Introduction to R and RStudio

Luigi Maglanoc and Anne Schad Bergsaker

March 2023

Before we begin...

Learning goals

1. Know the different windows in RStudio, and what the can be used for
2. Know how to write your own data into a data frame, and know how to combine single variables into a data set
3. Know how to import data regardless of file formats
4. Know how to efficiently document everything you do in R/Rstudio, and how to re-use your own procedures.
5. Manipulate data (recode variables, calculate new variables from raw data, etc.)
6. Present data using graphs made in RStudio

Technical prerequisites

If you do not have R and RStudio installed on your own device, go to view.uio.no and open RStudio.

Data that we use for examples can be downloaded as a .zip file from the course website. The file gapminder.txt is from gapminder.org, via the Software Carpentry website.

Try to do everything I do during the course. If you have questions or problems, just shout out!

Introduction to R and RStudio

R and Rstudio

What is it used for?

- Data analysis

Code

- Functions

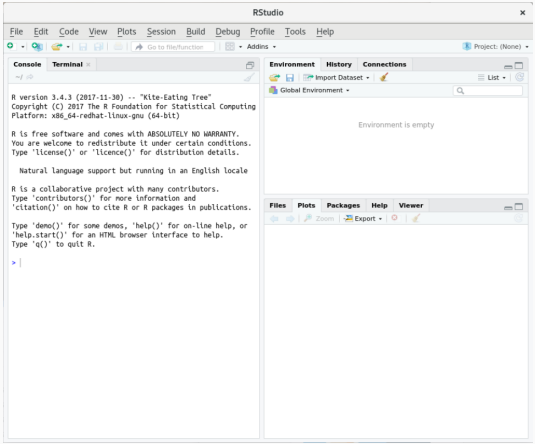
Why use R?

- Open source
- Reproducable analyses
- Many "packages"
- Data visualization
- Well documented

When you open RStudio

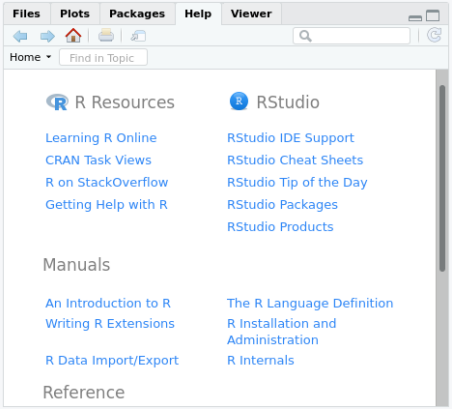
R is the actual software that does all the calculations and such, RStudio is the interface where you work. You must have both. RStudio without R is about as useful as Chrome or Firefox without an Internet connection.

When you open RStudio



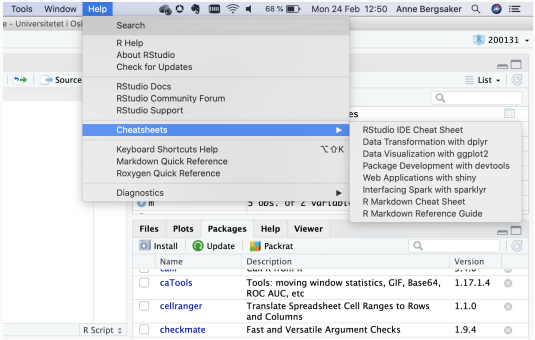
The first time you open RStudio, it looks like this. On the left hand side you can do calculations and write commands. On the right hand side you have several options. The most useful (in my opinion) are *Environment* in the top right panel, and *Help*, *Plots* and *Packages* in the bottom right panels.

THE BUILT IN HELP FUNCTION



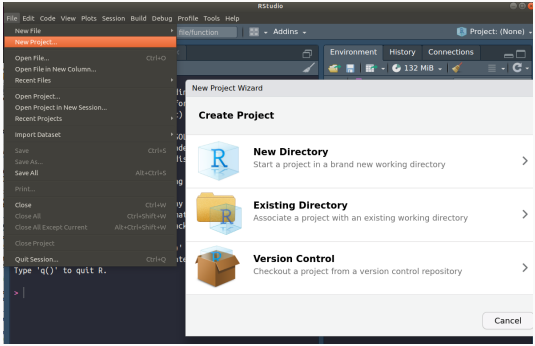
Here you can search for functions and commands. If you have no idea what you are looking for, or what you should do, it is usually better to do a google search instead.

CHEAT SHEETS



Some topics have built in cheat sheets, with lists of typical commands/functions and what they do. Examples include plotting with the ggplot library and data handling with the dplyr library.

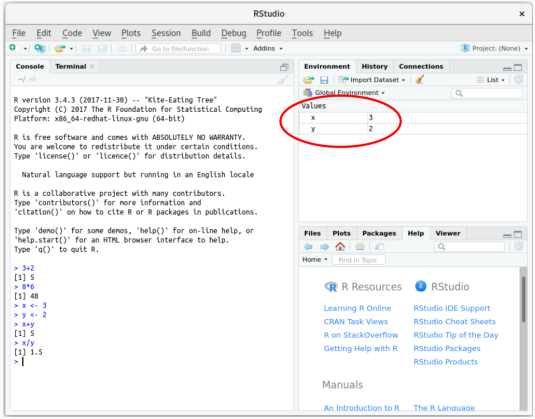
RSTUDIO PROJECTS



To make life a little easier, it is useful to tell R where our files will be found. There are seveeral ways of doing this, but the most reproducible method is to create and open an Rstudio "project".

Using the console

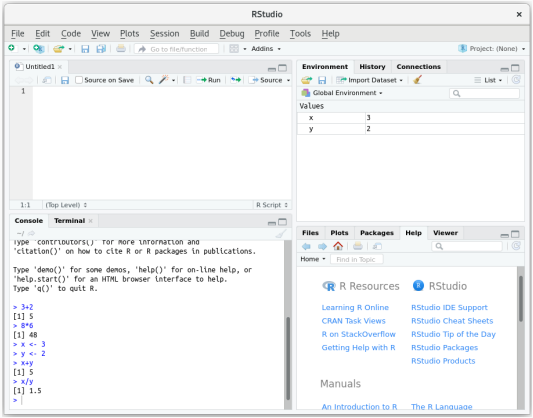
The console works a bit like a calculator, only it can do much more. Variables are defined by using the symbols $<-$, i.e. x $<-$ 5 means that the variable x gets the value 5. Make sure that there is no space between the $<$ and the $-$. We can do simple calculations with variables, just like with numbers. Notice that the variables you define show up in a list in *Environment* in the top right panel.



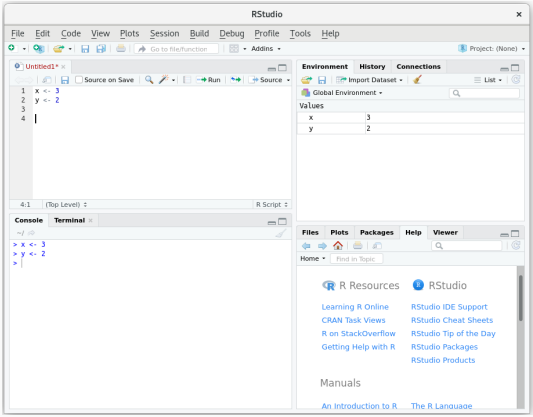
Opening a new script

File > New File > R script (Ctrl+Shift+N)

The script is where we should do most of our work. Here you can write commands, comments, etc. The script can be saved as a text document. This makes it easy to go back and check what we have done, and to do the same things over again, without the need to type the commands again.

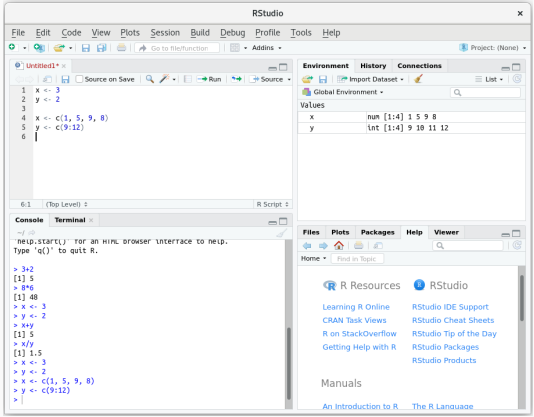


Using script instead of the console



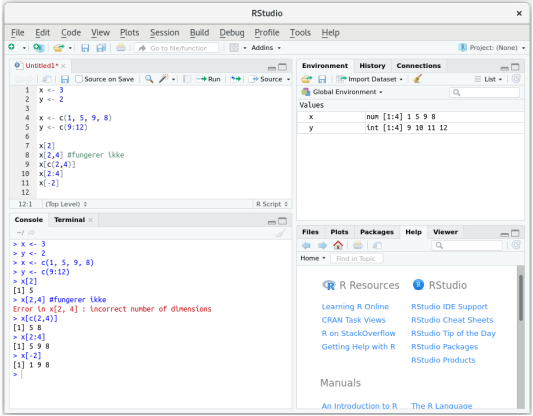
Anything you do in the console can also be done from the script. To run your commands, press Ctrl+Enter (Cmd+Enter on Mac) to run it. If you only press Enter, nothing happens. Ctrl+Alt+R and Ctrl+Alt+Enter runs the entire script from beginning to end. All commands you run are printed in the console.

How data is represented in R



Usually we have more than one measurement per variable. Using the function c(...), we can make variables containing many values. These are usually referred to as vectors. *c* is short for *combine*, i.e. combine elements into one.

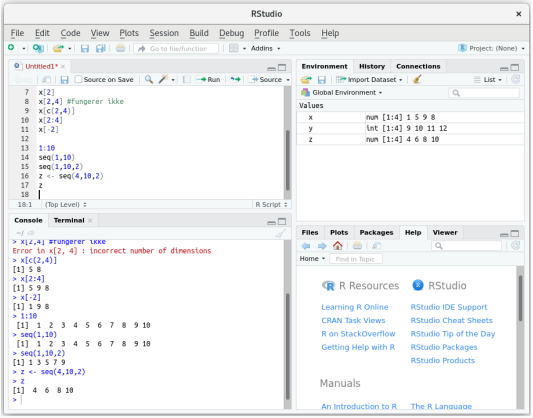
How data is represented in R



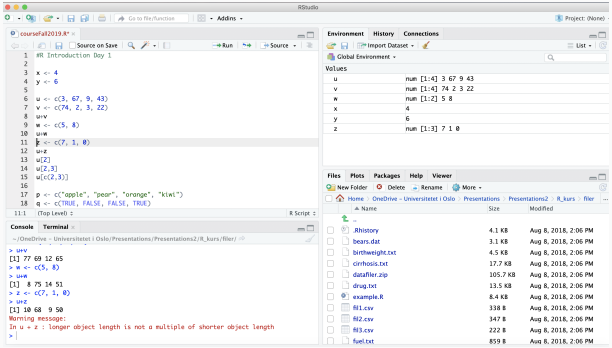
You can select single values form a variable by writing x[i] (selects value no i). You have to use square brackets. Select several values by using c(...) to combine values you want to select, i.e. write x[c(i, j, k)]. Select all values except i by writing x[$-i$].

How data is represented in R

Get R to make sequences of numbers for you using : e.g. 1:5 is the same as 1, 2, 3, 4, 5. seq(i, j, k) will create a sequence starting at i, ending at j with k as the interval between values. If you don't give k a value, the default is 1. seq(1, 7, 2) will give 1, 3, 5, 7, seq(2,6,3) gives 2, 5. rep(1,2) gives a sequence 1,1. rep(c(1,2),2) gives 1,2,1,2, and rep(c(1,2), each=2) gives 1,1,2,2.



How data is represented in R



If you add two vectors together, R will add the first item in one vector to the first in the other vector, working through both vectors. If one vector is shorter than the other, the shorter vector is recycled. Note that the length of the longer vector must be divisible by the length of the shorter vector.

Different types of variables

- Character - Words or single letters. Usually such a variable has to be recoded into a factor (see below).

Different types of variables

- Character - Words or single letters. Usually such a variable has to be recoded into a factor (see below).
- Date - R can handle dates as long as you tell R what kind of format the dates have, e.g. 30/01/18 or 1. jan 2015.

Different types of variables

- Character - Words or single letters. Usually such a variable has to be recoded into a factor (see below).
- Date - R can handle dates as long as you tell R what kind of format the dates have, e.g. 30/01/18 or 1. jan 2015.
- Factor - Words or single letters that R interprets as categories, e.g. gender (male, female), nationality (Norwegian, English, French,...). Often called a categorical variable

Different types of variables

- Character - Words or single letters. Usually such a variable has to be recoded into a factor (see below).
- Date - R can handle dates as long as you tell R what kind of format the dates have, e.g. 30/01/18 or 1. jan 2015.
- Factor - Words or single letters that R interprets as categories, e.g. gender (male, female), nationality (Norwegian, English, French,...). Often called a categorical variable
- Ordered factor - Word or single letters that R interprets as categories which can be ordered in a meaningful order, e.g. age categories, education level

Different types of variables

- Character - Words or single letters. Usually such a variable has to be recoded into a factor (see below).
- Date - R can handle dates as long as you tell R what kind of format the dates have, e.g. 30/01/18 or 1. jan 2015.
- Factor - Words or single letters that R interprets as categories, e.g. gender (male, female), nationality (Norwegian, English, French,...). Often called a categorical variable
- Ordered factor - Word or single letters that R interprets as categories which can be ordered in a meaningful order, e.g. age categories, education level
- Numerical - number, e.g. age, blood pressure, salary

Different types of variables

- Character - Words or single letters. Usually such a variable has to be recoded into a factor (see below).
- Date - R can handle dates as long as you tell R what kind of format the dates have, e.g. 30/01/18 or 1. jan 2015.
- Factor - Words or single letters that R interprets as categories, e.g. gender (male, female), nationality (Norwegian, English, French,...). Often called a categorical variable
- Ordered factor - Word or single letters that R interprets as categories which can be ordered in a meaningful order, e.g. age categories, education level
- Numerical - number, e.g. age, blood pressure, salary
- Logical - TRUE or FALSE, represented as 1 and 0.

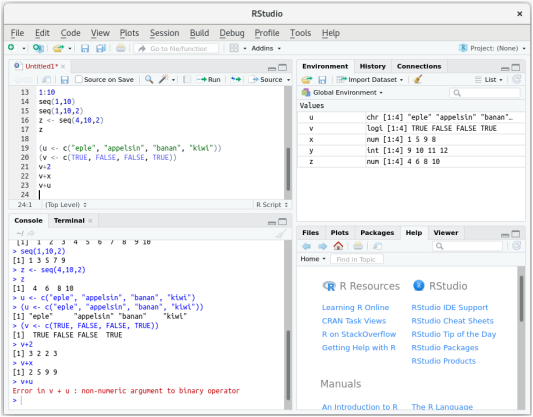
Changing the variable type

The different variable types have different properties. We can change type using functions such as *as.* $<$ *new type* $>$, e.g. *as.numeric*, *as.factor*, *as.character*, and so on.

If you are not sure which type your variable is, you can "ask" R, using functions such as *is.numeric*, *is.character*, etc.

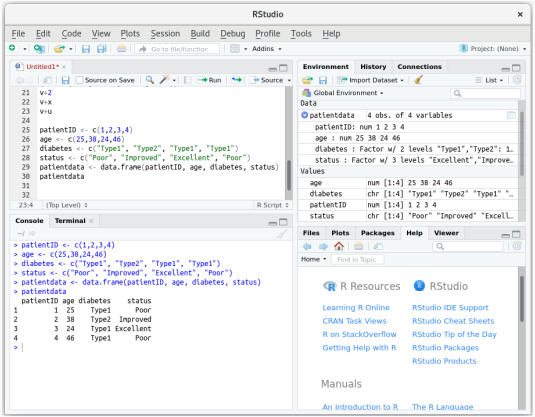
How data is represented in R

Logical statements are represented as 0 and 1, where 0=FALSE and 1=TRUE. Numerical and logical variables can be combined, but numerical and logical variables cannot be combined with factor variables.

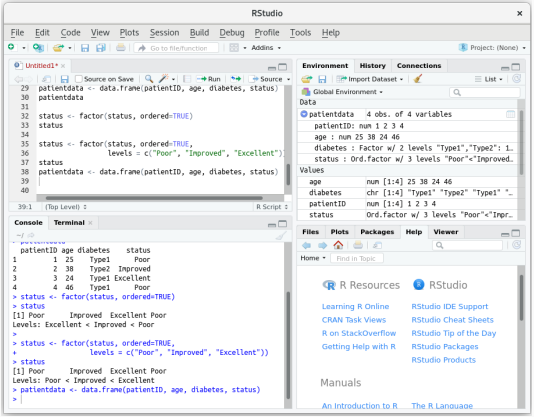


From single variables to a data set

We can combine variables to a data set by using the command data.frame(variable1, variable2, ...). A data frame is more or less the same as a spreadsheet. Notice that the variables *status* and *diabetes* have been converted to *factor* automatically.

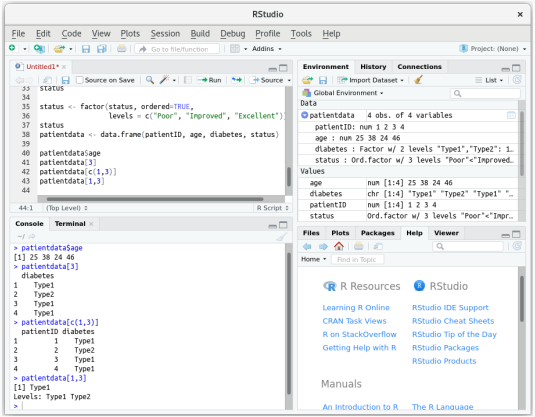


FACTOR VARIABLES



R often understands that a variable consisting of words is a factor, but it will not know if it is ordered. This must be coded manually, as must the order. If the actual order is not given, R will order the categories alphabetically.

SELECTING VARIABLES FROM A DATA SET



You can select single variables from a data set by using $, by writing data$variable, e.g. patientdata$age. You can also use patientdata[2], or if you wish to select more than one variable, patientdata[c(2,4)]. patientdata[1,3] will give you the value of variable 3, row 1.

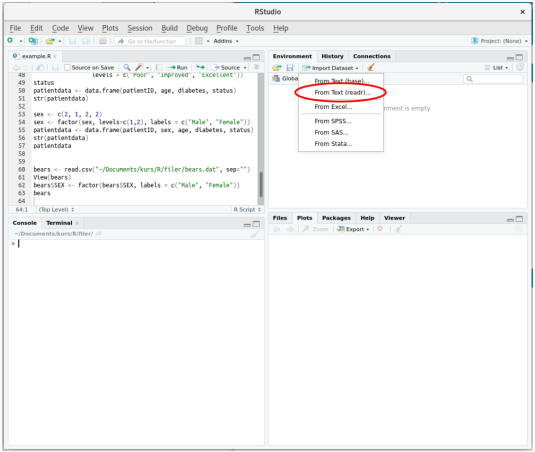
Exercises

We are creating a data set with four cases, each representing a person and how they respond to a type of treatment.

1. Create a variable containing the numbers 1-4 in ascending order, using the *seq* function. Call the variable *ID*.
2. Create a variable containing the numbers 2,2,1,1 in that order, using the *rep* function. Call the variable *group*.
3. Create a variable called *response*, where possible categories are *poor*, *medium* and *good*. You can decide how each person responds as you want, but remember that this variable must be changed into an ordered factor.
4. Combine all variables into a data frame
5. Print case no. 2, then variable 3, and finally the value of variable 3 only for case no. 2.
6. Print all cases except case no 1.

IMPORT/OPEN DATA FILES

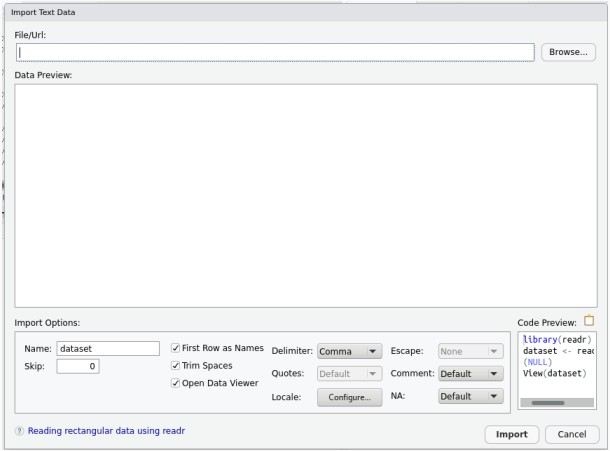
Importing a data file



To import a data file, click Import Dataset > From Text (readr)/From CSV. This can be used to open/import .txt, .csv, .tab, .dat and similar file formats. You can also use the option From Text (base).

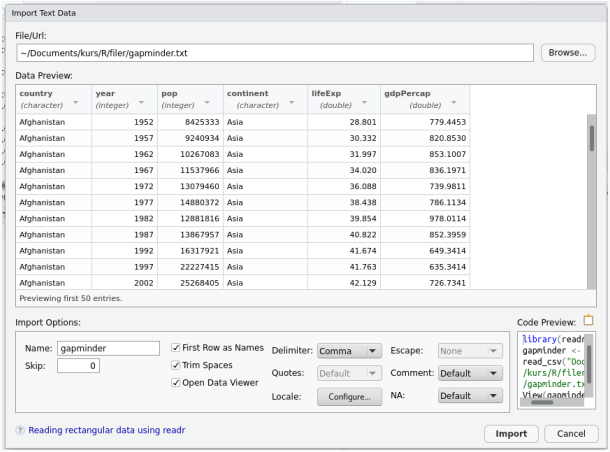
Importing a data file

Click *Browse* and select the appropriate file. The file should be a simple text file.



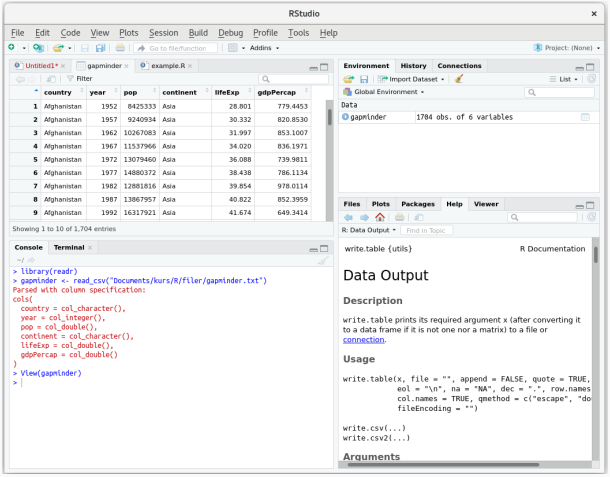
Importing a data file

In *Data Preview* you will see the data from the file. Check that it looks right. If it doesn't, you may have to change the default options under the preview window, like change the delimiter.



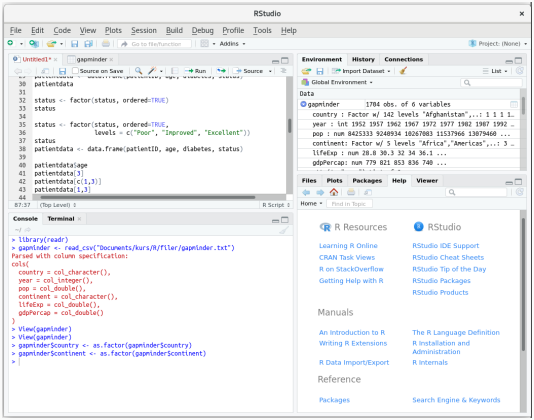
Importing a data file

You should copy the command lines that show up in the console (only the blue ones, not the red). That way, the next time you work on this data set, all you have to do is re-run the same few lines of code, rather than go through the dialogue box again.



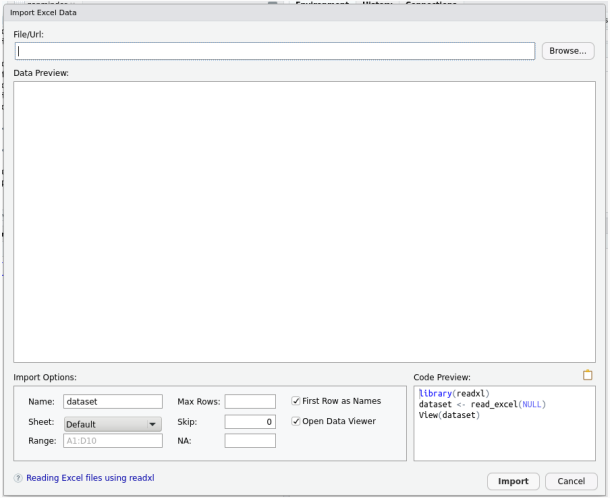
Importing a data file

Factor variables are not necessarily imported correctly, and may need to be converted to factors. You can check this in the Environment window. If they are designated as Char variables, they need to be converted. Use variabel <- as.factor(variabel). If you use Import Dataset From Text (base) instead, factors tend to be converted to factors automatically.

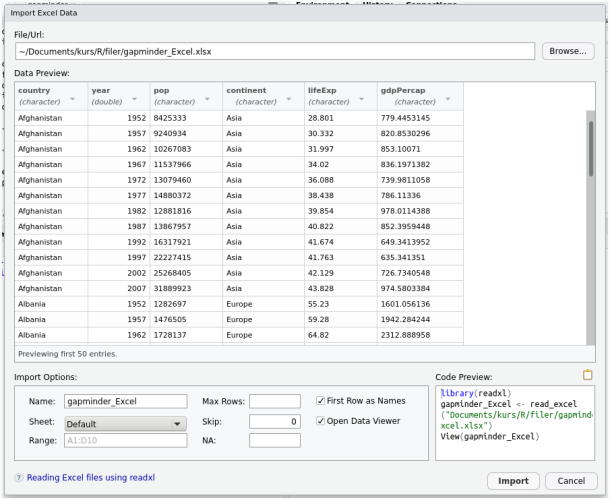


Importing a data file - xls/xlsx

To import data from Excel, select Import Dataset > From Excel



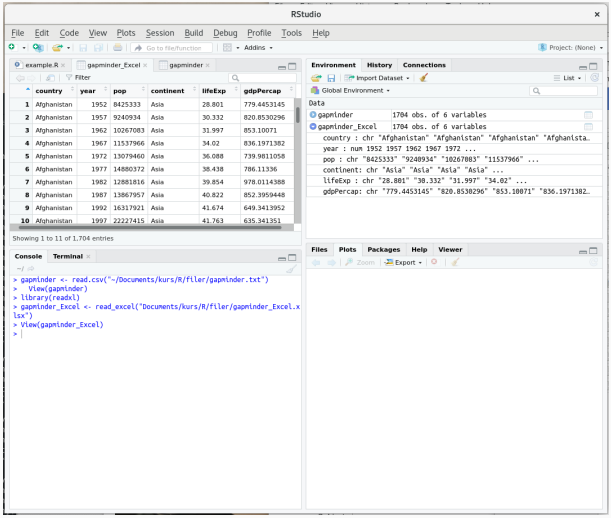
Importing a data file - xls/xlsx



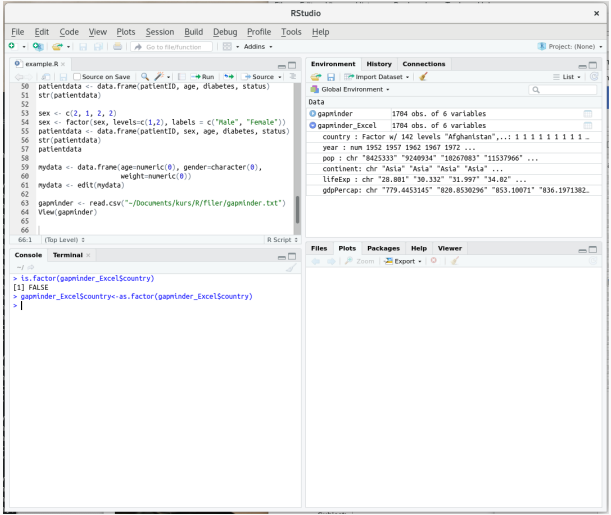
Here you can choose different codes for missing values, if certain rows or columns should be omitted, or if only a random selection of the data are imported.

IMPORTING A DATA FILE - XLS/XLSX

Check that the data have been imported correctly, and copy the commands into your script. The procedures for importing files from SPSS, SAS, State, etc. are similar.



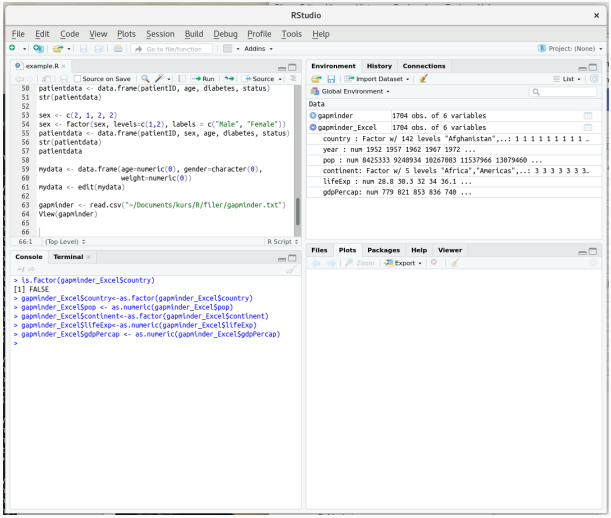
Importing a data file - xls/xlsx



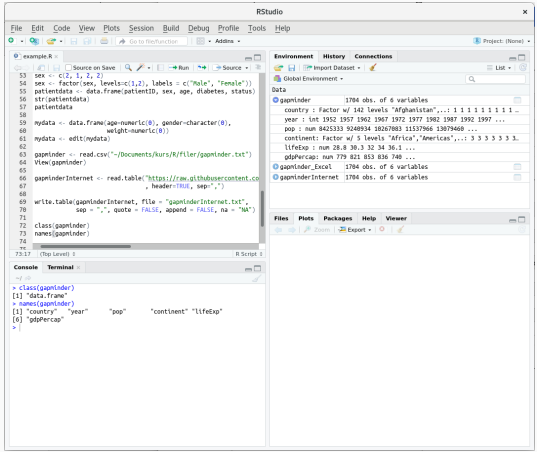
Factors should be recoded from Character to Factor. Use the same procedure as previously, variable <- as.factor(variable).

IMPORTING A DATA FILE - XLS/XLSX

Do the same for all categorical variables that have been imported as 'char', and for ordered factors, make sure to declare this and provide the order of the levels you want.



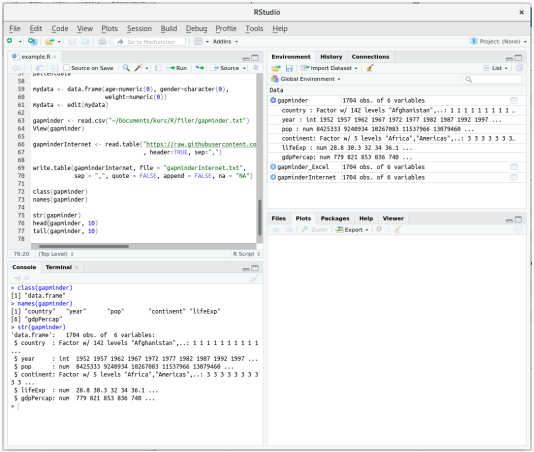
CHECK THE DATA SET



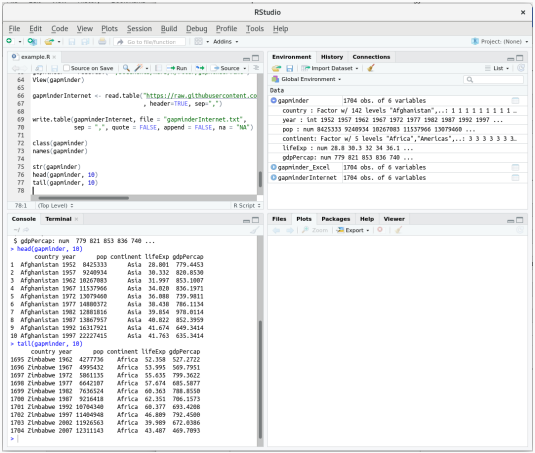
Check that your data set has been imported as a data frame, by using class(dataset). If it hasn't there are many analyses that will not run. Use the function as.data.frame(dataset) if your data set is not a data frame.

Get an overview of your data set

Different functions can be used to look at your data. View(dataset) will show the spreadsheet. Clicking the data set name in *Environment* does the same. str(dataset) will give information of the variables, their types, etc. You get the same information by clicking the symbol next to the data set name in *Environment*.



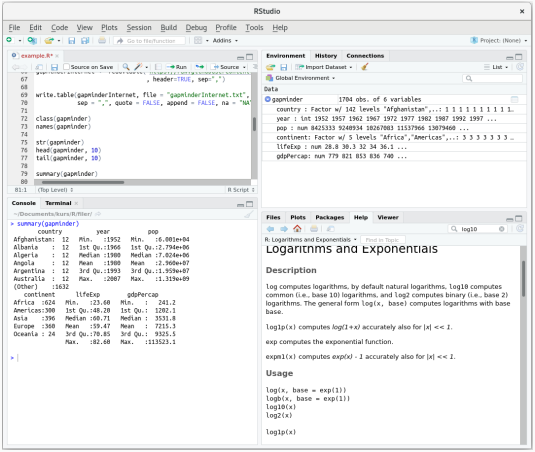
GET AN OVERVIEW OF YOUR DATA SET



To see the *n* first rows: head(dataset, n). See the *n* last rows: tail(dataset, n). If you write head(dataset), you automatically get the first 6 rows.

Get an overview of your data set

summary(dataset) gives simple descriptive statistics. For categorical variables you get the different categories with correspodning frequencies. For numerical variables you get mean, median, quartiles, maximum and minimum.



Exercises

1. Based on the information you can get from str(data$variable) and summary(data$variable), can you find out which years you have data from?
2. What happens if you combine the summary function with as.factor(date$variable) for the variable *year*? Does this change the variable itself?

DESCRIPTIVE STATISTICS

The difference between a sample and a population

We use statistics to say something about a population as a whole, based on results from a sample.

The difference between a sample and a population

We use statistics to say something about a population as a whole, based on results from a sample.

Even though a sample is randomly chosen and representative of the population, there will always be small individual variations, which can give rise to slightly different results each time an experiment is carried out.

The difference between a sample and a population

We use statistics to say something about a population as a whole, based on results from a sample.

Even though a sample is randomly chosen and representative of the population, there will always be small individual variations, which can give rise to slightly different results each time an experiment is carried out.

This means that our results can be a bit different from the true values of the population.

The difference between a sample and a population

We use statistics to say something about a population as a whole, based on results from a sample.

Even though a sample is randomly chosen and representative of the population, there will always be small individual variations, which can give rise to slightly different results each time an experiment is carried out.

This means that our results can be a bit different from the true values of the population.

The greater the sample, the closer we are likely to be to the true values. Rule of thumb: have at least 30 cases, ideally many more. If there is much variation, or if you are looking for small differences or effects, you need more cases.

Important statistical measures

Important statistical measures

- Frequency and relative frequency - how many have answered yes, no or maybe, how many are men how many women, etc.

Important statistical measures

- Frequency and relative frequency - how many have answered yes, no or maybe, how many are men how many women, etc.
- Mode - the most common value or category you measure. There can be more than one mode.

Important statistical measures

- Frequency and relative frequency - how many have answered yes, no or maybe, how many are men how many women, etc.
- Mode - the most common value or category you measure. There can be more than one mode.
- Mean, also called expectation value. The mean does not have to be a value you have actually measured

Important statistical measures

- Frequency and relative frequency - how many have answered yes, no or maybe, how many are men how many women, etc.
- Mode - the most common value or category you measure. There can be more than one mode.
- Mean, also called expectation value. The mean does not have to be a value you have actually measured
- Median - the middle measurement when data are arranged in ascending order

Important statistical measures

- Standard deviation and standard error - the variation or spread in the data, deviation from the mean

Important statistical measures

- Standard deviation and standard error - the variation or spread in the data, deviation from the mean
- Skewness - deviation from symmetry

Important statistical measures

- Standard deviation and standard error - the variation or spread in the data, deviation from the mean
- Skewness - deviation from symmetry
- Kurtosis - long tails and pointy distribution, or short tails and data mostly found around or close to the mean

Important statistical measures

- Standard deviation and standard error - the variation or spread in the data, deviation from the mean
- Skewness - deviation from symmetry
- Kurtosis - long tails and pointy distribution, or short tails and data mostly found around or close to the mean
- Percentiles - value below which a certain percentage of the data are found, e.g. the 25% percentile indicates the value marking the upper limit of the lower 25% of the data

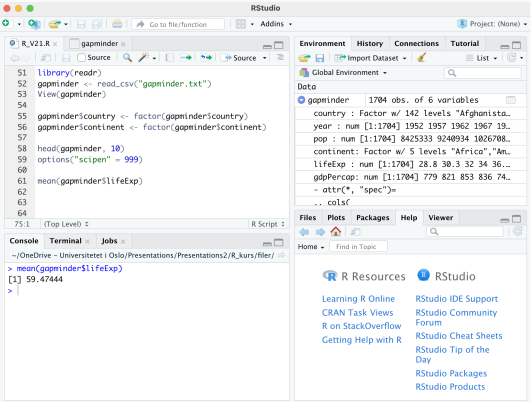
Important statistical measures

- Standard deviation and standard error - the variation or spread in the data, deviation from the mean
- Skewness - deviation from symmetry
- Kurtosis - long tails and pointy distribution, or short tails and data mostly found around or close to the mean
- Percentiles - value below which a certain percentage of the data are found, e.g. the 25% percentile indicates the value marking the upper limit of the lower 25% of the data
- Interquartile Range (IQR) - the middle half of the data

Important statistical measures

- Standard deviation and standard error - the variation or spread in the data, deviation from the mean
- Skewness - deviation from symmetry
- Kurtosis - long tails and pointy distribution, or short tails and data mostly found around or close to the mean
- Percentiles - value below which a certain percentage of the data are found, e.g. the 25% percentile indicates the value marking the upper limit of the lower 25% of the data
- Interquartile Range (IQR) - the middle half of the data
- Confidence interval (CI) - interval estimate rather than a point estimate of a statistical measure. If we conduct the same experiment many times, then the 95% CI for the mean will in 95% of the experiments contain the true mean of the population

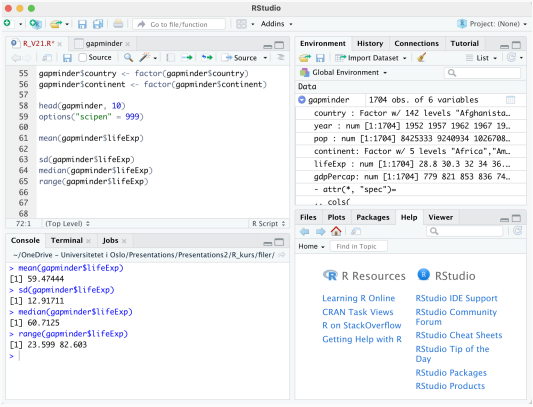
DESCRIPTIVE STATISTICS FUNCTIONS



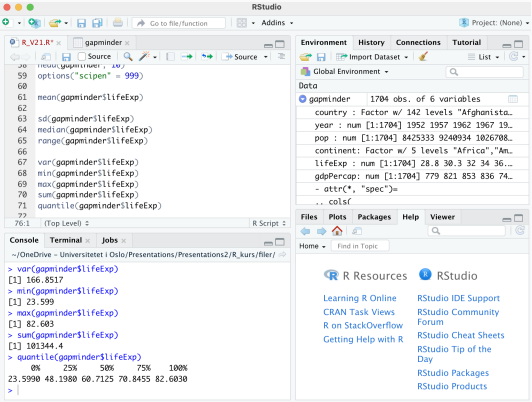
R has many built-in functions for calculating descriptive statistics. To calculate the mean you can use the function mean(variable).

DESCRIPTIVE STATISTICS FUNCTIONS

Just like mean(variable), we have sd(variable) for standard deviation, median(variable), range(variable).



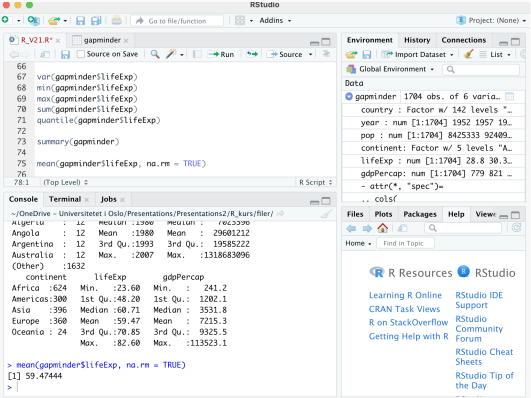
Descriptive statistics functions



var(variable) calculates variance, quantile(variable, *percentile*) finds the percentile(s) you ask for. If you only write quantile(variable), you will get the 0%, 25%, 50%, 75% and 100% percentiles automatically. Other useful functions are e.g. sum(variable), min(variable), max(variable).

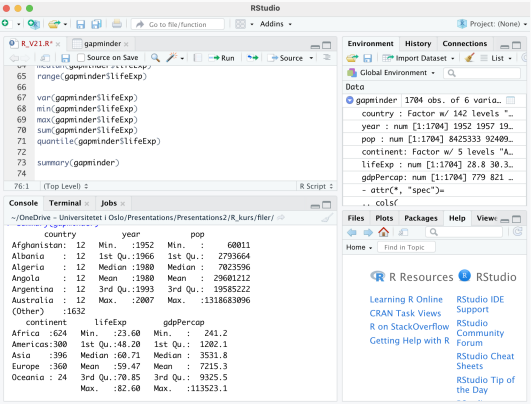
HANDLING MISSING VALUES

Most functions have a specific way to handle missing values. The function mean(variable) where the variable has missing values will give the answer NA. You must use mean(variable, na.rm=TRUE), which 'removes' the missing values from the calculation. Some functions require na.omit or na.exclude instead of na.rm.



SUMMARY

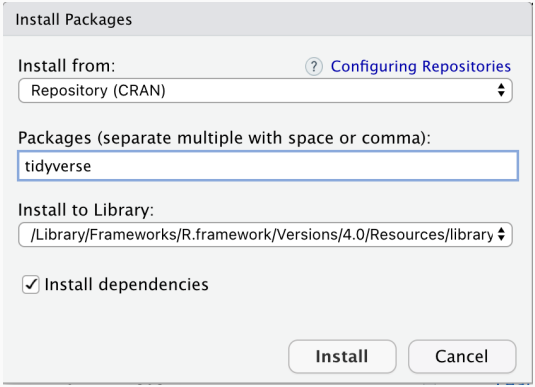
The summary functions gives you a small selection of descriptive information about all the variables in a set. It gives different types of output for numerical and categorical variables.



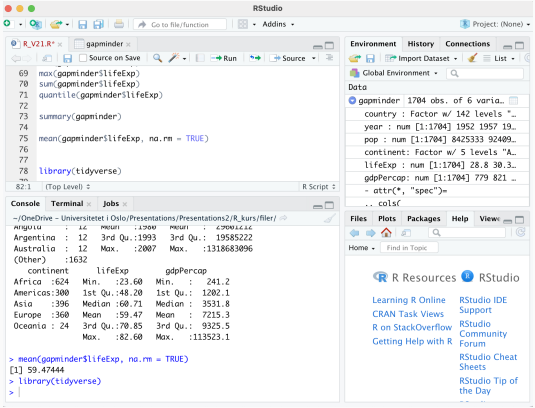
DATA HANDLING

Installing new packages

You can do a lot with the basic packages that you get automatically, but there are many more options available through additional packages. To install a new package, go to Tools > Install Packages. Write the name of the package you want, and click *Install*.



USING NEW PACKAGES



To start using the new package, you can either write library("*name of the package*"), or go to the list of packages on the right hand side, find it in the list, and check the box next to it.

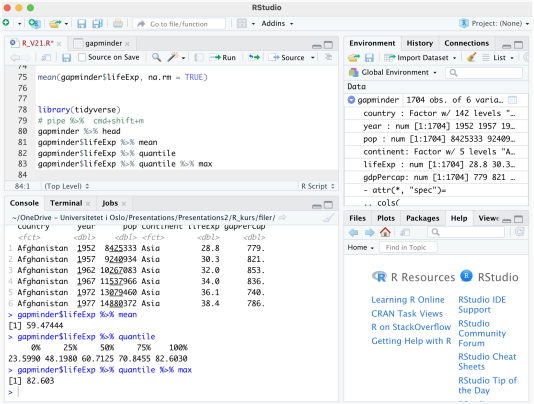
TIDYVERSE - A COLLECTION OF PACKAGES TO MAKE YOUR LIFE EASIER

Tidyverse is really a library made up of mulitple libraries. We have already used *readr* to import our data, and later we will use *ggplot2* to make graphs.



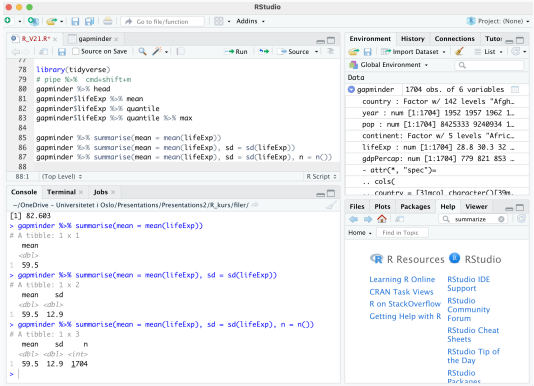
PIPE % > %

Pipe (% > %, cmnd/ctrl + shift + m) is a tool that allows you to "push" an object (e.g. a dataset or a variable) to a function. Whatever comes out of the function can then be pushed on further to a next step with another pipe. Useful whenever you are performing multiple operations after each other and don't need to save the in-between steps as separate datasets or variables.



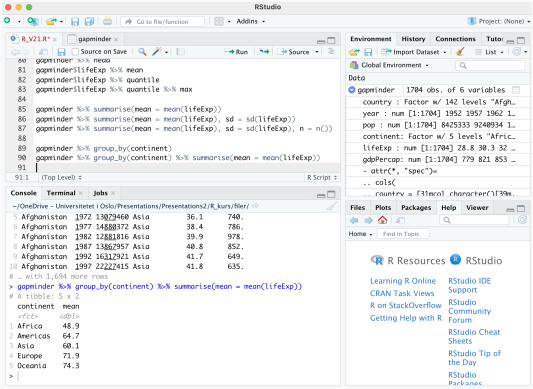
THE SUMMARIZE/SUMMARISE FUNCTION

The *summarize/summarise* function lets you calculate as many different summary statistics as you wish. The input it needs is which function(s) you wish to use and on which variable(s). In addition the caluclated value should be given a name (not required, but recommended).



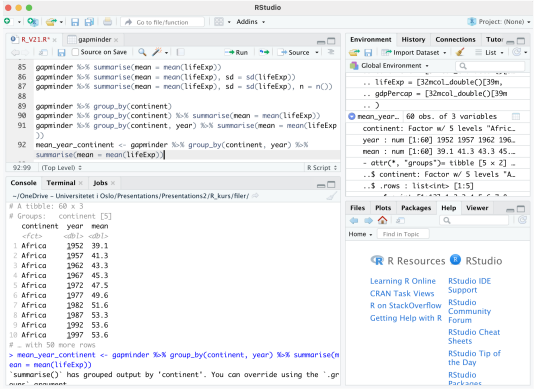
THE GROUP_BY FUNCTION

In order to work with groups defined by a categorical variable you can use the function *group_by*. If it is combined with e.g. *summarise* you can calculate descriptive statistics for each group.

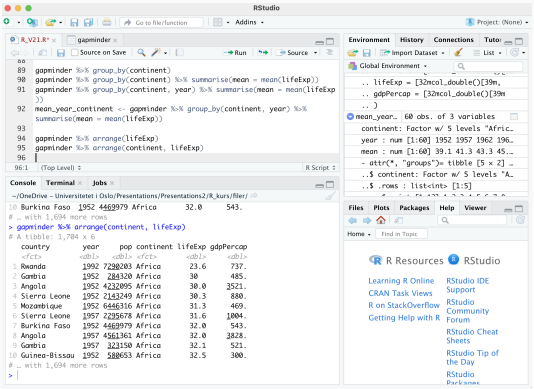


THE GROUP_BY FUNCTION

You can also use more than one variable to create subgroups.



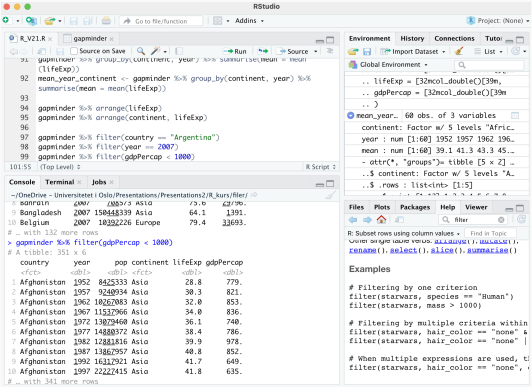
SORTING DATA - ARRANGE



If you wish to arrange or sort the rows in your dataset according to some criterion defined by one or more variables you can do this with *arrange*. This can be especially useful when looking for cases with extreme values in one or more variables.

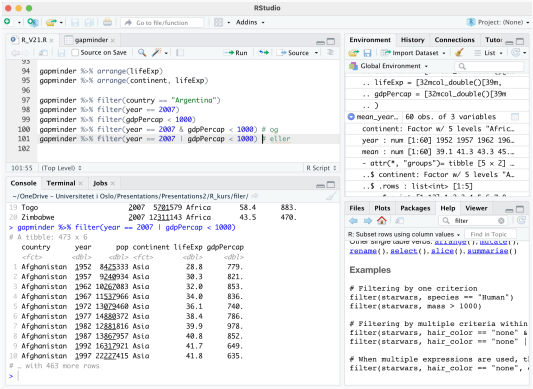
SELECT SUBSETS - SELECTING CASES USING FILTER

You can select subsets based on a condition using the *filter* function. You can set different kinds of conditions, either by selecting all cases from a particular category in a variable, or by requiring cases to satisfy some numerical condition (e.g. greater than/smaller than).

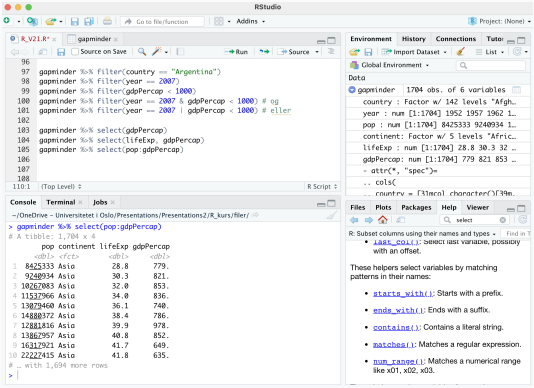


SELECT SUBSETS - SELECTING CASES USING FILTER

You can combine conditions to create a more complex filter. If you want data to satisfy to different conditions simultaneously, you combine conditions using & (AND), if you wish data to satisfy either one condition or another you can use| (OR).

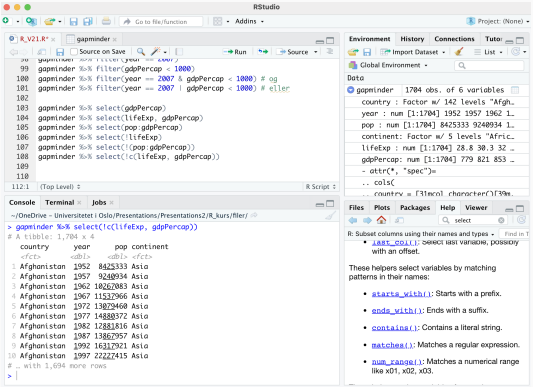


SELECT SUBSETS - SELECTING VARIABLES USING SELECT



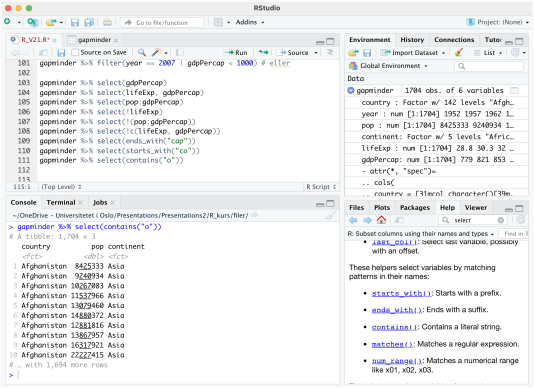
It is also possible to create subsets including all cases, but where you only select some of the variables. This is especially useful when you have very large datasets. The function *select* lets you choose variables by listing the variable names.

SELECT SUBSETS - SELECTING VARIABLES USING SELECT



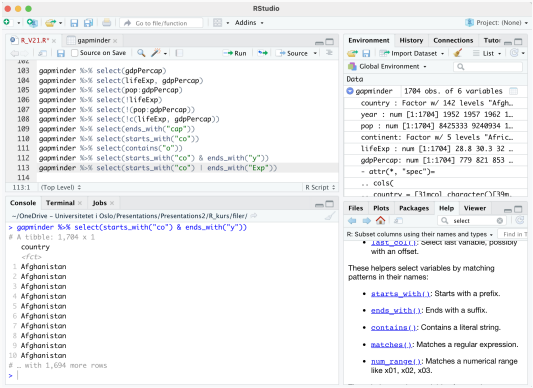
You can also exclude variables with !. You will then get all variables except the one(s) with ! written before the variable name.

SELECT SUBSETS - SELECTING VARIABLES USING SELECT



In addition you can provide the variable names using additional functions like *starts_with()*, *ends_with()* and *contains()*.

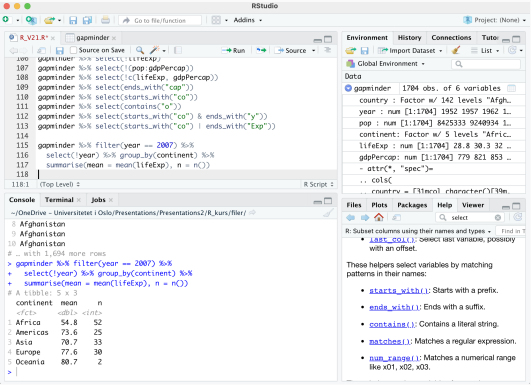
SELECT SUBSETS - SELECTING VARIABLES USING SELECT



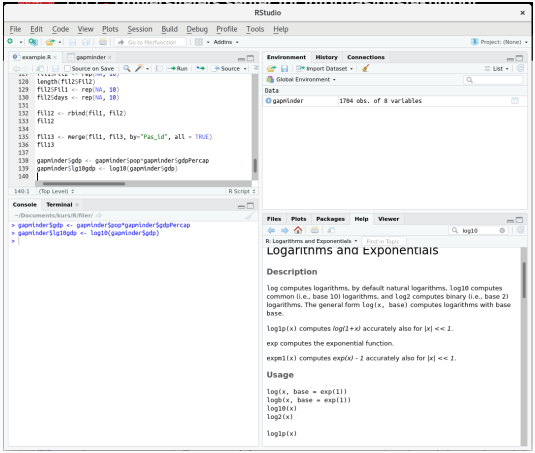
As with filter you can also combine the selection criteria, either with & (meaning *and*) or with | (meaning *or*).

COMBINING DIFFERENT FUNCTIONS USING THE PIPE

You can combine all these functions one after the other using the pipe. Whatever was calculated at the current step is then passed on to the next step through the pipe, for further processing.

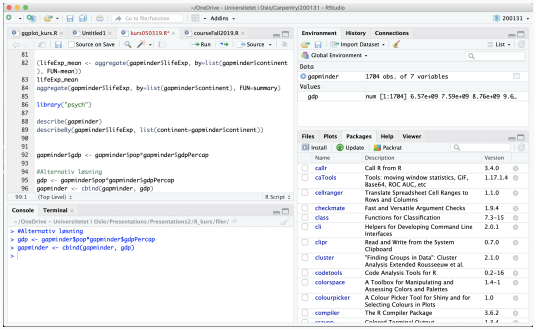


Creating new variables



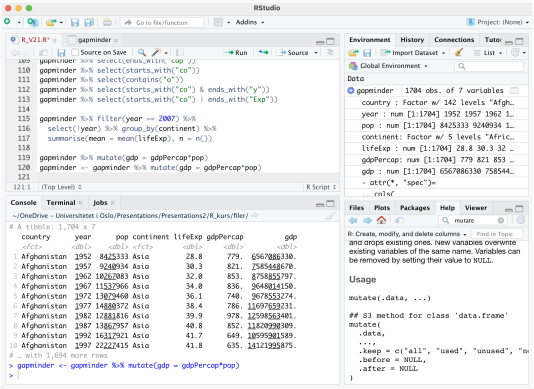
data$NewVariabel <- <*some expression*> will create a new variable in the data set. Using this, we can e.g. calculate new variables, based on the old ones.

CREATING NEW VARIABLES WITH CBIND



Alternatively, you can calculate the new variable (as a separate vector), and combine it with your data set usinng *cbind*(data set, new variable).

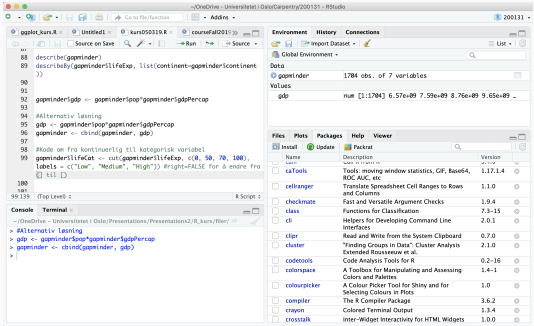
CREATING NEW VARIABLES WITH MUTATE



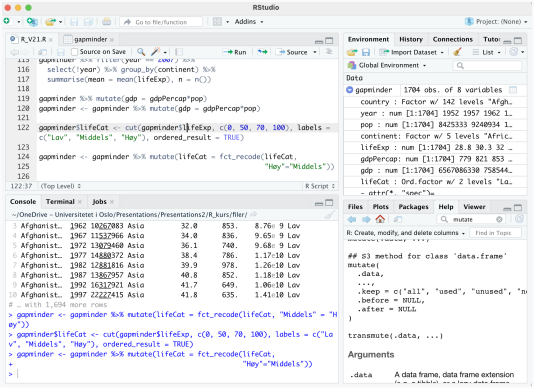
Tidyverse also has a function for creating new variables. The function is called *mutate* and needs as input the name of the new variable and the mathematical expression needed to calculate the variable in question.

RECODE VARIABLES

Recode from a continuous to a categorical variable. We use the function *cut*(), which needs as input the variable we want to recode, and the limits between the new categories. The lower and upper limits define the max and min, the others definne the cut points.



RECODE VARIABLES



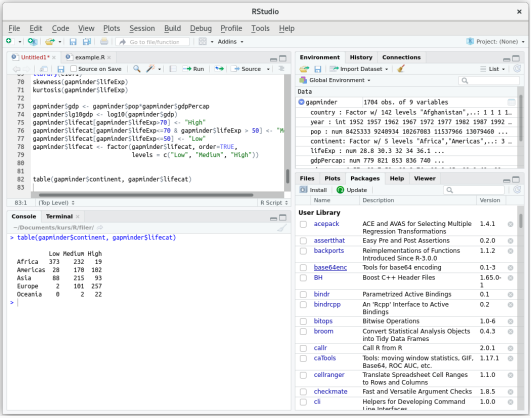
If you wish to change the codes in an existing variable you can do this in several different ways. One option is to use the *mutate* function along with a function called *fct_recode("new value" = "old value")*.

Exercises

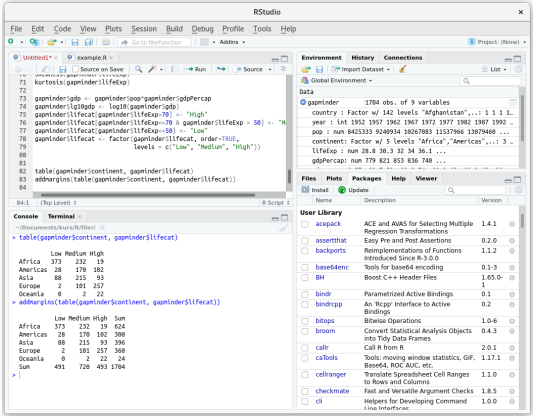
1. Create a new variable called gdpPercap1000, which corresponds to the GDP per capita in 1000 USD instead of USD.
2. Calculate the average population for 2007 group by the life expectancy categories in lifeCat, and count the number of countries you have in each of the categories (hint: n()).
3. Create a new dataset that only contains the data from Argentina, by choosing those that have country=="Argentina". Call this set gpArgentina.

Frequency and contingency tables

The frequencies of different categories can be presented for one or two (or more) categorical variables, using table(var1, ...). If you include three variables, you will get one table for each category in the third variable. It is not recommended to include more than three variables at a time.



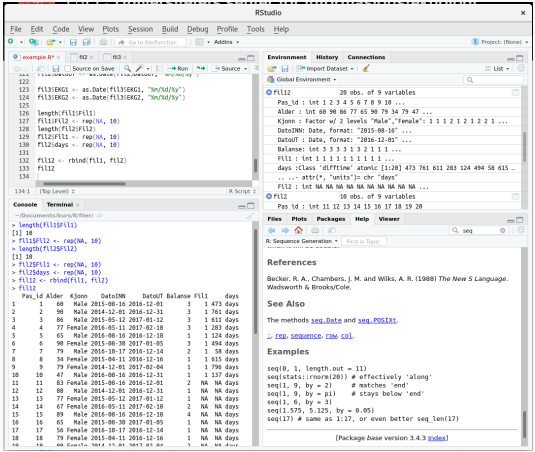
Frequency and contingency tables



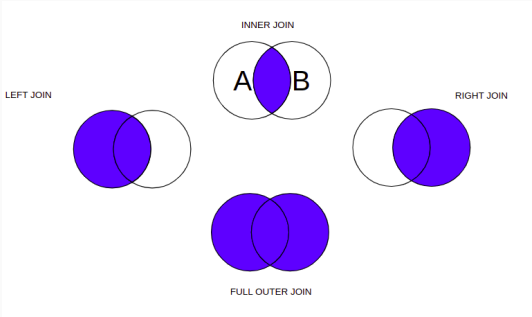
addmargins(table(variable)) adds the total sum for each row and column.

MERGING DATA SETS - ADD CASES

To merge two files, where we want to add cases, both files MUST gave the same variables. If they do not have all the same variables, empty dummy variables must be created where needed. Use rbind(fil1, fil2) to make a new set with all the rows/cases from both sets.



MERGING DATA SET - ADD VARIABLES



There are several different "join" operations that will merge datasets in slightly different ways

By using a FULL JOIN, you will get all IDs, even if some of them only occur in one set originally.

Variables where data is missing will be coded as NA.

Adding and removing specified rows

```
# Add rows
patient_data <- patientdata %>%
  add_row(ID = 5, age = 22, diabetes = "Type1", status = "Poorly")

# keep rows
patient_data %>%  slice(1)
patient_data %>%  slice(2:3)

# remove rows
patient_data %>% slice(-1)
patient_data %>% slice(-(1:2))
```

You can add and remove/select rows using the add_row() and slice() functions respectively

Dealing with missing values

```r
airquality %>%
  drop_na(Ozone, Solar.R)

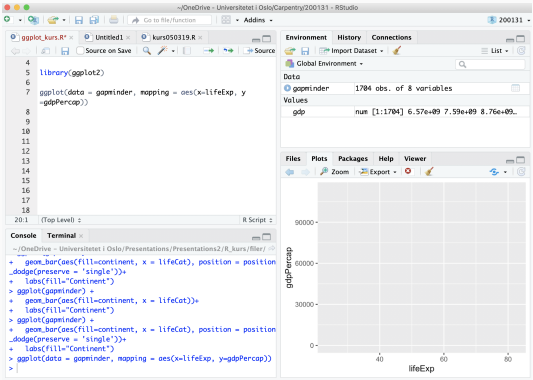

# replace NA
airquality$Ozone

airquality %>%
  mutate(Ozone = replace_na(Ozone, 9999))
```

You can remove rows with missing using drop_na().
Alternatively, NAs can be replaced using replace_na().
Another strategy is imputation, of which R has many methods. But that is out of the scope of this course

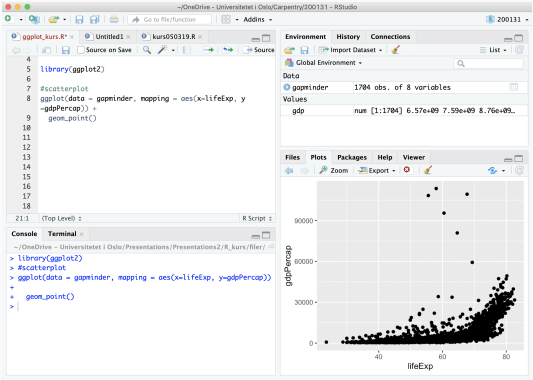
MAKING GRAPHS

The ggplot2 library

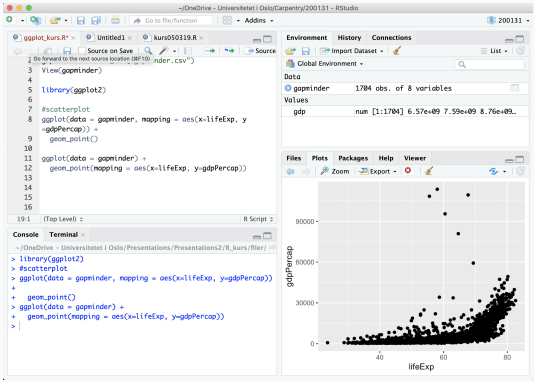


With the ggplot2 library you make plots by creating layers that lie on top of each other. This gives great flexibility when it comes to what and how much information you want to include in a single plot. If we don't ask for a specific type of markers or lines, we only get an empty graph.

The second layer specifies that we want a scatter plot. First you choose the dataset, and then which variables within that dataset that should be plotted. In this case you can choose if you set the data set and variables inside ggplot() or inside geom_point(), but that is not always the case. ggplot() is common for the entire plot, geom_point() specifically relates to the layer with the scatter points.



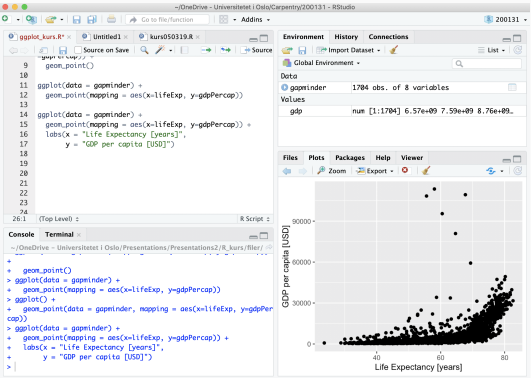
The ggplot2 library



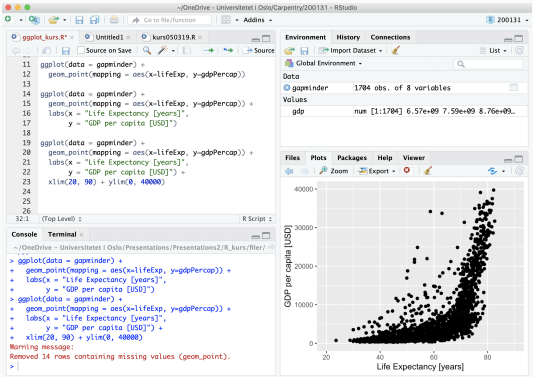
You can also define variables inside the scatter point layer. The data set where the variables should be taken from can also be assigned in the point layer. This can be useful if you want to include data from multiple sets in the same plot.

EDITING PLOTS

To change the labels on the axes we add the layer *labs*, and inside this layer we specify x="Label on the x-axis" and y="Label on the y-axis", with a comma between the two.



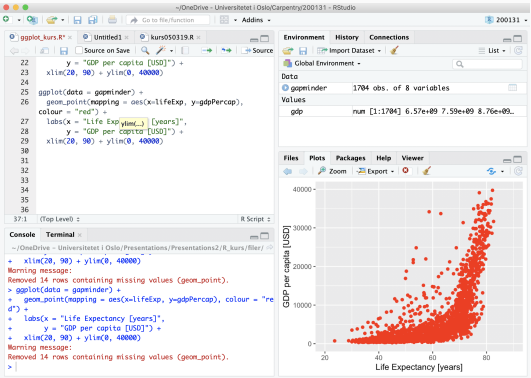
EDITING PLOTS



To change the area shown in the graph we can add two more layers, one for each axis: xlim and ylim. The limits are specified by writing xlim (*start*, *stop*) and ylim (*start*, *stop*), where *start* and *stop* should be numbers.

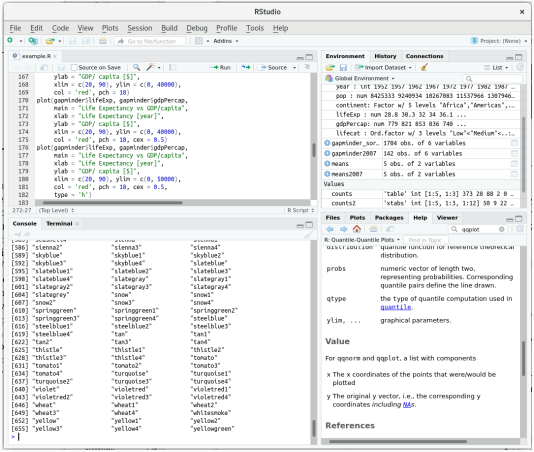
EDITING PLOTS

To change the colour of the points, you can specify this inside the point layer, i.e. inside geom_point(aes(x,y), color="some colour").



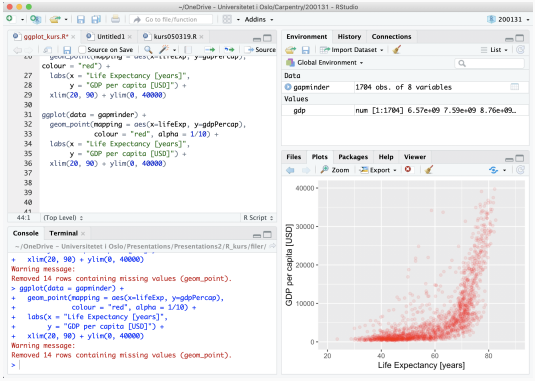
EDITING PLOTS

There are a lot of colours you can choose from in R. By typing colours() in the console you can get an overview. There are for instance over 100 different shades of grey.



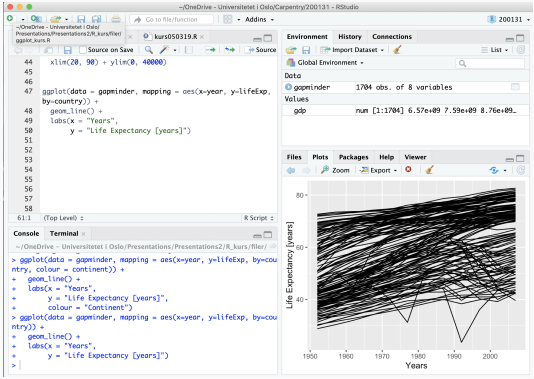
EDITING PLOTS

You can also change how transparent the points should be with the option alpha, which is a relative number between 0 and 1, where 1 means completely solid and 0 is completely transparent. This can be assigned the same way as colour, inside geom_point().



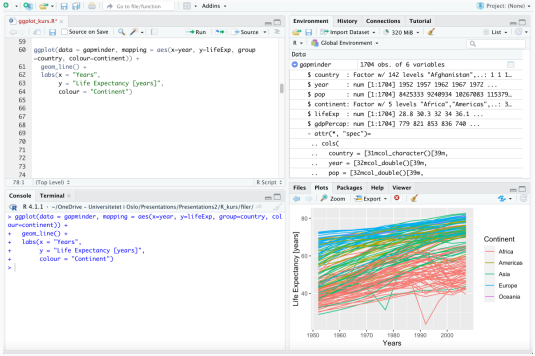
MULTIPLE LAYERS

In the same way that we use geom_point() to create points, we can use similar functions to create other types of plots. One option is geom_line(). The benefit is that we can easily ask for separate lines for e.g. different countries.



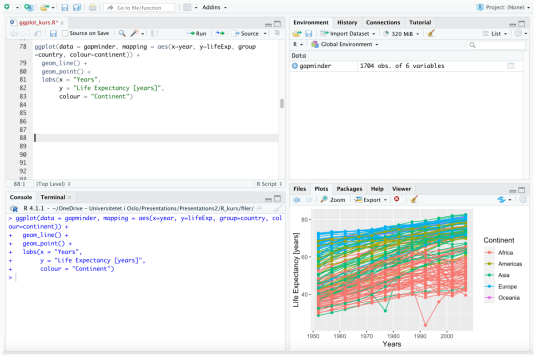
MULTIPLE LAYERS

We can also ask for different colour lines for the countries from different continents. If you want to control the text in the legend (the colour indicator on the side), you can do this with colour="alternative text" inside the labs layer.



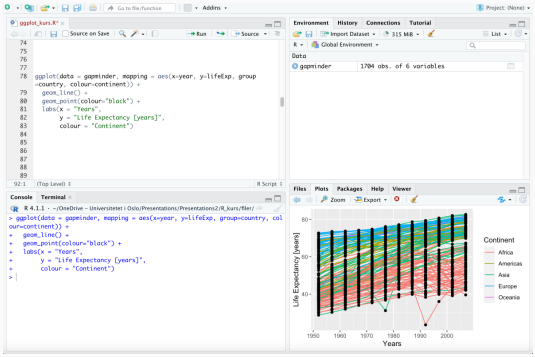
MULTIPLE LAYERS

If we want to highlight the data points we can add a layer of points on top of the layer of lines.



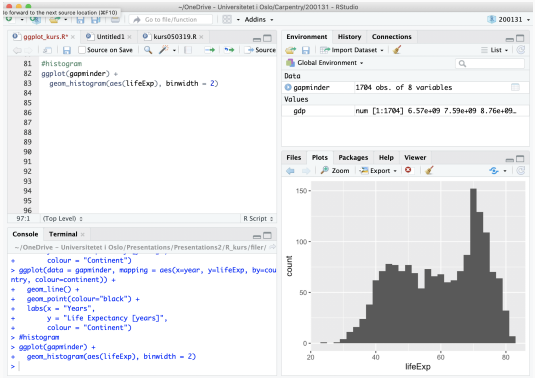
MULTIPLE LAYERS

We can also define other properties for this layer, like change the colour of the dots, but keep the colours of the lines as they were.

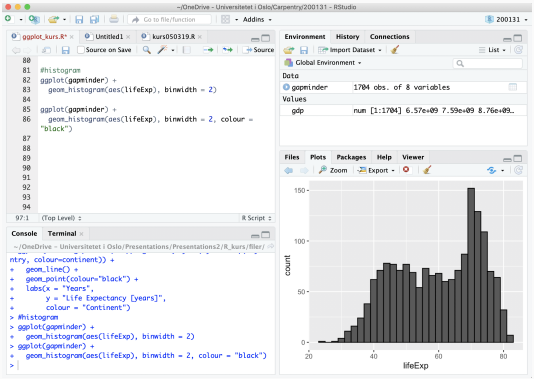


HISTOGRAM

To show how the data in a continuous variable are distributed you can use a histogram. You can make such a diagram with a layer like geom_histogram(aes(variabel), binwidth=<column width>)



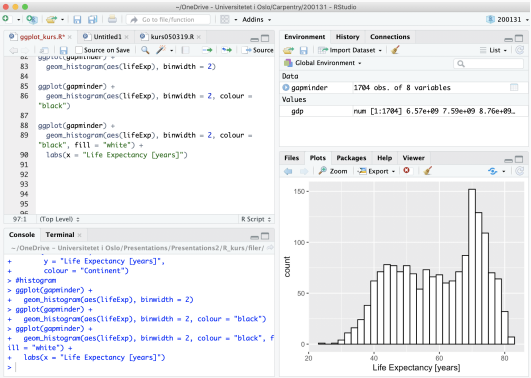
HISTOGRAM



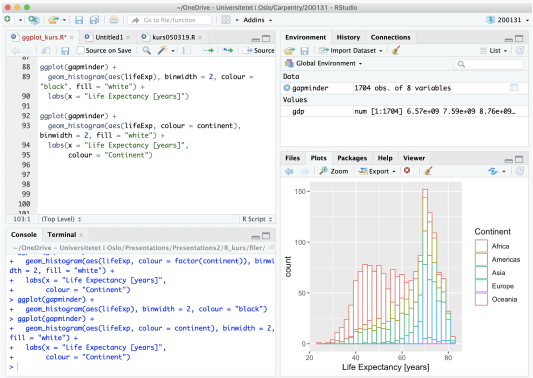
You can change the colour of the column contours with colour, inside the histogram layer.

Histogram

To change the colour of the columns themselves, you asign a colour to the fill, by including fill="wanted colour" inside the histogram layer. To change the label on the x-axis you can do this the same way as before, with a labs layer where you set x="the text you want".



HISTOGRAM

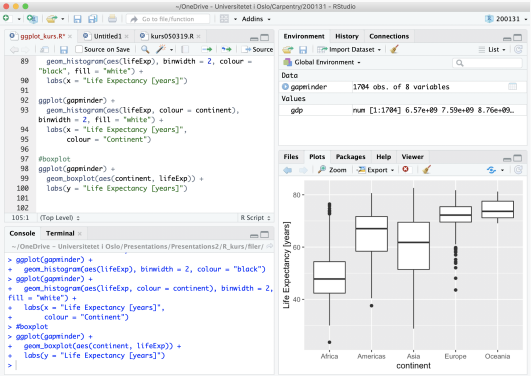


You can also separate the columns into different categories defined by a categorical variable, creating a stacked histogram, where e.g. each continent gets its own colour.

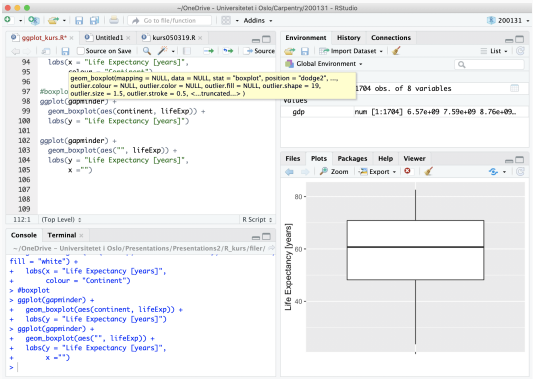
.

Box plot

A different way to present continuous variables, especially when you want to group the variable, is with a box plot. This plot also shows the median and the quartiles.The command that creates such a layer is geom_boxplot(aes(categoricalVariable, continuousVariable)).

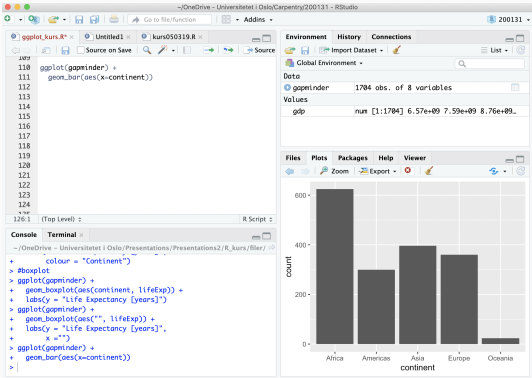


BOXPLOT



If you do not wish to divide the continuous variable into separate groups defined by a categorical variable,you can simply define this spot as empty. .

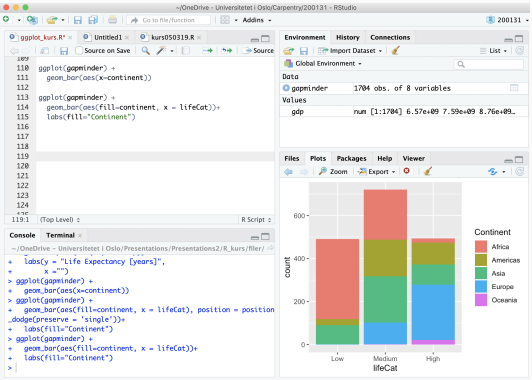
BAR CHART



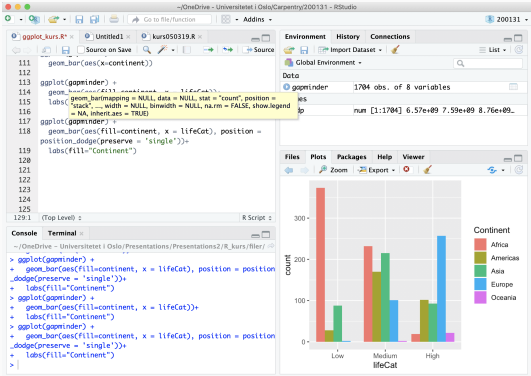
To show frequencies for the different categories of a categorical variable you can use a bar chart. For this we must add a layer geom_bar().

Stacked bar chart

A stacked bar chart presents the frequencies of two categorical variables. You do this by specifying that the fill of the bars should be coloured according to a second categorical variable, like here with fill=continent. This is assigned along with the first categorical variable inside aes().

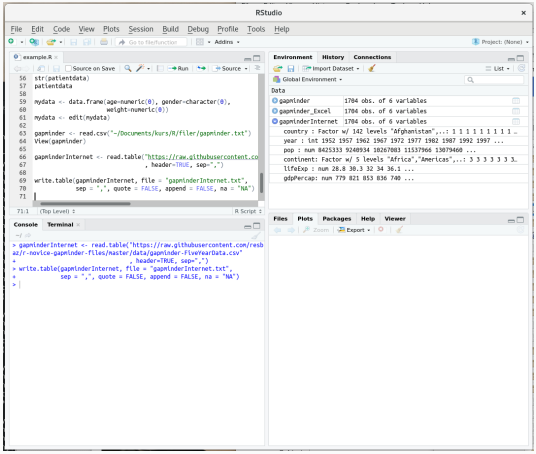


CLUSTERED BAR CHART



By adding an extra command inside geom_bar(...), we can get the bars clustered next to each other on the x-axis instead. Use position = position_dodge(preserve = 'single')

EXPORTING DATA



Save a data set to file with write.table(...). Inside the parenthesis you give the name of the data set, the file name you want it to have (file=), which symbol should separate columns (sep=), what indicates missing data, etc. append=FALSE decides that this should not be added to an existing file, quote=FALSE means that string values are saved without "".

BACK TO THE STUDY HALL/OFFICE

Further help with R

Workshops in R offered by the Carpentries. More information can be found here –> https://www.ub.uio.no/english/libraries/dsc/carpentry-uio/

Ask Google

Ask us: statistikk@usit.uio.no

Sign up for our course announcement email list:
https://sympa.uio.no/usit.uio.no/subscribe/task-kurs