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.

Syllabus

Moodle

Piazza

Chapter 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 (`/`).

Figure 1.1:

```
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?

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
```

Figure 1.2:

Figure 1.3:

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 wrote `by = 5` as the third argument. It is a common practice to specify the name of argument when the argument is optional. The arguments `from` and `to` are not optional, se we can write `seq(0, 100, by = 5)` instead of `seq(from = 0, to = 100, by = 5)`. Both, are valid ways of achieving the same outcome. You can code whichever way you like. We recommend to write code such that you make it easy for your future self and others to read and understand the code.

```
seq(from = 0, to = 100, by = 5)
```

```
 [1]   0   5  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80
[18]  85  90  95 100
```

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
```

Figure 1.4:

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

```
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' (0).

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:

Income Distributions in the UK and in France
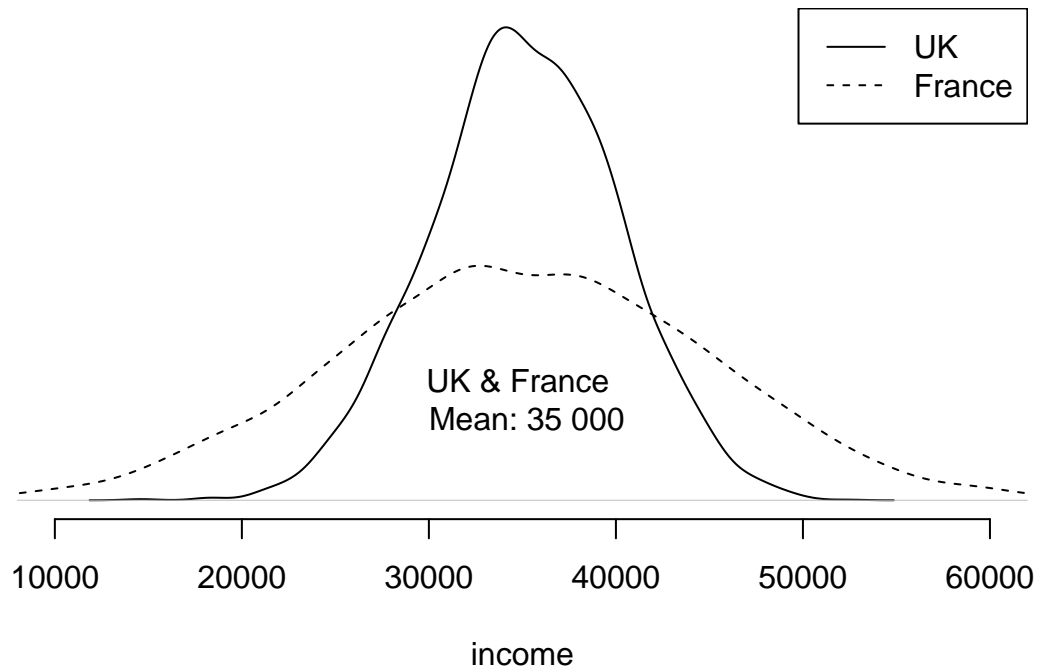
The variance gives us an idea about the variability of data. The formula for the variance in the population is

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

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
$\sqrt{}$			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.