An Introduction to R

Dr Simon Rudkin

This document accompanies the one-day session. It is assumed that you have some familiarity with statistics. However, it is not necessary to have previous experience of coding in R or any other language. This file, the code files, data and solutions will be provided via the GitHub site associated with the session. You can find the GitHub at github.com/srudkin12/IntrotoR/

All of the examples and exercises require the file citiesdata.csv and are based upon the paper by Castelli et al. (2023):

Castelli, C., d’Hombres, B., Dominicis, L. de, Dijkstra, L., Montalto, V. and Pontarollo, N. (2023). What makes cities happy? Factors contributing to life satisfaction in European cities. *European Urban and Regional Studies*, iFirst. <https://doi.org/10.1177/09697764231155335>

All of the code required is included within the file citiescode.txt but solutions to the open questions are not provided within that file. A separate file of solutions will be made available through the GitHub immediately after the session. You are encouraged to think about the structure of the code and how it would be adapted to work with your own data.

The objectives for this course are that on completion of this workshop, the participants will be able to:

* Install R and RStudio on their own machine
* Be familiar with the RStudio GUI
* Understand R variables, data types and objects
* Understand the use of vectors and Dataframes
* Understand how to get help and make use of R libraries
* Read datasets of different formats into the R environment
* Perform data cleaning and data manipulation using core R and the ‘tidyverse’ package
* Perform simple and some not so simple visualisations of data using the ggplot2 package
* Appreciate how such things as statistical analysis, machine learning and mapping of data can be performed using a variety of R packages readily available

We will be covering all of these objectives, though not necessarily in the same order as they are stated. See the schedule below

To work successfully through this session you are encouraged to use this document alongside:

1. A blank word document into which you can paste output from the session
2. A blank notepad file into which you can paste any code
3. A folder which contains the downloaded material from the GitHub site

The GitHub site for this session is: github.com/srudkin12/IntrotoR/

*We will cover this set up as part of the opening of the session.*

**Structure of Session**

The approximate timeline for the day will be as follows:

|  |  |
| --- | --- |
| 09:30 – 10:00 | Introduction and Session Aims |
| 10:00 – 10:30 | Loading and Viewing Data |
| 10:30 – 10:50 | Introduction to the Tidyverse |
| 11:00 – 11:30 | Working in the Tidyverse |
| 11:30 – 11:50 | Creating variables and basic coding |
| 11:50 – 12:10 | Introduction to plotting |
| 12:10 – 12:30 | Summary of Morning Session |
| 12:30 – 13:30 | Lunch Break |
| 13:30 – 14:15 | Further Plotting |
| 14:15 – 14:45 | Summarising Data |
| 14:45 – 15:45 | Exercises and own data |
| 15:45 – 16:15 | Further Topics |
| 16:15 – 16:30 | Review |

There will be ample opportunity for questions and the sharing of solutions. In the final session there are further exercises, or you may choose to spend time working on your own data files with the support available in the room.

**Variables Provided**

|  |  |  |
| --- | --- | --- |
| Name | Description | Format |
| CODE | Eurostat code identifying the city |  |
| NAME | Name of the city in the home language |  |
| Ptrans | Satisfaction with public transport |  |
| Health | Satisfaction with available health facilities |  |
| Cultfc | Satisfaction with available cultural facilities |  |
| Greens | Satisfaction with green spaces, including parks and countryside |  |
| public | Satisfaction with public spaces, including squares and civic buildings |  |
| Cleans | Satisfaction with the general level of cleanliness |  |
| Trusts | Extent to which the respondent believes they can trust others in the city |  |
| Safety | Extent to which the respondent feels safe when walking in their city |  |
| Afford | Extent to which the respondent believes that living in their city is affordable |  |
| Lives | Overall satisfaction with living in the city |  |

All of these variables are calculated as the percentage of respondents who say they are satisfied, or very satisfied, in response to the particular question. For example “Are you satisfied with the availability of public transport within your city?”. Following Castelli et al. (2023) we combine the top two levels of satisfaction. If you wish to work with the data with different levels of satisfaction included, then you will need to prepare the data manually from the Eurostat page.

*You should complete the final column using the learning from the session.*

**RStudio**

In this session we will use RStudio to run the R code. RStudio is a windows based graphical user interface for R which allows you to see your files, outputs and codes within the same piece of software. Running code in the basic R terminal will not allow you to see any of the elements, although they would still be stored within the workspace environment. To introduce the environment, see the screenshot below.

A screenshot of a computer

Description automatically generated

Output

Environment

Terminal

There are three key elements to the R Studio environment:

1. The Terminal which mimics the standard R interface and allows you to type code. All code is typed next to the prompt > and run by pressing Enter
2. The Environment which shows what objects exist within the R Environment. Objects can be tables, variables or collections of summary statistics.
3. The Output area which shows the files in the working directory, plots you have generated, packages which have been loaded and has further elements which can be used to display the work which has been produced.

You are encouraged to play with the various elements of the RStudio window. As we work through the material in this document.

The first step is to make sure that RStudio knows where to find the data files and save any output that you generate.

A screenshot of a computer

Description automatically generated

Zooming in on the output window you can see that there is a little house and an up arrow. These are common navigation symbols and allow you to move around through the folders of your computer. For this session it is recommended to have a folder and then keep everything in one place. In the example the folder is called Intro to R.

As recommended in the set-up, the file citiesdata.csv is already included within the folder. We will not load the file from here, but if you were using the R terminal then R would read directly from here.

**Importing Data**

In this session we are using the file citiesdata.csv and so we will read the file in using the Import Dataset option.

The first option to select is CSV as the file format. You will see that you can read in files from Stata and other statistical packages, as well as Excel and Text files.

After selecting CSV you will see something like the picture below. The data is designed to be read easily by R, but there are options to allow for files with different separators, decimals, quote (symbol used to tell R to recreate exactly what is between the symbols), and other further settings.

Graphical user interface, application

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

Once the dataset has been read into R you will see that the citiesdata.csv file has now appeared in the Environment. We are told that there are 99 observations with 12 variables.

Clicking on the citiesdata.csv within the Environment tab causes a further window to be opened within the RStudio environment. In the window in the top left appears the data which has just been loaded. We will consider more about the data in a moment.

The second element which has appeared is the code for the action which has just been undertaken by first loading the data and second clicking to view the data. If you were typing the code then this is what would be needed to load data without the menus.

Note that the filepath displays with a ~ symbol replacing the first elements. The actual code to produce the results so far would be:

Setwd(“C://CMShortCourses/Intro\_to\_R/”)

citiesdata<-read.csv(“citiesdata.csv”)

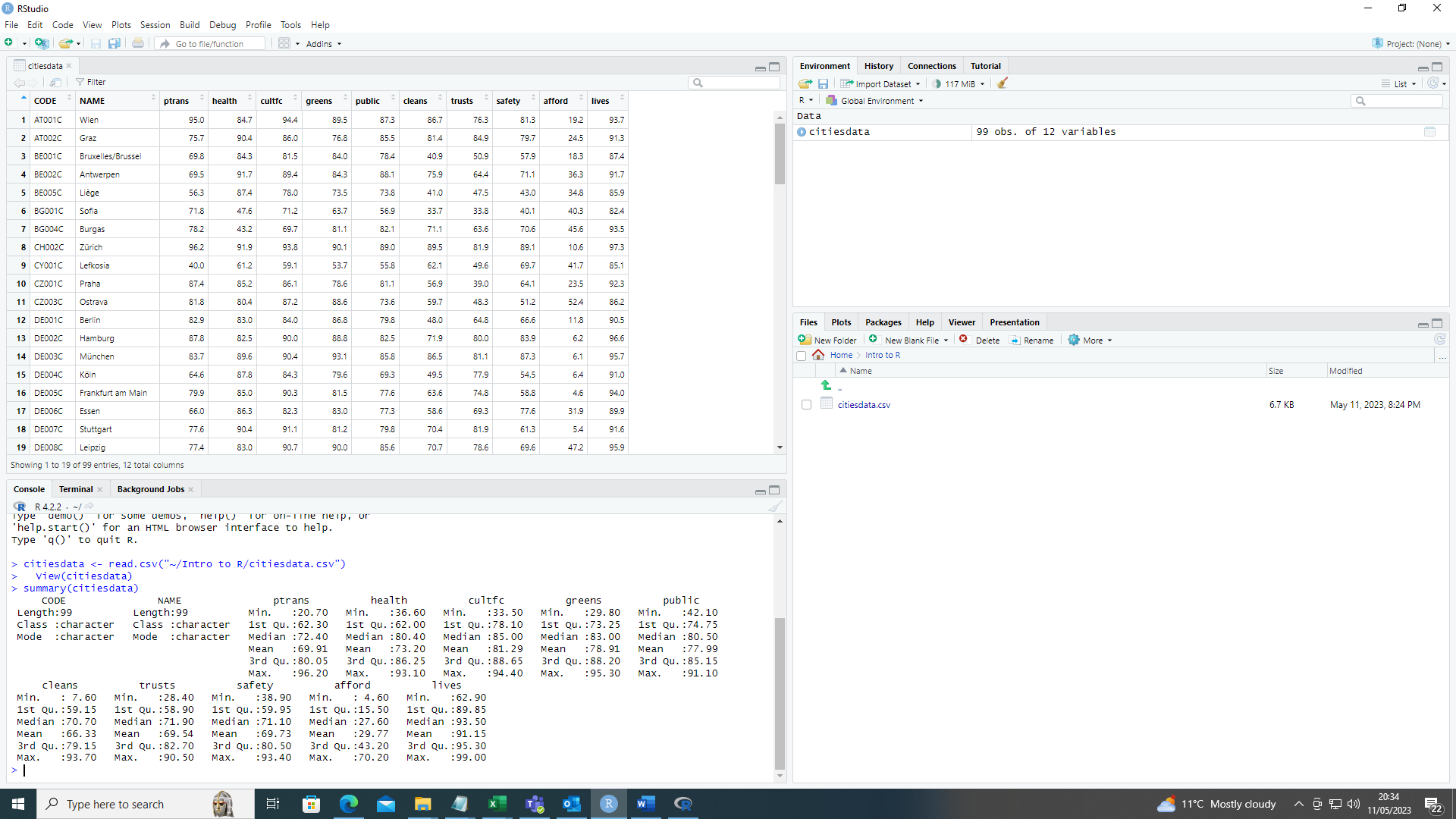
You can see this code within the code file. Note that Word will cause “ to display in a better looking format but R does not like the “ to slope. Within the .txt file you will see that the symbol displays correctly as vertical lines. If you wish to copy paste R code you are strongly encouraged to use Notepad (Windows), TextEdit (Apple) or another text editor.

With the data loaded we are ready to begin looking at the information it contains. If you wish to simultaneously try the commands on your own data then you may either read in your own data file following the steps above, or open a second RStudio window and then read your data into that separate environment.

**Viewing Data**

We have already seen that the data can be viewed by clicking on the file within the Environment. From the view we can see that the variables are all percentages and that we also have a code and a name for each city.

R has commands to allow you to view the data. In the screenshot we show what happens when you use the summary() command. You can summarise a variable by specifying just one variable within the brackets. In the example we use summary(citiesdata)



Attempt the following questions:

1. Why do the summaries of CODE and NAME appear differently?
2. What happens if you type summary(citiesdata$lives)? What do you think is the interpretation of the $ here?
3. What is the standard deviation of the variable ptrans? Hint: The standard deviation function is sd()
4. What do you get if you type cor(citiesdata$ptrans,citiesdata$public)?

Answers to these questions are available via the GitHub site after the session

**Packages**

R is supplied with a set of basic functions which can do many of the tasks which you are likely to need. However, there is also a large set of user generated functions that are designed to perform additional statistical analyses. Often a new methodology will be accompanied by new R code to allow other users to put the method into practice. New functions are stored in packages and these packages are available to download from the R repository (CRAN) or from the developers websites. *A note of caution when using new packages that are not yet on CRAN is that you have to make sure that you trust the source of the code before running it.*

A screenshot of a computer

Description automatically generated

The first time that you use a new package on your computer you will need to install that package. The install packages option can be found on the Tools menu.

A screenshot of a computer

Description automatically generated

If you know the name of the package you can just type it in. As you type you will see that RStudio will fill in the box automatically with suggestions for the package you are looking for. Here we are installing the package Tidyverse

The corresponding R code for installing a package will appear in the Console and then execute. Depending on how many packages are already installed on the computer you are using. R will always download all of the packages that you need.

Graphical user interface, text

Description automatically generated

This screen shot shows the packages downloading from the CRAN server. There is a list of all of the additional packages which R will install. The command to install tidyverse is simply:

install.packages(“tidyverse”)

A screenshot of a computer

Description automatically generated

In order to use a package you must add it to the library. This is done by typing library() and placing the package name within the parentheses. To load tidyverse you should type the following:

library(tidyverse)

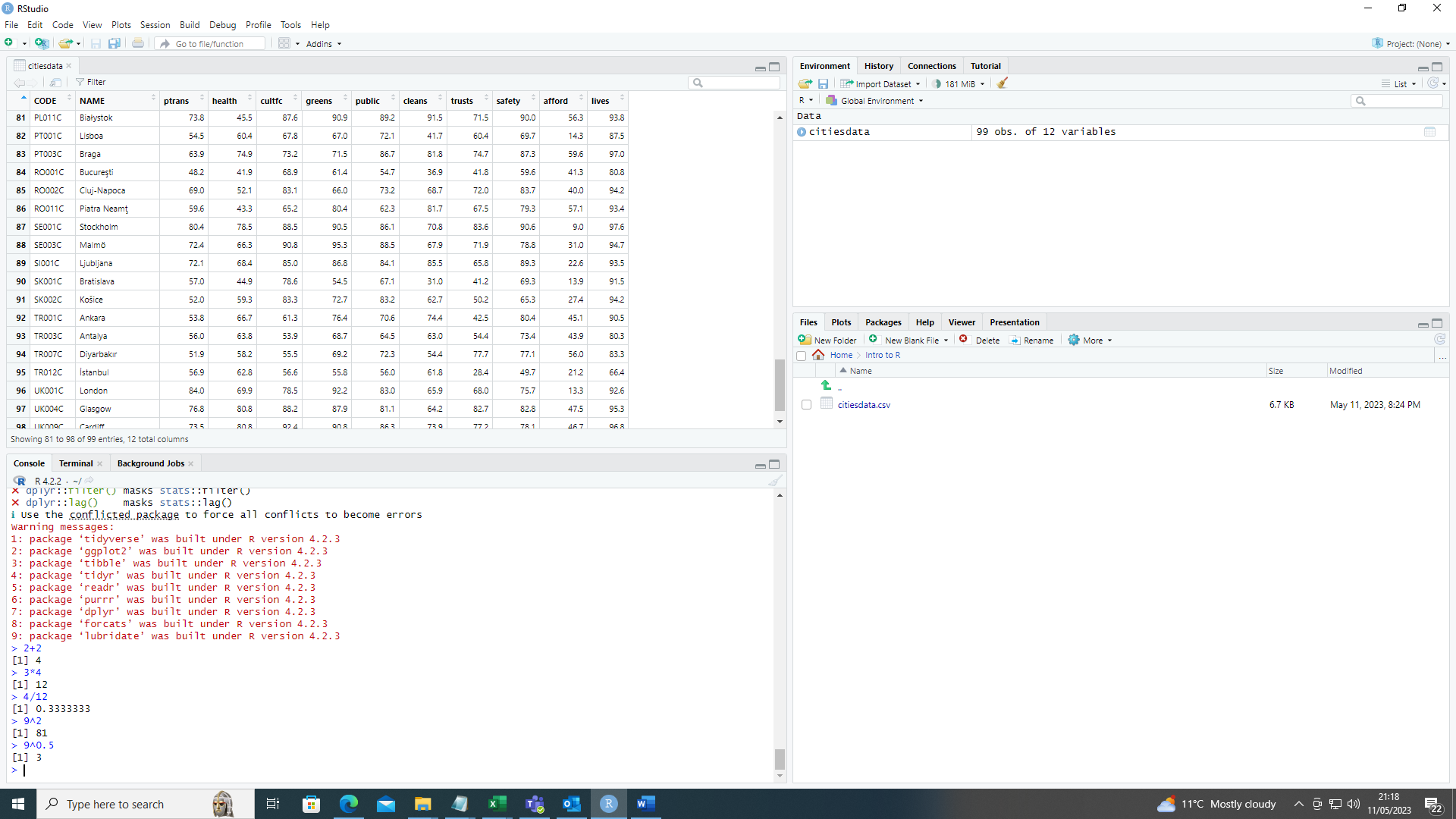
The data included with this course already meets the tidyverse criteria. The data is rectangular with each observation being a row in the table and each variable being a column. There are no missing values to consider.

The principles of tidy data are:

1. Each variable you measure should be in one column
2. Every observation of a variable should be in a different row
3. There should be one spreadsheet for each type of data
4. If you have multiple tables they should include a column in each spreadsheet with the same column label that allows them to be joined or merged

Verify that the data we include here does indeed meet the tidyverse criteria.

**R as an Object Based Language**

R is capable of performing calculations using matrices, vectors and simple numbers. To get used to the commands which are available we give some examples of simple arithmetic. The key operators which are used here are + - \* / for addition, subtraction, multiplication and division respectively. For powers the ^ is used. For the square root you may use the power 0.5 or the function sqrt()

To work with variables we can use the $ symbol. So, for example we would use the following command to make a decimal equivalent for the life satisfaction variable.

citiesdata$livesd<-citiesdata$lives/100

A screenshot of a computer

Description automatically generatedAt this stage it is helpful to talk more about the <- operator and the creation of objects within R.

R is described as an object-based language because the user can define objects to help with calculation. In the code on the right, we set a equal to 2 and b equal to 3. When we allocate values to letters, we need to use the <- operator. Once a value is allocated, R can perform manipulations. The example has a+b=5 as expected.

A screenshot of a computer

Description automatically generatedOnce objects are defined we can ask R questions about the objects. Although the current suggestions appear trivial, the logic is useful to use when we want to identify relationships between variables in a dataset. Because this is an introduction to R we will not be performing many comparisons between variables

1. Assign the values 8, 2 and 10 to variables c, d, and e respectively.
2. Calculate the product of a and c?
3. What is d divided by b?
4. Is a+b greater than d divided by e?

Graphical user interface, text, application

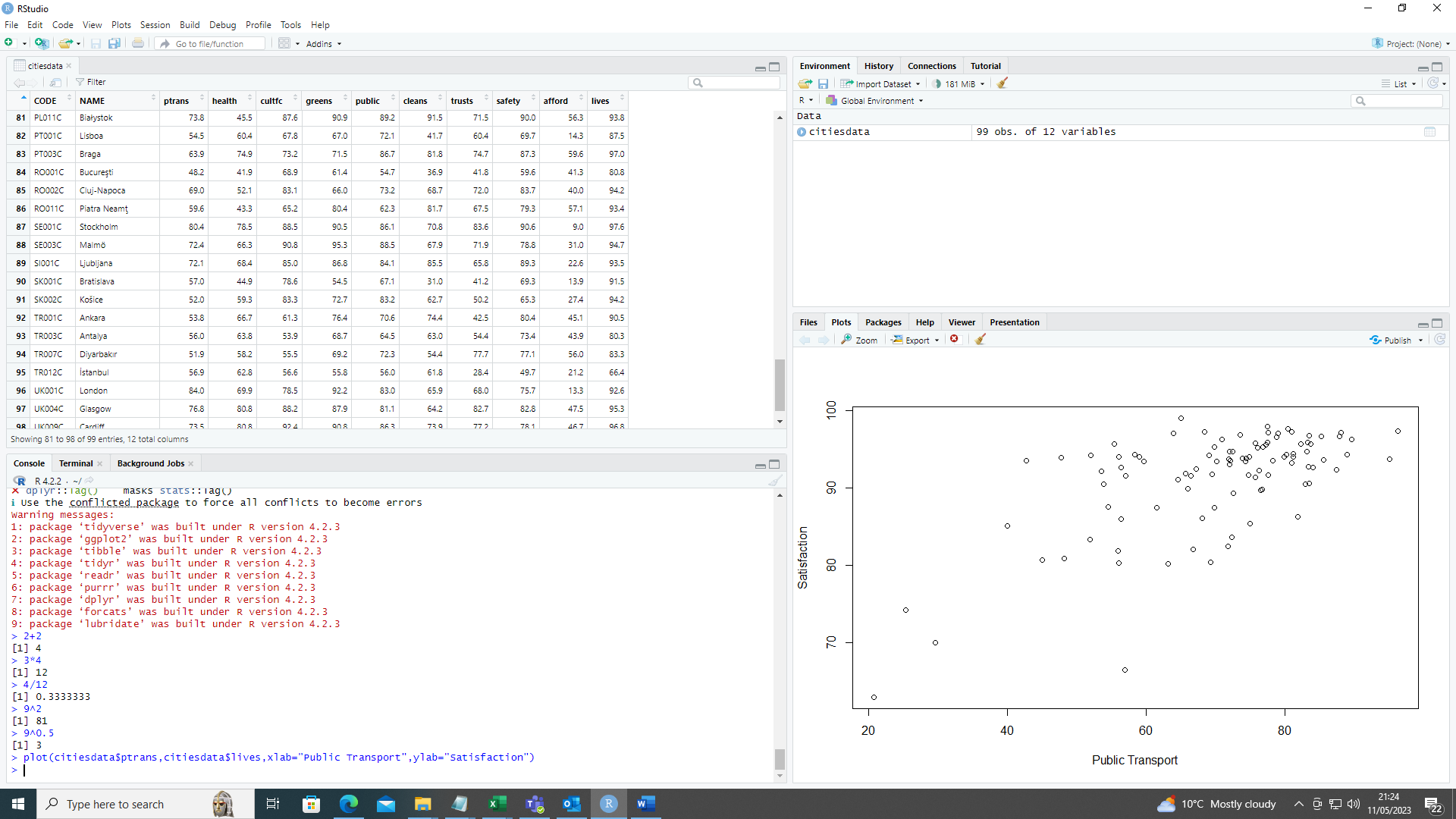
Description automatically generated

We will now create two new variables using the code above. The two variables are a dummy for the level of life satisfaction being greater than 90% and a variable which states which country the city is in. Eurostat codes have a two character country code within the overall code. To isolate the codes we are using the substr() command which creates a substring from the text beginning at the second argument and ending at the third argument. In this case we start with character 1 and end with character 2.

Make sure that you are comfortable with what we are doing with this code.

**Plotting**

To understand data we may use summary statistics of the type that have been considered already within this session. But as Anscombe (1973) argues, it is essential to actually look at your data before modelling. In this section we will look at the plotting options within R.



As a first exercise we use the plot() command to construct a plot in base R. The command used is:

plot(citiesdata$ptrans,citiesdata$lives,xlab="Public Transport",ylab="Satisfaction")

Graphical user interface, text, application

Description automatically generatedThe elements here are the horizontal axis variable, the vertical axis variable, the label used for the horizontal axis and the label used for the vertical axis.

Once you have generated a plot you can save the plot by right clicking on the plot. There is an option to save as.. and there is also the option to copy your output to the clipboard. The menu that appears when you right click on your plot can be seen on the left. R has commands for outputting files but we will leave these for a more advanced class. Note that many academic journals have particularly specifications about file types and R can make most of them.

We can add more elements to the command, separating each by a comma. Attempt the following exercises:

1. Add the option pch=16 to the command. What happens?
2. Add the option color=”blue” to the command you created in 9. What happens?
3. Draw a plot of affordability (afford) and the live satisfaction variable (livesa). Use pch=15 and color=”red” in your plot. Make sure that the axes have the correct names.
4. Draw a plot of green spaces (green) and public squares (public). Use blue solid circles for the points. Make sure that your plot has the correct axis labels.

**Plotting with ggplot2**

R can produce the majority of data visualisations which you can think of, but the presentation is basic. There are other graphics packages to help produce better looking plots. In this session we are considering the most commonly used additional graphics package, ggplot2. As we have already installed the tidyverse the ggplot2 package has already been loaded into the R environment. If not you could use library(ggplot2) to load the package.

Graphical user interface, text, application

Description automatically generatedThe plot on the left shows a scatterplot for public transport and the overall life satisfaction. This plot may be directly compared to the example constructed using base R. The biggest difference is that in the ggplot the points are solid as we saw with pch=16 rather than hollow (pch=1) in the base R plots.

A screenshot of a computer

Description automatically generatedGraphical user interface, text, application

Description automatically generatedThe command which produces the plot is a little more complicated than the basic plot, but is intuitive.

The saving process is initiated with a right click on the plot and selection of Save as…

The screenshot on the left shows how you can select the file type and set the name. Here I have called the file Basic ggplot ptrans but you should make sure that you use a name that you will understand when looking at it later.

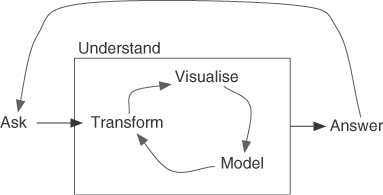
Graphical user interface, text, application

Description automatically generated

Once saved the object appears within the folder. Here you can see the Basic ggplot ptrans is on the list of files above the citiesdata.csv file.

*Here we have used the basic settings. In journals you may be asked to provide higher quality images. To create a higher resolution image you need to adjust the settings of the plot functions and/or the saving code. These elements are beyond the scope of this course, but some code is included on the GitHub file for your reference.*

**Summary of First Half**

It is anticipated that the lunch break will be taken after the construction of the basic plots in ggplot. The figure on the left is from Wickham (2011) and shows the data analysis cycle. We have seen how to transform data and do some basic visualisation. In the afternoon session we will continue the visualisation and transformation cycle.

Over the lunchtime period give some thought to the following question:

**What makes a city a satisfying place to live for its residents?**

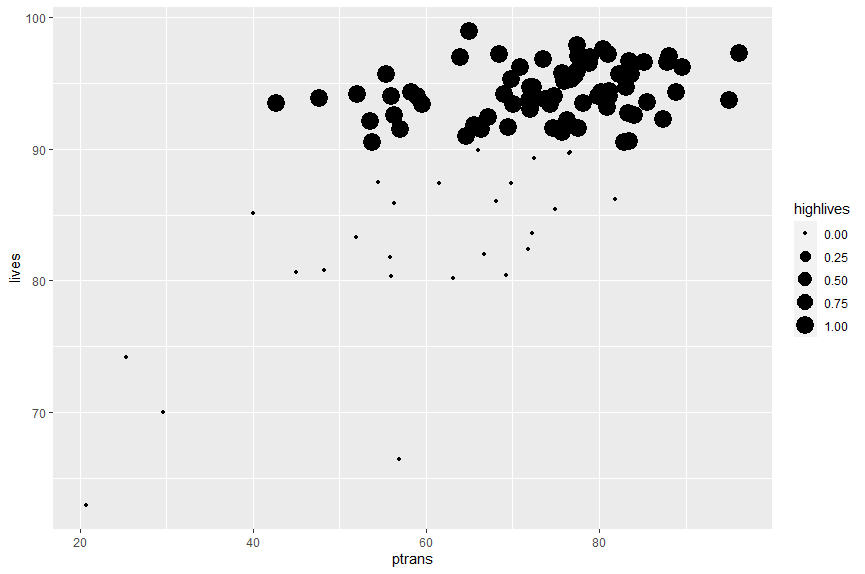
The process of getting from Ask to Answer will be the focus of our afternoon.

**Plotting Continued**

The morning sessions showed how to make a simple scatter plot using the ggplot2 package. In this section we are looking at some further plotting functionality from within the ggplot2 package.

|  |  |  |
| --- | --- | --- |
| ggplot(data=citiesdata) + | | |
|  | geom\_point(mapping=aes(x = ptrans, y = lives, size = highlives)) | |
|  | |  |

As a first step we adjust the size of the plot to set the size according to whether or not satisfaction is greater than 90%, that is more than 90% of respondents say that they are satisfied, or very satisfied with living in the city.

****The plot on the left is generated by the code on the previous page. We can see that the effect of the 90 threshold means that all of the dots above the 90 gridline are large. This plot is not very informative at this stage. We can see that there is a wide range of satisfactions with public transport amongst the cities with satisfactions above 90%

To introduce ggplot we have used a single line. However, the plots are not ideal because the axis labels are still just the names of the variables. For presentation reasons a plot should have axis labels which are more meaningful. Let us change the axis labels using the labels() command

|  |  |
| --- | --- |
| ggplot(data=citiesdata) + | |
|  | geom\_point(mapping=aes(x=ptrans, y = lives, size = highlives))+ |
|  | labs(x = "Public Transport", y = "Life Satisfaction") |

A picture containing text, screenshot, diagram

Description automatically generated

The new plot is shown above. Although the inference is still the same, it is helpful to see the correctly labelled axes.

The initial question for this session asked us to think about how the variables in our dataset are related to the satisfaction variable. We get some inference from the scatterplots with life satisfaction on the vertical axis.

1. Produce a correctly labelled graph with satisfaction with green spaces greens on the horizontal axis. What is the inference from your plot?
2. Produce a correctly labelled graph with the proportion of people who agree, or strongly agree, that their area is affordable, afford, on the horizontal axis. Size the points according to the highlives dummy variable
3. Produce a correctly labelled graph with the proportion of people who are satisfied, or very satisfied, with the level of cleanliness in their city, cleans, on the horizontal axis. The vertical axis should be the proportion of individuals who are satisfied and very satisfied with living in their town. Size the points according to affordability, afford.

We now continue to think about what happens when we use two of the potential explanatory variables on the axes. In this case we will use the satisfaction with public transport, ptrans, and the proportion who feel that their area is affordable, afford, on the vertical axis. Again we will add code to ensure that the axes are correctly labelled.

|  |  |
| --- | --- |
| ggplot(data=citiesdata) + | |
|  | geom\_point(mapping=aes(x=ptrans, y = lives, size = highlives))+ |
|  | labs(x = "Public Transport", y = "Affordability of Houses") |

A picture containing screenshot, text, diagram

Description automatically generated

Our graph now shows that the areas with higher satisfaction than 90% cover the full range of affordability. There are some small dots with high affordability and high satisfaction with public transport. Likewise, we see a large point with low satisfaction with public transport and low affordability. Seeing the data in this way suggests that there is limited linkage between the two variables and life satisfaction.

To convey more information we have to introduce colour. Here we can use the country variable which was created in the first part of the session. Notice now that the code is much longer and therefore the second line actually fits across two lines in this document.

|  |  |
| --- | --- |
| ggplot(data=citiesdata) + | |
|  | geom\_point(mapping = aes(x = ptrans, y = lives, size = highlives, color = country)) + |
|  | labs(x = "Public Transport", y = "Affordability of Houses") |

*When you use code in ggplot it is necessary to end every line with a + symbol, except the last line. You may think of this as being adding another thing onto the plot through the subsequent line*

A picture containing screenshot, colorfulness, text

Description automatically generated

Although the colours allocated to each country are different there are many similarities and so it is hard to really know exactly which point is from which country. We may resolve the issue partially by switching to dummies for countries that we are interested in.

1. Create a dummy variable for the country being Italy. The relevant condition here is citiesdata$country==”IT”
2. Plot the graph again with public transport on the horizontal axis and affordability on the vertical axis. Colour the points according to whether or not the city is in Italy. What do you notice about Italian cities?
3. Repeat questions 16 and 17, but this time consider cities in the UK. The relevant condition here is citiesdata$country==”UK”

**Bar Charts**

Scatterplots are useful when we want to understand two continuous variables, but many variables we encounter in our datasets are not continuous. In the dataset used in this session we have the country variable which is not continuous. As a first step let us construct a simple bar chart where the height of the bars is the number of cities in each country.

|  |  |
| --- | --- |
| ggplot(data=citiesdata) + | |
|  | geom\_bar(mapping = aes(x = country))+ |
|  | labs(x = "Country", y = "Number of Cities") |

A picture containing text, screenshot, font, plot

Description automatically generated

In this case we see that the majority of cities are from Germany (DE) and then the second highest number is Italy (IT). You should check back to the graph you produced in question 17 to make sure that the number of points on the graph matches the height of the bar here.

We can colour the bar charts according to the country by adding a colour argument to the code in the same way as before.

|  |  |
| --- | --- |
| ggplot(data=citiesdata) + | |
|  | geom\_bar(mapping = aes(x = country , fill=country))+ |
|  | labs(x = "Country", y = "Number of Cities") |

A picture containing text, screenshot, plot, diagram

Description automatically generated

The impact of using the country is to produce a rainbow-like colouration. The information provided by the colour is redundant since there is a label for each country in any case. In your own datasets there may be more obvious ways to colour the data.

In some cases we may not want to see the legend. You can drop the legend by adding a show.legend=FALSE into the arguments for the geom\_bar or the geom\_plot.

|  |  |
| --- | --- |
| ggplot(data=citiesdata) + | |
|  | geom\_bar(mapping = aes(x = country , fill=country), show.legend = FALSE)+ |
|  | labs(x = "Country", y = "Number of Cities") |

1. Produce another version of the bar chart where the colouration is based upon the Italy dummy that you created in question 16. Why might you want to produce a plot like this?
2. Produce another version of the bar chart where France (FR) is the bar which has a different colour to the other plots

We can further divide the bars to show another variable. In this case let us show the proportion of cities in each country which feature in our high life satisfaction group.

|  |  |
| --- | --- |
| ggplot(data=citiesdata) + | |
|  | geom\_bar(mapping = aes(x = country , fill=highlives, group=highlives), show.legend = FALSE)+ |
|  | labs(x = "Country", y = "Number of Cities") |

In this case we are setting the fill equal to the highlives variable but we are also grouping by highlives. To be used suitable as a group variable we need a variable with a limited number of levels. Our highlives variable only has two levels. The legend is turned off since we know that light blue is high and dark blue is low. We could leave the legend there or add the description in the caption.

A picture containing screenshot, diagram, plot, line

Description automatically generated

From this plot we see that the UK and Netherlands are both 100% in the high group. An interesting question which emerges here is which city in Germany did not reach the high group. We return to the question shortly.

**Arranging Data**

To sort the data we use the function arrange() and input the variable as the second argument. For example to arrange citiesdata by the lives variable we would write:

arrange(citiesdata,lives)

The data is rearranged in the console and so here the output is copy pasted. The other option within R studio is to use the column and click on the name of the variable in the header.

CODE NAME ptrans health cultfc greens public cleans trusts safety afford lives

IT005C Palermo 20.7 36.6 59.8 34.1 46.4 7.6 52.1 50.9 56.6 62.9

TR012C İstanbul 56.9 62.8 56.6 55.8 56.0 61.8 28.4 49.7 21.2 66.4

The default is to arrange in ascending order. Therefore if you wish to arrange in descending order we need to specify desc() around the variable name

arrange(citiesdata,desc(lives))

CODE NAME ptrans health cultfc greens public cleans trusts safety afford lives

EE002C Tartu 65.0 84.0 93.0 88.0 90.0 87.0 80.0 70.0 29.0 99.0

DK001C København 77.5 78.9 84.2 89.9 80.3 70.8 87.2 93.4 8.8 97.9

SE001C Stockholm 80.4 78.5 88.5 90.5 86.1 70.8 83.6 90.6 9.0 97.6

1. Verify that the outputs on the previous page match the results you get when clicking in the citiesdata window at the top left of the RStudio environment
2. Which German city does not have a satisfaction value above 90%?

**Summarising Data**

The argument of Anscombe (1973) is that researchers should give more attention to the visualisation of data because there is often insufficient information in the summary statistics alone. We looked at the summary statistics for all data, and we have seen the visualisation of data using ggplot2. Now we will look at the construction of simple summary statistics using functions within the tidyverse.

There are two key elements for us to consider:

* A grouping variable to construct the summaries – use group\_by()
* The functions which we want to apply to the data – use within summarise()

Let us compute some simple averages grouped by each country.

by\_country<-group\_by(citiesdata,country)

clives<-summarise(by\_country,mlives=mean(lives,na.rm=TRUE))

A simple code is provided here which first states that we want to group our data by country. This is then stored as the object by\_country . Then we produce the summary and store the summary in the clives object. Both by\_country and clives are names provided here to help remember what the variable is actually doing. Remember that under the principles of the tidyverse all object names should be short and intuitive about what they are doing.

A screenshot of a computer

Description automatically generated

Running the code does not change much on the display, but you should now see that there is a by\_country and a clives object in the Environment. To see the summary, we can click on the clives in the top right window.

1. Which country has the lowest average satisfaction?
2. Which country has the highest average satisfaction?

The function in the first example is mean() but we can use many others. We can add sd() for standard deviation, quantile() to identify quantiles of the distribution, min() and max() for minimum and maximum respectively. Quantiles need you to state the quantile and variable.

At the top of the next page is the code for using all of these functions to make a more familiar summary table complete with quantiles within each country. Note here that some countries only have a few entries and as such the quantiles are meaningless. Where there is only one city standard deviation cannot be computed.

clives2<-summarise(by\_country,obs=n(),mlives=mean(lives,na.rm=TRUE),sdlives=sd(lives,na.rm=TRUE),minlives=min(lives,na.rm=TRUE),q25lives=quantile(lives,0.25,na.rm=TRUE),q50lives=quantile(lives,0.50,na.rm=TRUE),q75lives=quantile(lives,0.75,na.rm=TRUE),maxlives=max(lives,na.rm=TRUE))

A screenshot of a computer

Description automatically generated with medium confidence

We can see that in Austria the average level of satisfaction is 92.5%, but there are only two cities. Germany, in row 7, has an average satisfaction of 94.4% and a lower quartile of 93.1%. We know that there is one city below 90 and can see from our summary that the minimum is actually 89.9%.

1. Produce summary statistics for the satisfaction with public transport in each country
2. Produce summary statistics for the satisfaction with green spaces in each country
3. Produce a grouping variable according to whether a city has a satisfaction above 90%
4. Produce averages for each of the percentage variables (those shown in the table on page 2) grouped by whether or not the city has a satisfaction above 90%.

We can make many permutations of the grouping variable, summary statistics and variables being summarised. Questions 25 to 28 simply represent some examples of what can be done.

**Summary of Session**

This session has acted as an introduction to R. We have seen how to load data, load packages, applied plotting functions, performed some basic coding tasks and then constructed summaries of groups. Much of what we have done has used the tidyverse. The final part of the session allowed us to complete the cycle proposed by Wickham (2014) to understand what factors are associated with higher life satisfaction. As you progress with R you will learn more functions and ways to work more efficiently. There is a wealth of opportunity provided by packages for R