are other variables that should be included in the model.
    - HINT: There are some missing values in the dataset. Remove the NA's from the dataset like we have seen previously.
    a. Add two variables from the list above that are highly correlated with both the independent and the dependent variable and run a linear regression again.
17. Compare the two models. Did the first model suffer from omitted variable bias?
18. Plot the residuals against the fitted values from the second model to visually check for heteroskedasticity.
19. Run the Breusch-Pagan Test to see whether the model suffers from heteroskedastic errors.
20. Correct for heteroskedasticity (if needed) and present the results with corrected standard errors and p-values.
21. Save everything and source your script. If you get error messages, clean your script.