Exploring the Social World - Quantitative Block: Statistics

Gabriele Filomena and Zi Ye

2025-09-22

Table of contents

Welcome **3**
Contact . . . 3

Overview **4**
Aim and Learning Objectives . . . 4
Module Structure . . . 5
Software and Data . . . 6

Assessment **7**
Required Report Structure . . . 7
How to get there? . . . 8
How is it graded? . . . 9

Assessment: How to submit **18**

1 Lab: Introduction to R for Statistics **20**
1.1 R? . . . 20
1.2 R(Studio) Basics . . . 21
1.2.1 Starting a session in RStudio . . . 21
1.2.2 Using the console . . . 24
1.2.3 R as a simple calculator . . . 24
1.2.4 Variables Assignment . . . 26
1.2.5 Working with Scripts . . . 27
1.2.6 R Packages . . . 28
1.3 Practice: Dataset and Dataframes . . . 29
1.3.1 Datasets in R . . . 31
1.3.2 Importing Data in R . . . 32
1.4 Practice: Descriptive Statistics . . . 35
1.4.1 Summarizing Data . . . 35
1.4.2 Understanding the Structure of the FRS Datafile . . . 42
1.4.3 Explore the Distribution of Your Outcome Variable . . . 43

Welcome

This is the website for "Exploring the Social World - Quantitative Block: Statistics" (module **ENVS225**) at the University of Liverpool. This block of the module is designed and delivered by Dr. Gabriele Filomena and Dr. Zi Ye from the Geographic Data Science Lab at the University of Liverpool. The module seeks to provide hands-on experience and training in introductory statistics for human geographers.

The website is **free to use** and is licensed under the Attribution-NonCommercial-NoDerivatives 4.0 International. A compilation of this web course is hosted as a GitHub repository that you can access:

- As an html website.
- As a GitHub repository.

Contact

Gabriele Filomena - gfilo [at] liverpool.ac.uk Lecturer in Geographic Data Science Office 1xx, Roxby Building, University of Liverpool - 74 Bedford St S, Liverpool, L69 7ZT, United Kingdom.

Zi Ye - zi.ye [at] liverpool.ac.uk Lecturer in Geographic Information Science Office 107, Roxby Building, University of Liverpool - 74 Bedford St S, Liverpool, L69 7ZT, United Kingdom.

Overview

Aim and Learning Objectives

This sub-module aims to provide training and skills on a set of basic quantitative research methods for data collection, analysis, and interpretation. You will learn how to define coherent, relevant research questions, utilise various research quantitative methods, and identify appropriate methodologies to tackle your research questions. **This block serves as the foundation for the dissertation and fieldwork modules.**

Background

Data and research are key pillars of the global economy and society today. We need rigorous approaches to collecting and analysing both the statistics that can tell us 'how much' and if there are observable relationships between phenomena; and the information gives us a nuanced understanding of cultural contexts and human dynamics. Quantitative skills enable us to explore and measure socio-economic activities and processes at large scales, while qualitative skills enable understanding of social, cultural, and political contexts and diverse lived experiences. Rather than being in opposition, qualitative and quantitative research can complement one another in the investigation of today's pressing research questions.

To these ends, this block will help you develop your quantitative (statistical) skills, as critical tools. This course will help you understand what quantitative statistical researchers use and develop a set of research techniques that can be used in your field classes and dissertations.

Learning objectives:

- Understand how to explore a dataset, containing a number of observations described by a set of variables.
- Demonstrate an understanding in the application and interpretation of commonly used quantitative research methods.
- Demonstrate an understanding of how to work with quantitative data to address real-world research questions.

Module Structure

Staff: Dr Zi Ye and Dr Gabriele Filomena

Where and When

Week 1, 2, 4, 6 Tuesday @ Central Teaching Hub PCTC

- Lecture (10 – 11 am).
- Practical PC session (11 am – 1 pm).

Week 3

- Lecture (3 – 4 pm) Thursday @ Central Teaching Hub – Lect. Theatre C.
- Practical PC session (9 – 11 am) Friday @ Central Teaching Hub PCTC.

Week 5

- Lecture (3 – 4 pm) Thursday @ Central Teaching Hub – Lect. Theatre C.
- Practical PC session (10 – 12 am) Friday @ Central Teaching Hub PCTC.

Lectures will introduce and explain the fundamentals of quantitative methods, with the opportunity to apply the method introduced in the labs later in the week.

The computer practical sessions, will give you the chance to use and apply quantitative methods to real-world data. These are primarily self-directed sessions, but with support on hand if you get stuck. Support and training in R will be provided through these sessions. Weekly sessions will be driven by empirical research questions.

Week	Topic	Format	Staff
1	Introduction & Review	Lecture and Computer Lab Practical	GF
2	Single & Multiple Linear Regression	Lecture and Computer Lab Practical	GF
3	Multiple Linear Regression with Categorical Variables	Lecture and Computer Lab Practical	ZY
4	Logistic Regression	Lecture and Computer Lab Practical	ZY
5	Data Visualisation	Lecture and Computer Lab Practical	GF
6	Summary and Assessment Support	Lecture and Computer Lab Practical	ZY

Software and Data

For quantitative training sessions, ensure you have installed and/or have access to **RStudio**. To run the analysis and reproduce the code in R, you need the following software installed on your machine:

- R-4.2.2 (or later)
- RStudio 2022.12.0-353 (or later)

To install and update:

- R, download the appropriate version from The Comprehensive R Archive Network (CRAN).
- RStudio, download the appropriate version from here.

This software is already installed on University Machines. But you will need it to run the analysis on your personal devices.

Data

Example datasets could be accessed through Canvas or (some) on GitHub Repository of the module. These include:

- 2021 UK Census Data.
- 2021 Annulation Population Survey (APS) - only on Canvas.
- 2016 Family Resource Survey (FRS) - only on Canvas.
- 2011 Sample of Anonymised Records (SAR).

Note: The Annual Population Survey requires the completion of a quiz prior to its usage, as it is licensed.

Assessment

Deadline: Monday 3rd November 2025 - 2pm. **Word count**: 2000 words - including tables, excluding references.

The assignment **Data Exploration and Analysis** consists of writing a research report using one of the regression techniques learned during the module. The basic idea is to put in practice the methods learned during the quantitative block of the module. You are required to apply a linear or logistic regression model to the data provided for the module. The report needs to include the following sections (in brackets, % of the whole lenght):

- Introduction (5%).
- Literature Review (20%).
- Methods and data (30%).
- Results and discussion (40%).
- Conclusion (5%).
- Reference List.

Required Report Structure

1. **Introduction**
    - Context: Why is the topic relevant or worth being investigated?
    - Brief discussion of existing literature.
    - Knowledge gap and Aim.
    - 1 Research question.
2. **Literature review**
    - More detailed Literature review, i.e. what do we already know about this subject
    - Rationale for including certain predictor variables in the model.
    - What knowledge gap remains that this article will address? (includes "not studied before in this area"). *Note: there is no expectation on totally original research. The focus is on a clean, sensible, data analysis situated in existing ideas.*
3. **Methodology:**
    - A brief introduction to the dataset being analysed (who collected it? When? How many responses? etc.)

- A description of the variables chosen to be analysed.
- A description of any transformation made to the original data, i.e. turning a continuous variable of income into intervals, or reducing the number of age groups from 11 to 3.
- A description and justification of the statistical techniques used in the subsequent analysis (i.e. the Multivariate regression model: Multiple or Logistic Linear Regression).

4. **Results and Discussion**

   - Descriptive statistics and summary of the variables employed.
   - Correct interpretation of correlation coefficients.
   - Usage and results of an appropriate multivariate regression model.
   - Interpretation of the results, including links and contrasts to existing literature.
   - Selective illustrations (graphs and tables) to make your findings as clear as possible.

5. **Conclusion**

   - Summary of main findings.
   - Limitations of study (self-critique).
   - Highlight any implications derived from the study.

Follow this structure and include **ALL** these points, do not make your life harder.

How to get there?

The first stage is to identify **ONE** a relevant research question to be addressed. Based on the chosen question, you will need to identify a dependent (or outcome) variable which you want to explain, and at least two relevant independent variables that you can use to explain the chosen dependent variable. The selection of variables should be informed by the literature and empirical evidence.

To detail in the Methods Section: Once the variables have been chosen, you will need to describe the data and **appropriate** type of regression to be used for the analysis. You need to explain any transformation done to the original data source, such as reclassifying variables, or changing variables from continuous to nominal scales. You also need to briefly describe the data use: source of data, year of data collection, indicate the number of records used, state if you are using individual records or geographical units, explain if you are selecting a sample, and any relevant details. You also need to identify type of regression to be used and why.

To detail in the Results and Discussion Section: Firstly, you need to provide two types of analyses. First, you need to provide a descriptive analysis of the data. Here you could use tables and/or plots reporting relevant descriptive statistics, such as the mean, median and standard deviation; variable distributions using histograms; and relationships between

variables using correlation matrices or scatter plots. Secondly, you need to present an estimated regression model or models and the interpretation of the estimated coefficients. You need a careful and critical analysis of the regression estimates. You should think that you intend to use your regression models to advice your boss who is expecting to make some decisions based on the information you will provide. As part of this process, you need to discuss the model assessment results for the overall model and regression coefficients. Remember to substantiate your arguments using relevant literature and evidence, and present results clearly in tables and graphs.

How is it graded?

Grade	Score Range	UG	Descriptor	Assignment Expectations
Fail	0-34%	Fail	**Inadequate**	**Literature Review**: Lacks relevance and fails to justify variable choice. Evidence is irrelevant or missing, providing no support to the research question. **Methods**: Data is not described, and the regression model is entirely missing. No appropriate statistical method is applied. **Results and Discussion**: No descriptive statistics, graphs, or tables are provided. Model results and interpretation are absent. **Structure and References**: Report is disorganized with significant referencing and citation errors throughout.

Grade	Score Range	UG	Descriptor	Assignment Expectations
Narrow Fail	35-39%	Fail	**Highly Deficient**	**Literature Review**: Review is present but lacks coherence and fails to justify variable choice. Evidence is poorly aligned with the research question and mostly irrelevant. **Methods**: Minimal data description; the regression model is missing but some statistical methods are mentioned. **Results and Discussion**: Few or no descriptive statistics or visuals are present. Statistical methods are unclear or incorrectly applied. Results are vague and lack meaningful interpretation. **Structure and References**: Report structure is poor, with referencing errors in multiple sections.

Grade	Score Range	UG	Descriptor	Assignment Expectations
Third / Fail	40-49%	Third (UG)	**Deficient**	**Literature Review**: Relevant literature is partially addressed but lacks depth, with limited justification for variable choice. Evidence is minimally aligned with the research question. **Methods**: A very basic data description is provided, but the selected regression model is deeply inadequate or incorrect (e.g., multiple linear regression for a categorical outcome; logistic regression for a continuous outcome). **Results and Discussion**: Descriptive statistics or visuals may be present but insufficient. Model results are presented with little to no interpretation. **Structure and References**: Report structure is present but lacks clarity, with inconsistencies

Grade	Score Range	UG	Descriptor	Assignment Expectations
2.2 / Pass	50-59%	2.2 (UG)	**Adequate**	**Literature Review**: Addresses relevant literature but with limited justification of variable choices. Evidence generally supports the research question but lacks detail. **Methods**: Data description is present but brief; a regression model is included but applied illogically or incorrectly (e.g., multiple linear regression for a categorical outcome; logistic regression for a continuous outcome) and with little explanation. **Results and Discussion**: Basic descriptive statistics, graphs, or tables are presented; the regression model is applied with some inaccuracies and/or interpretation is minimal. **Structure and References**: Report is mostly organized

Grade	Score Range	UG	Descriptor	Assignment Expectations
2.1 / Merit	60-69%	2.1 (UG)	**Good**	**Literature Review**: Relevant literature is discussed, with some justification for variable choice. Evidence supports the research question well. **Methods**: Data is described with some detail, though potential data transformations are under-explored. The regression model is appropriate for the selected variable types. **Results and Discussion**: Descriptive statistics and visuals are provided. Model results are discussed, though interpretation lacks depth. Findings are compared to existing literature. **Structure and References**: Report is logically structured and clear, with mostly correct citations.

Grade	Score Range	UG	Descriptor	Assignment Expectations
First / Distinction	70-79%	First (UG)	**Very Good**	**Literature Review**: Strong grasp of relevant literature, with well-justified variable selection. Evidence aligns well with the research question. **Methods**: Data is comprehensively described with consideration of relevant transformations. The regression model is appropriate and well-justified. **Results and Discussion**: Descriptive statistics and clear visuals support findings. Model results are accurately interpreted with strong connections to existing literature. **Structure and References**: Report has a coherent, professional structure with only minor referencing errors.

Grade	Score Range	UG	Descriptor	Assignment Expectations
High First / High Distinction	80-100%	High First (UG)	**Excellent to Outstanding**	**Literature Review**: Critical and thorough literature review with strong, well-justified variable selection. Evidence fully supports the research question with insightful connections. **Methods**: Detailed data description and transformation steps are clearly articulated. Regression model is expertly applied and justified. **Results and Discussion**: Comprehensive descriptive statistics, graphs, and tables are provided. Model results are innovatively interpreted with strong links to existing research. **Structure and References**: Report is professionally structured, with flawless citations and a high standard of organization.

In summary:

1. **Introduction**: Should establish the topic's relevance, present a concise literature overview, identify a knowledge gap, and outline research questions.
2. **Literature Review**: Requires an in-depth review of relevant studies, justification for chosen independent variables, and identification of a potential knowledge gap or unexplored area aligned with the chosen research question.
3. **Methods and Data**: Should describe the dataset, variable transformations, and justify the regression technique. Key transformations, such as reclassifying variables, should be explained with clarity and relevance.
4. **Results and Discussion**: Involves presenting descriptive statistics, followed by a clear regression analysis. Discussion should interpret results, compare findings with existing literature, and include meaningful tables and graphs.
5. **Conclusion**: Summarize findings, discuss limitations, and suggest future directions.
6. **Referencing**: Requires correct and consistent citations and a well-structured reference list.

Employing a novel dataset, i.e. not employed during the practical sessions, for the assignment will be awarded with a higher grade. For example, see other quantitative dataset from Secondary datasets for Human Geography.

Assessment: How to submit

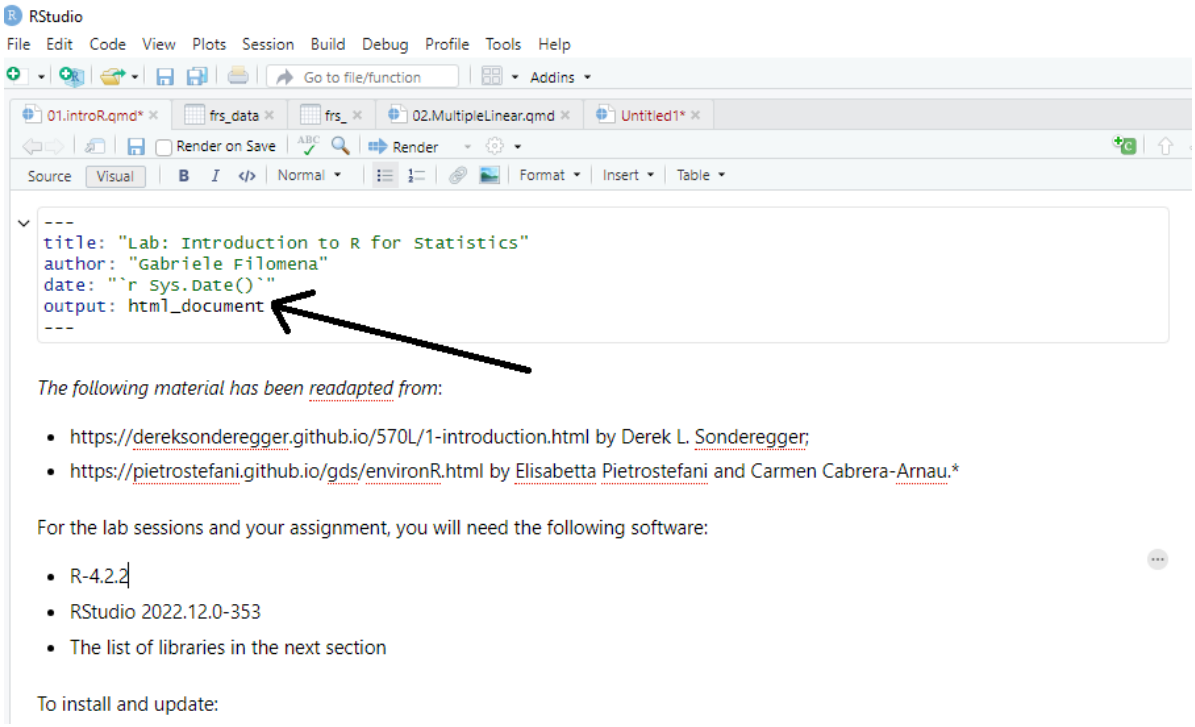
You must submit a `.pdf` file, that is a rendered version of a Quarto Markdown file (`qmd` file). This will allow you to write a research paper that also includes your working code, without the need of including the data (rendered `.qmd` files are executed before being converted to R).

How to get a PDF?

1. **Install Quarto**: Make sure you have Quarto installed. You can download it from quarto.org.
2. **LaTeX Installation**: For PDF output, you'll need a LaTeX distribution like **TinyTeX** from R, by executing this in the R console:

```
install.packages("tinytex")
tinytex::install_tinytex()
```

3. **Open the Quarto File**: Open your `.qmd` file in RStudio.
4. **Set Output Format**: In the YAML header at the top of your Quarto file, specify `pdf` under `format`:




```
title: "Your Document Title"
author: "Anonymous" # do not change
format: pdf
```

5. Click the **Render** button in the RStudio toolbar (next to the Knit button).

1 Lab: Introduction to R for Statistics

The following material has been readapted from:

- https://dereksonderegger.github.io/570L/1-introduction.html by Derek L. Sonderegger;
- https://pietrostefani.github.io/gds/environR.html by Elisabetta Pietrostefani and Carmen Cabrera-Arnau

The lecture's slides can be found here.

Before completing the practical, please take this quiz.

For the lab sessions and your assignment, you will need the following software:

- R-4.2.2 (or higher)
- RStudio 2022.12.0-353 (or higher)
- The list of libraries in the next section

To install and update:

- R, download the appropriate version from The Comprehensive R Archive Network (CRAN)
- RStudio, download the appropriate version from Posit

1.1 R?

R is an open-source program that is commonly used in Statistics. It runs on almost every platform and is completely free and is available at www.r-project.org. Most of the cutting-edge statistical research is first available on R.

R is a script based language, so there is no point and click interface. While the initial learning curve will be steeper, understanding how to write scripts will be valuable because it leaves a clear description of what steps you performed in your data analysis. Typically you will want to write a script in a separate file and then run individual lines. This saves you from having to retype a bunch of commands and speeds up the debugging process.

1.2 R(Studio) Basics

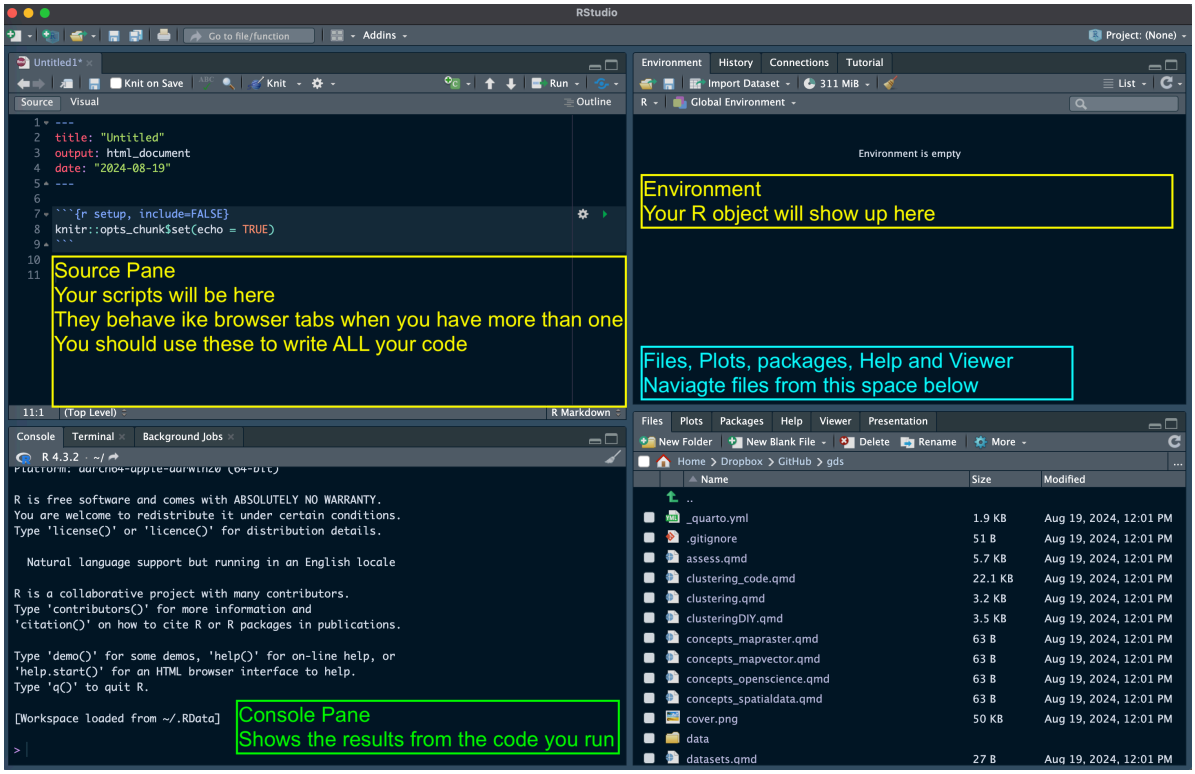
We will be running R through the program RStudio which is located at rstudio.com. When you first open up RStudio the console window gives you some information about the version of R you are running and then it gives the prompt `>`. This prompt is waiting for you to input a command. The prompt + tells you that the current command is spanning multiple lines. In a script file you might have typed something like this:

```
for( i in 1:5 ){
    print(i)
}
```

Finding help about a certain function is very easy. At the prompt, just type `help(function.name)` or `?function.name`. If you don't know the name of the function, your best bet is to go the the web page www.rseek.org which will search various R resources for your keyword(s). Another great resource is the coding question and answer site stackoverflow.

1.2.1 Starting a session in RStudio

Upon startup, RStudio will look something like this.



Note: the **Pane Layout** and **Appearance** settings can be altered:

- on Windows by clicking RStudio>Tools>Global Options>Appearance or Pane Layout
- on Mac OS by clicking RStudio>Preferences>Appearance or Pane Layout.

You will also have a standard white background; but you can choose specific themes.

Source Panel (Top-Left)

This is where you write, edit, and view scripts, R Markdown/Quarto documents, or R scripts. It allows:

- Editing Scripts: Write and edit R scripts or documents (`.R`, `.Rmd`, `.qmd`).
- Executing the Code: Run lines, blocks, or the entire script directly from the editor.

Console Panel (Bottom-Left)

The Console is the main place to run R commands interactively. It allows:

- Executing the Code: Type and run R commands directly.
- Viewing outputs, warnings, and errors for immediate feedback.
- Browsing and reusing past commands (History Tab).
- Toggling between the R Console, and the Terminal (yuo don't really need the latter).

Environment Panel (Top-Right)

This panel helps track variables, functions, and the history of commands used. It contains:

- Environment Tab: Shows all current variables, datasets, and objects in your session, including their structure and values.
- History Tab: Provides a record of past commands. You can re-run or move commands to the console or script.

Files / Plots / Packages / Help Panel (Bottom-Right)

This multifunctional panel is for file navigation, plotting, managing packages, viewing help, and managing jobs. It contains:

- Files Tab: Navigate, open, and manage files and directories within your project.
- Plots Tab: Displays plots generated in your session. You can export or navigate through multiple plots here.
- Packages Tab: Lists installed packages and allows you to install, load, and update packages.
- Help Tab: Displays help documentation for R functions, packages, and other resources. You can search for documentation by typing a function or package name.

Important: Unless you are working with a script, you will be likely writing code on the console.

At the start of a session, it's good practice clearing your R environment (console):

```
rm(list = ls())
```

In R, we are going to workin with **relative paths**. With the command `getwd()`, you can see where your working directory is currently set.

```
getwd()
```

For ENVS225, download the material of the module an unzip it whever you like.

The folder structure should look like:

```
stats/
   data/
   labs_img/
   labs/
```

You can delete other sub-folders (e.g. docs).

This should be on your personal computer. On a local machine, I suggest using the directory M: to store the folder, it can then be accessed from every other computer on campus.

Then, in R Studio - on Windows by clicking *RStudio>Tools>Global Options>General.*. - on Mac OS by clicking *RStudio>Preferences>Appearance* or Pane Layout>browse and set the folder you just created as your working directory.

Restart Rstudio and check if that has been applied.

```
getwd()
```

File paths in R work like this:

File Path	Description
`MyFile.csv`	Look in the working directory for `MyFile.csv`.

File Path	Description
`MyFolder/MyFile.csv`	In the working directory, there is a subdirectory called `MyFolder` and inside that folder is `MyFile.csv`.

You do not need to set your working directory if you are using an R-markdown or Quarto document and you have it saved in the right location. The pathway will start from where your document is saved.

1.2.2 Using the console

Try to use the console to perform a few operations. For example type in:

```
1+1
```

```
[1] 2
```

Slightly more complicated:

```
print("hello world")
```

```
[1] "hello world"
```

If you are unsure about what a command does, use the "Help" panel in your Files pane or type `?function` in the console. For example, to see how the `dplyr::rename()` function works, type in `?dplyr::rename`. When you see the double colon syntax like in the previous command, it's a call to a package without loading its library.

1.2.3 R as a simple calculator

You can use R as a simple calculator. At the prompt, type 2+3 and hit enter. What you should see is the following

```
# Some simple addition
2+3
```

```
[1] 5
```

In this fashion you can use R as a very capable calculator.

```
6*8
```

```
[1] 48
```

```
4^3
```

```
[1] 64
```

```
exp(1)   # exp() is the exponential function
```

```
[1] 2.718282
```

R has most constants and common mathematical functions you could ever want. For example, the absolute value of a number is given by `abs()`, and `round()` will round a value to the nearest integer.

```
pi     # the constant 3.14159265...
```

```
[1] 3.141593
```

```
abs(1.77)
```

```
[1] 1.77
```

Whenever you call a function, there will be some arguments that are mandatory, and some that are optional and the arguments are separated by a comma. In the above statements the function `abs()` requires at least one argument, and that is the number you want the absolute value of.

When functions require more than one argument, arguments can be specified via the order in which they are passed or by naming the arguments. So for the `log()` function, for example, which calculates the logarithm of a number, one can specify the arguments using the named values; the order woudn't matter:

```
# Demonstrating order does not matter if you specify
# which argument is which
log(x=5, base=10)
```

```
[1] 0.69897
```

```
log(base=10, x=5)
```

```
[1] 0.69897
```

When we don't specify which argument is which, R will decide that `x` is the first argument, and `base` is the second.

```
# If not specified, R will assume the second value is the base...
log(5, 10)
```

```
[1] 0.69897
```

```
log(10, 5)
```

```
[1] 1.430677
```

When we want to specify the arguments, we can do so using the `name=value` notation.

1.2.4 Variables Assignment

We need to be able to assign a value to a variable to be able to use it later. R does this by using an arrow `<-` or an equal sign `=`. While R supports either, for readability, I suggest people pick one assignment operator and stick with it.

Variable names cannot start with a number, include spaces, and they are case sensitive.

```
var <- 2*7.5       # create two variables
another_var = 5   # notice they show up in 'Environment' tab in RStudio!
var
```

```
[1] 15
```

```
var * another_var
```

```
[1] 75
```

As your analysis gets more complicated, you'll want to save the results to a variable so that you can access the results later. *If you don't assign the result to a variable, you have no way of accessing the result.*

1.2.5 Working with Scripts

This is what you are going to do for the practical sessions. Avoid using the Console, if not for quick calculations

R Scripts (.R files)

Traditional script files look like this:

```
# Problem 1
# Calculate the log of a couple of values and make a plot
# of the log function from 0 to 3
log(0)
log(1)
log(2)
x <- seq(.1,3, length=1000)
plot(x, log(x))

# Problem 2
# Calculate the exponential function of a couple of values
# and make a plot of the function from -2 to 2
exp(-2)
exp(0)
exp(2)
x <- seq(-2, 2, length=1000)
plot(x, exp(x))
```

In RStudio you can create a new script by going to `File -> New File -> R Script`. This opens a new window in RStudio where you can type commands and functions as a common text editor.

This looks perfectly acceptable as a way of documenting what one does, but this script file doesn't contain the actual results of commands you ran, nor does it show you the plots. Also anytime you want to comment on some output, it needs to be offset with the commenting character `#`. It would be nice to have both the commands and the results merged into one document. This is what the R Markdown file does for us.

R Markdown (`.Rmd` and `.qmd` files)

The R Markdown is an implementation of the Markdown syntax that makes it extremely easy to write webpages or scientific documents that include conde. This syntax was extended to allow users to embed R code directly into more complex documents. Perhaps the easiest way to understand the syntax is to look at an at the RMarkdown website.

The R code in a R Markdown document (`.rmd` file extension) can be nicely separated from regular text using the three backticks (3 times ‘, see below) and an instruction that it is R code that needs to be evaluated. A code chunk will look like:

```
    for (i in 1:5) {print(i)}
```

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

In this module we will be using `.qmd` a more flexible development of `.rmd` files.

Markdown files present several advantages compared to writing your code in the console or just using scripts. You'll save yourself a huge amount of work by embracing Markdown files from the beginning; you will keep track of your code and your steps, be able to document and present how you did your analysis (helpful when writing the methods section of a paper), and it will make it easier to re-run an analysis after a change in the data (such as additional data values, transformed data, or removal of outliers) or once you spot an error. Finally, it makes the script more readable.

1.2.6 R Packages

One of the greatest strengths about R is that so many people have developed add-on packages to do some additional function. To download and install the package from the Comprehensive R Archive Network (CRAN), you just need to ask RStudio it to install it via the menu `Tools -> Install Packages...`. Once there, you just need to give the name of the package and RStudio will download and install the package on your computer.

Once a package is downloaded and installed on your computer, it is available, but it is not loaded into your current R session by default. To improve overall performance only a few packages are loaded by default and the you must explicitly load packages whenever you want to use them. You only need to load them once per session/script.

```
library(dplyr)   # load the dplyr library, will be useful later
```

Stop here for 5 min and wait for instructions

1.3 Practice: Dataset and Dataframes

First of all, create a new Markdown document. We use the `File -> New File -> Quarto Document..` dropdown option, and a menu will appear asking you for the document title, author, and preferred output type. You can select HTML, but you will need your assignment to be submitted in PDF; more on that later.

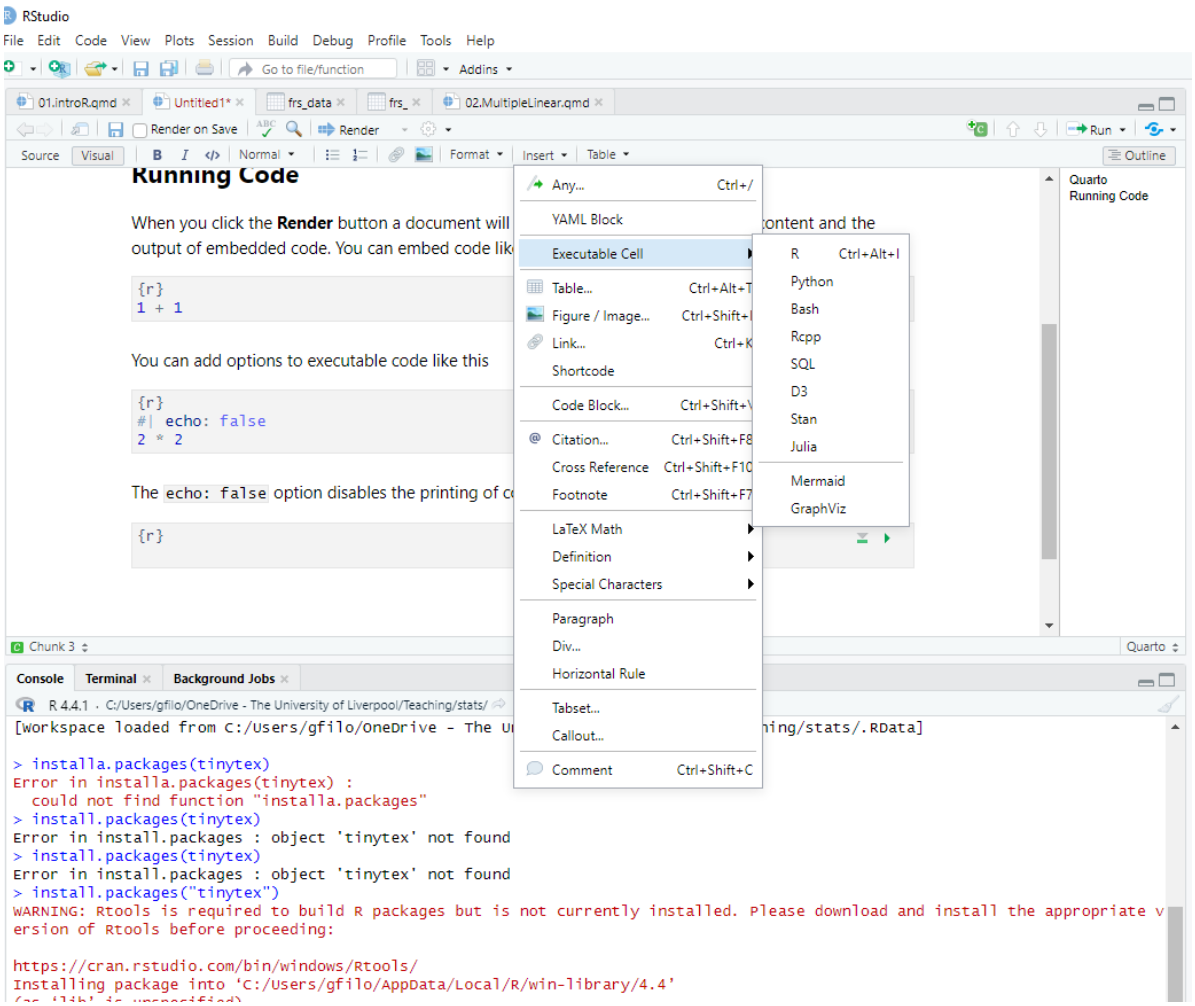
Follow the practical below. You can describe what you are doing in normal text. See here for how to format normal text in Markdown documents

Remember, when you want to write code in a markdown document you have to enclose it like this:

```{r}

```

or you can insert it manually:



Within this module we will be working with data stored in so-called datasets. A dataset is a structured collection of data points that represent various measurements or observations, often organized in a tabular format with rows and columns. A dataset might contain information about different locations, such as neighborhoods or cities, with each row representing a place and each column detailing characteristics like population density, average income, or number of green parks. For example, a dataset could be compiled to study patterns in urban mobility, where the data includes the number of daily commuters, the distance they travel, and the mode of transport they use. Datasets provide the essential building blocks for statistical analysis; they enable exploring relationships, identifying patterns, and drawing conclusions about certatin phenomena.

Examples of everyday datasets:

- **Premier League Standings**: Each row represents a team, with columns for points, games played, wins, draws, and losses.

- **Movie Dataset**: Each row represents a movie, with columns showing its title, genre, release year, director, and rating.
- **Weather Dataset**: Each row shows a day's weather in a city, with columns for temperature, humidity, wind speed, and precipitation.

Usually, data is organized in

- **Columns** of data representing some trait or variable that we might be interested in. In general, we might wish to investigate the relationship between variables.
- **Rows** represent a single object on which the column traits are measured.

For example, in a grade book for recording students scores throughout the semester, their is one row for every student and columns for each assignment. A greenhouse experiment dataset will have a row for every plant and columns for treatment type and biomass.

1.3.1 Datasets in R

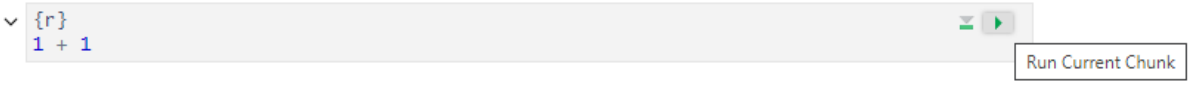
In R, we want a way of storing data where it feels just as if we had an Excel Spreadsheet where each row represents an observation and each column represents some information about that observation. We will call this object a `data.frame`, an R represention of a data set. The easiest way to understand data frames is to create one.

Task: Copy the code below in your markdown. Create a `data.frame` that represents an instructor's grade book, where each row is a student, and each column represents some sort of assessment.

```
Grades <- data.frame(
  Name  = c('Bob','Jeff','Mary','Valerie'),
  Exam.1 = c(90, 75, 92, 85),
  Exam.2 = c(87, 71, 95, 81)
)
# Show the data.frame
# View(Grades)  # show the data in an Excel-like tab.  Doesn't work when knitting
Grades          # show the output in the console. This works when knitting
```

```
     Name Exam.1 Exam.2
1     Bob     90     87
2    Jeff     75     71
3    Mary     92     95
4 Valerie     85     81
```

To execute just one chunk of code press the green arrow top-right of the chunk:



R allows two differnt was to access elements of the `data.frame`. First is a matrix-like notation for accessing particular values.

Format	Result
`[a,b]`	Element in row `a` and column `b`
`[a,]`	All of row `a`
`[,b]`	All of column `b`

Because the columns have meaning and we have given them column names, it is desirable to want to access an element by the name of the column as opposed to the column number.

Task: Copy and Run:

```
Grades[, 2]       # print out all of column 2
```

```
[1] 90 75 92 85
```

```
Grades$Name       # The $-sign means to reference a column by its label
```

```
[1] "Bob"     "Jeff"    "Mary"    "Valerie"
```

1.3.2 Importing Data in R

From: https://raw.githubusercontent.com/dereksonderegger/570L/master/07_DataImport.Rmd

Usually we won't type the data in by hand, but rather load the data from some package. Reading data from external sources is a necessary skill.

***Comma Separated Values* Data**

To consider how data might be stored, we first consider the simplest file format: the comma separated values file (`.csv`). In this file time, each of the "cells" of data are separated by a comma. For example, the data file storing scores for three students might be as follows:

```
Able, Dave, 98, 92, 94
Bowles, Jason, 85, 89, 91
Carr, Jasmine, 81, 96, 97
```

Typically when you open up such a file on a computer with MS Excel installed, Excel will open up the file assuming it is a spreadsheet and put each element in its own cell. However, you can also open the file using a more primitive program (say Notepad in Windows, TextEdit on a Mac) you'll see the raw form of the data.

Having just the raw data without any sort of column header is problematic (which of the three exams was the final??). Ideally we would have column headers that store the name of the column.

```
LastName, FirstName, Exam1, Exam2, FinalExam
Able, Dave, 98, 92, 94
Bowles, Jason, 85, 89, 91
Carr, Jasmine, 81, 96, 97
```

Reading (`.csv`) files

To make R read in the data arranged in this format, we need to tell R three things:

1. Where does the data live? Often this will be the name of a file on your computer, but the file could just as easily live on the internet (provided your computer has internet access).
2. Is the first row data or is it the column names?
3. What character separates the data? Some programs store data using tabs to distinguish between elements, some others use white space. R's mechanism for reading in data is flexible enough to allow you to specify what the separator is.

The primary function that we'll use to read data from a file and into R is the function `read.csv()`. This function has many optional arguments but the most commonly used ones are outlined in the table below.

Argument	Default	Description
`file`	Required	A character string denoting the file location.
`header`	`TRUE`	Specifies whether the first line contains column headers.
`sep`	`","`	Specifies the character that separates columns. For `read.csv()`, this is usually a comma.

Argument	Default	Description
`skip`	`0`	The number of lines to skip before reading data; useful for files with descriptive text before the actual data.
`na.strings`	`"NA"`	Values that represent missing data; multiple values can be specified, e.g., `c("NA", "-9999")`.
`quote`	`"`	Specifies the character used to quote character strings, typically `"` or `'`.
`stringsAsFactors`	`FALSE`	Controls whether character strings are converted to factors; `FALSE` means they remain as character data.
`row.names`	`NULL`	Allows specifying a column as row names, or assigning `NULL` to use default indexing for rows.
`colClasses`	`NULL`	Specifies the data type for each column to speed up reading for large files, e.g., `c("character", "numeric")`.
`encoding`	`"unknown"`	Sets the text encoding of the file, which can be useful for files with special or international characters.

Most of the time you just need to specify the file. |

Task: Let's read in a dataset of terrorist attacks that have taken place in the UK:

```
attacks <- read.csv(file   = '../data/attacksUK.csv')  # where the data lives
View(attacks)
```

1.4 Practice: Descriptive Statistics

1.4.1 Summarizing Data

It is very important to be able to take a data set and produce summary statistics such as the mean and standard deviation of a column. For this sort of manipulation, we use the package `dplyr`. This package allows chaining together many common actions to form a particular task.

The fundamental operations to perform on a data set are:

- **Subsetting** - Returns a dataframe with only particular columns or rows
  - `select` - Selecting a subset of columns by name or column number.
  - `filter` - Selecting a subset of rows from a data frame based on logical expressions.
  - `slice` - Selecting a subset of rows by row number.
- `arrange` - Re-ordering the rows of a data frame.
- `mutate` - Add a new column that is some function of other columns.
- `summarise` - calculate some summary statistic of a column of data. This collapses a set of rows into a single row.

Each of these operations is a function in the package `dplyr`. These functions all have a similar calling syntax:

- The first argument is a data set.
- Subsequent arguments describe what to do with the input data frame and you can refer to the columns without using the `df$column` notation.

All of these functions will return a data set.

Let's consider the `summarize` function to calculate the mean score for `Exam.1`. Notice that this takes a data frame of four rows, and summarizes it down to just one row that represents the summarized data for all four students.

```
library(dplyr) # load the library
Grades %>%
  summarize( Exam.1.mean = mean( Exam.1 ) )
```

```
  Exam.1.mean
1        85.5
```

Similarly you could calculate the **standard deviation** for the exam as well.

```
Grades %>%
  summarize( Exam.1.mean = mean( Exam.1 ),
             Exam.1.sd   = sd(   Exam.1   ) )
```

```
  Exam.1.mean Exam.1.sd
1        85.5  7.593857
```

Task: Write the code above in your markdown file and run it. Do not to copy it this time.

The `%>%` operator works by translating the command `a %>% f(b)` to the expression `f(a,b)`. This operator works on any function `f`. This is useful when we want to start with `x`, and first apply a function `f()`, then `g()`, and then `h()`; the usual R command would be `h(g(f(x)))` which is hard to read. Using the pipe command `%>%`, this sequence of operations becomes `x %>% f() %>% g() %>% h()`.

Below, the code takes the Grades dataframe and calculates a column for the average exam score, and then sorts the data according to the that average score

```
Grades %>%   mutate( Avg.Score = (Exam.1 + Exam.2) / 2 ) %>%   arrange( Avg.Score )
```

```
     Name Exam.1 Exam.2 Avg.Score
1    Jeff     75     71      73.0
2 Valerie     85     81      83.0
3     Bob     90     87      88.5
4    Mary     92     95      93.5
```

You don't have to memorise this.

Let's go back to the terrorist attacks dataframe. There are attacks perpetrated by several different groups. Each record is a single attack and contains information about who perpetrated the attack, what year, how many were killed and how many were wounded. You can get a glimpse of the dataframe with the function head

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

```
   nrKilled nrWound year        country                         group
1         0       0 2005 United Kingdom   Abu Hafs al-Masri Brigades
2         0       0 2005 United Kingdom   Abu Hafs al-Masri Brigades
3         0       0 2005 United Kingdom   Abu Hafs al-Masri Brigades
4         0       0 2005 United Kingdom   Abu Hafs al-Masri Brigades
```

```
5         0       1 1982 United Kingdom Abu Nidal Organization (ANO)
6         0       0 2014 United Kingdom                   Anarchists
7         0       0 2014 United Kingdom                   Anarchists
8         0       0 2014 United Kingdom                   Anarchists
9         0       0 2014 United Kingdom                   Anarchists
10        0       0 2014 United Kingdom                   Anarchists
                          attack                     target
1               Bombing/Explosion             Transportation
2               Bombing/Explosion             Transportation
3               Bombing/Explosion             Transportation
4               Bombing/Explosion             Transportation
5                   Assassination     Government (Diplomatic)
6  Facility/Infrastructure Attack                   Business
7  Facility/Infrastructure Attack                   Business
8  Facility/Infrastructure Attack                   Business
9  Facility/Infrastructure Attack Private Citizens & Property
10 Facility/Infrastructure Attack                     Police
                      weapon
1  Explosives/Bombs/Dynamite
2  Explosives/Bombs/Dynamite
3  Explosives/Bombs/Dynamite
4  Explosives/Bombs/Dynamite
5                  Firearms
6                Incendiary
7                Incendiary
8                Incendiary
9                Incendiary
10               Incendiary
```

We might want to compare different actors and see the mean and standard deviation of the number of people wound, by each group's attack, across time. To do this, we are still going to use the `summarize`, but we will precede that with `group_by(group)` to tell the subsequent `dplyr` functions to perform the actions separately for each breed.

```
attacks %>%
  group_by( group) %>%
  summarise( Mean = mean(attacks$nrWound),
             Std.Dev = sd(attacks$nrWound))
```

```
# A tibble: 38 x 3
   group                                                Mean Std.Dev
   <chr>                                               <dbl>   <dbl>
```

```
 1 Abu Hafs al-Masri Brigades                          0.963    7.22
 2 Abu Nidal Organization (ANO)                        0.963    7.22
 3 Anarchists                                          0.963    7.22
 4 Animal Liberation Front (ALF)                       0.963    7.22
 5 Animal Rights Activists                             0.963    7.22
 6 Armenian Secret Army for the Liberation of Armenia 0.963    7.22
 7 Black September                                     0.963    7.22
 8 Continuity Irish Republican Army (CIRA)             0.963    7.22
 9 Dissident Republicans                               0.963    7.22
10 Informal Anarchist Federation                       0.963    7.22
# i 28 more rows
```

Task: Write the code above in your markdown file and run it. Try out another categorical variable instead of `group` (e.g. `year`) and `nrKilled` instead of `nrWound`.

Let's now move to another dataset to address a research question.

For illustration purposes, we will use the **Family Resources Survey (FRS)**. The FRS is an annual survey conducted by the UK government that collects detailed information about the income, living conditions, and resources of private households across the United Kingdom. Managed by the Department for Work and Pensions (DWP), the FRS provides data that is essential for understanding the economic and social conditions of households and informing public policy.

Consider questions such as:

- How many respondents (persons) are there in the 2016-17 FRS?
- How many variables (population attributes) are there?
- What types of variables are present in the FRS?
- What is the most detailed geography available in the FRS?

Task: To answer these questions, download from Canvas, save in the data folder, load and inspect the dataset.

```r
# the FRS dataset should be already loaded, otherwise
frs_data <- read.csv("../data/FRS/FRS16-17_labels.csv")

# Display basic structure
glimpse(frs_data)
```

```
Rows: 44,145
Columns: 45
$ household        <int> 6087, 6101, 6103, 6122, 6134, 6136, 6138, 6140, 6143,~
$ family           <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
```

```
$ person           <int> 5, 3, 3, 3, 2, 4, 4, 3, 3, 4, 4, 3, 4, 2, 5, 3, 4, 3,~
$ country          <chr> "England", "England", "England", "Northern Ireland", ~
$ region           <chr> "London", "South East", "Yorks and the Humber", "Nort~
$ age_group        <chr> "05-10", "05-10", "05-10", "05-10", "05-10", "05-10",~
$ sex              <chr> "Female", "Male", "Male", "Female", "Female", "Female~
$ marital_status   <chr> "Single", "Single", "Single", "Single", "Single", "Si~
$ ethnicity        <chr> "Mixed / multiple ethnic groups", "White", "White", "~
$ hrp              <chr> "Not HRP", "Not HRP", "Not HRP", "Not HRP", "Not HRP"~
$ rel_to_hrp       <chr> "Son/daughter (incl. adopted)", "Son/daughter (incl. ~
$ lifestage        <chr> "Child (0-17)", "Child (0-17)", "Child (0-17)", "Chil~
$ dependent        <chr> "Dependent", "Dependent", "Dependent", "Dependent", "~
$ arrival_year     <chr> "UK Born", "UK Born", "UK Born", "UK Born", "UK Born"~
$ birth_country    <chr> "Dependent child", "Dependent child", "Dependent chil~
$ care_hours       <chr> "0 hours per week", "0 hours per week", "0 hours per ~
$ educ_age         <chr> "Dependent child", "Dependent child", "Dependent chil~
$ educ_type        <chr> "School (full-time)", "School (full-time)", "School (~
$ fam_youngest     <chr> "7", "4", "0", "7", "0", "9", "10", "0", "3", "10", "~
$ fam_toddlers     <int> 0, 1, 1, 0, 2, 0, 0, 2, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1,~
$ fam_size         <int> 4, 4, 4, 3, 4, 4, 3, 5, 4, 4, 3, 4, 4, 3, 5, 4, 4, 4,~
$ happy            <chr> "Dependent child", "Dependent child", "Dependent chil~
$ health           <chr> "Not known", "Not known", "Not known", "Not known", "~
$ hh_accom_type    <chr> "Terraced house/bungalow", "Detached house/bungalow",~
$ hh_benefits      <int> 10868, 0, 1768, 8632, 8372, 1768, 1768, 1768, 0, 0, 1~
$ hh_composition   <chr> "Three or more adults, 1+ children", "One adult femal~
$ hh_ctax_band     <chr> "Band D", "Band F", "Band A", "Band B", "Band A", "Ba~
$ hh_housing_costs <chr> "4316", "10296", "5408", "Northern Ireland", "5720", ~
$ hh_income_gross  <int> 54236, 180804, 26936, 19968, 17992, 76596, 31564, 366~
$ hh_income_net    <int> 44668, 120640, 23556, 19968, 17992, 62868, 29744, 287~
$ hh_size          <int> 5, 4, 4, 3, 4, 4, 4, 5, 4, 4, 4, 4, 4, 5, 5, 4, 4, 4,~
$ hh_tenure        <chr> "Mortgaged (including part rent / part own)", "Mortga~
$ highest_qual     <chr> "Dependent child", "Dependent child", "Dependent chil~
$ income_gross     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ income_net       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ jobs             <chr> "Dependent child", "Dependent child", "Dependent chil~
$ life_satisf      <chr> "Dependent child", "Dependent child", "Dependent chil~
$ nssec            <chr> "Dependent child", "Dependent child", "Dependent chil~
$ sic_chapter      <chr> "Dependent child", "Dependent child", "Dependent chil~
$ sic_division     <chr> "Dependent child", "Dependent child", "Dependent chil~
$ soc2010          <chr> "Dependent child", "Dependent child", "Dependent chil~
$ work_hours       <chr> "Dependent child", "Dependent child", "Dependent chil~
$ workstatus       <chr> "Dependent Child", "Dependent Child", "Dependent Chil~
$ years_ft_work    <chr> "Dependent child", "Dependent child", "Dependent chil~
$ survey_weight    <int> 2315, 1317, 2449, 427, 1017, 1753, 1363, 1344, 828, 1~
```

and summary:

```
summary(frs_data)
```

```
   household          family           person       country
 Min.   :    1   Min.   :1.000   Min.   :1.00   Length:44145
 1st Qu.: 4816   1st Qu.:1.000   1st Qu.:1.00   Class :character
 Median : 9673   Median :1.000   Median :2.00   Mode  :character
 Mean   : 9677   Mean   :1.106   Mean   :1.98
 3rd Qu.:14553   3rd Qu.:1.000   3rd Qu.:3.00
 Max.   :19380   Max.   :6.000   Max.   :9.00
    region           age_group             sex            marital_status
 Length:44145       Length:44145       Length:44145       Length:44145
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character


  ethnicity            hrp             rel_to_hrp         lifestage
 Length:44145       Length:44145       Length:44145       Length:44145
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character


  dependent          arrival_year       birth_country       care_hours
 Length:44145       Length:44145       Length:44145       Length:44145
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character


   educ_age           educ_type          fam_youngest        fam_toddlers
 Length:44145       Length:44145       Length:44145       Min.   :0.0000
 Class :character   Class :character   Class :character   1st Qu.:0.0000
 Mode  :character   Mode  :character   Mode  :character   Median :0.0000
                                                          Mean   :0.2557
                                                          3rd Qu.:0.0000
                                                          Max.   :4.0000
    fam_size         happy              health          hh_accom_type
 Min.   :1.000   Length:44145       Length:44145       Length:44145
 1st Qu.:2.000   Class :character   Class :character   Class :character
```

```
Median :2.000   Mode  :character   Mode  :character   Mode  :character
Mean   :2.599
3rd Qu.:4.000
Max.   :9.000
 hh_benefits    hh_composition     hh_ctax_band       hh_housing_costs
Min.   :    0   Length:44145       Length:44145       Length:44145
1st Qu.:    0   Class :character   Class :character   Class :character
Median : 1768   Mode  :character   Mode  :character   Mode  :character
Mean   : 5670
3rd Qu.:10192
Max.   :54080
hh_income_gross   hh_income_net        hh_size      hh_tenure
Min.   :-326092   Min.   :-334776   Min.   :1.00   Length:44145
1st Qu.:  22256   1st Qu.:  20748   1st Qu.:2.00   Class :character
Median :  35984   Median :  31512   Median :3.00   Mode  :character
Mean   :  46076   Mean   :  37447   Mean   :2.96
3rd Qu.:  57252   3rd Qu.:  47008   3rd Qu.:4.00
Max.   :1165216   Max.   :1116596   Max.   :9.00
highest_qual        income_gross       income_net          jobs
Length:44145       Min.   :-354848   Min.   :-358592   Length:44145
Class :character   1st Qu.:     52   1st Qu.:      0   Class :character
Mode  :character   Median :  12740   Median :  12012   Mode  :character
                   Mean   :  17305   Mean   :  14204
                   3rd Qu.:  23712   3rd Qu.:  20384
                   Max.   :1127360   Max.   :1110928
life_satisf           nssec           sic_chapter        sic_division
Length:44145       Length:44145       Length:44145       Length:44145
Class :character   Class :character   Class :character   Class :character
Mode  :character   Mode  :character   Mode  :character   Mode  :character



  soc2010           work_hours         workstatus        years_ft_work
Length:44145       Length:44145       Length:44145       Length:44145
Class :character   Class :character   Class :character   Class :character
Mode  :character   Mode  :character   Mode  :character   Mode  :character



survey_weight
Min.   :  221
1st Qu.: 1097
Median : 1380
```

```
Mean   : 1459
3rd Qu.: 1742
Max.   :39675
```

1.4.2 Understanding the Structure of the FRS Datafile

In the FRS data structure, each row represents a person, but:

- Each person is nested within a family.
- Each family is nested within a household.

Below is an example dataset structure:

household	family	person	region	age_group	sex	marital_status	rel_to_hrp
1	1	1	London	40-44	Female	Married/Civil partnership	Spouse
1	1	2	London	40-44	Male	Married/Civil partnership	Household Representative
1	1	3	London	5-10	Male	Single	Son/daughter (incl. adopted)
1	1	4	London	5-10	Female	Single	Son/daughter (incl. adopted)
1	1	5	London	16-19	Male	Single	Step-son/daughter
2	1	1	Scotland	35-39	Male	Single	Household Representative
3	1	1	Yorks and the Humber	35-39	Female	Married/Civil partnership	Household Representative
3	1	2	Yorks and the Humber	35-39	Male	Married/Civil partnership	Spouse
3	1	3	Yorks and the Humber	5-10	Male	Single	Step-son/daughter
4	1	1	Wales	0-4	Male	Single	Son/daughter (incl. adopted)

household	family	person	region	age_group	sex	marital_status	rel_to_hrp
4	1	2	Wales	60-64	Male	Married/Civil partnership	Household Representative
4	1	3	Wales	55-59	Female	Married/Civil partnership	Spouse
4	2	3	Wales	30-34	Female	Single	Son/daughter (incl. adopted)

The first five people in the FRS all belong to the same household (household 1); they also all belong to the same family. This family comprises a married middle-aged couple plus their three children, one of whom is a stepson.

The second household (household 2) comprises only one person – a single middle-aged male.The third household comprises another married couple, this time with two children.

Superficially the fourth household looks similar to households 1 and 2: a married couple plus their daughter. The difference is that this particular married couple is nearing retirement age, and their daughter is middle-aged. Consequently, despite being a child of the married couple, the middle-aged daughter is treated as a separate 'family' (family 2 in the household). This is because the FRS (and Census) define a 'family' as a couple plus any 'dependent' children. A dependent child is defined as a child who is either' aged 0-15 or aged 16-19, unmarried and in full-time education. All children aged 16-19 who are married or no longer in full-time education are regarded as 'independent' adults who form their own family unit, as are all children aged 20+.

The inclusion of all persons in a household allows us more flexibility in the types of research question we can answer. For example, we could explore how the likelihood of a woman being in paid employment `WorkStatus` is influenced by the age of the youngest child still living in her family (if any) `fam_youngest`.

In the FRS (and Census), a "family" is defined as a couple and any "dependent" children. Dependent children are defined as those aged 0–15, or aged 16–19 if unmarried and in full-time education.

1.4.3 Explore the Distribution of Your Outcome Variable

Before starting your analysis, it is critical to know the type of scale used to measure your outcome variable: is it categorical or continuous? Here we will start off by exploring a continuous variable which can then turn into a categorical variable (e.g. top earners: yes or no). We explore the income distribution in the UK by first looking at the low and high end of the distribution ie. What sorts of people have high (or low) incomes?

In the FRS each person's annual income is recorded, both gross (pre-tax) and net (post-tax). This income includes all income sources, including earnings, profits, investment returns, state benefits, occupational pensions etc. As it is possible to make a loss on some of these activities, it is also possible (although unusual) for someone's gross or net annual income in a given year to be negative (representing an overall loss).

Task: Load the FRS dataset into your R environment, if it's not already loaded, and inspect the data.

Open the dataset in RStudio's **Data Viewer** to explore its structure, including the `income_gross` and `income_net` variables.

```
 # Open the data in the RStudio Viewer
View(frs_data)
```

in the **Data Viewer** tab, scroll horizontally to locate the `income_gross` and `income_net` columns. If columns are listed alphabetically, they will appear near other attributes that start with "income."

You should notice two things:

- Incomes are recorded to the nearest £, NOT in income bands.
- Dependent children almost all have a recorded income of £0.

This second observation highlights the somewhat loose wording of our question above (*What sorts of people have high (or low) incomes?*). To avoid reaching the somewhat banal conclusion that those with the lowest of all incomes are almost all children, we should re-frame the question more precisely as *What sorts of people (excluding dependent children) have low incomes?*

Task: Determine the Scale of the Outcome Variable.

```
# Summarize income variables
summary(frs_data$income_gross)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
-354848      52   12740   17305   23712 1127360 
```

```
# Summarize income variables
summary(frs_data$income_net)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
-358592       0   12012   14204   20384 1110928 
```

Task: Exclude Dependent Children.

You need to select all cases (persons) that are independent, that is where the variable `dependent` has values different from `!=` "Dependent" or equal `==` "Independent".

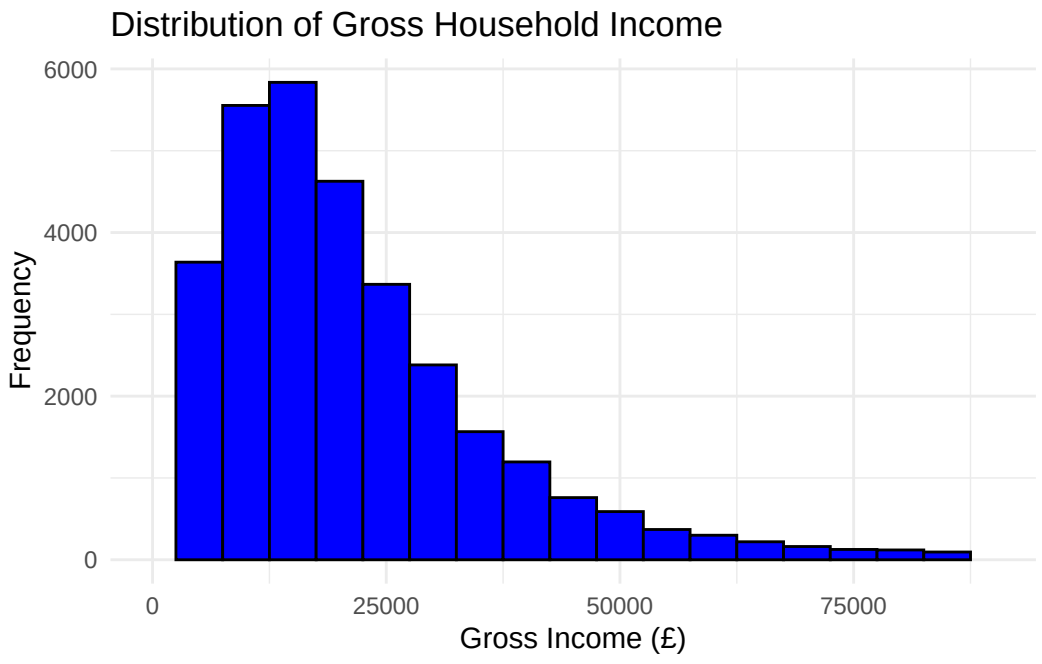
```
# Filter to include only independent persons
frs_independent <- frs_data %>% filter(dependent != "Dependent")
```

Task: Create a basic histogram (a visualisation lecture is scheduled later on).

The income variables in the FRS are all scale variables so a good starting point is to examine its distribution looking at a histogram of `income_gross`.

```
library(ggplot2)

 ggplot(frs_independent, aes(x = income_gross)) +
   geom_histogram(binwidth = 5000, fill = "blue", color = "black") +
   labs(
     title = "Distribution of Gross Household Income",
     x = "Gross Income (£)",
     y = "Frequency"
   ) +
   xlim(0, 90000) +
   theme_minimal()
```



You should see the histogram below. It reveals that the income distribution is very skewed with few people earning high salaries and the majority earning just over or less 35,000 annually.

Task: Adopt a regrouping strategy.

You can also cross-tabulate gross (or net) income with any of the other variables in the FRS to your heart's content – or can you?

Again, here is important to recall that the income variables in the FRS are all 'scale' variables; in other words, they are precise measures rather than broad categories. Consequently, every single person in the FRS potentially has their own unique income value. That could make for a table c. 44,000 rows long (one row per person) if each person has their own unique value. The solution is to create a categorical version of the original income variable by assigning each person to one of a set of income categories (income bands). Having done this, cross-tabulation then becomes possible.

But which strategy to use? Equal intervals, percentiles or 'ad hoc'. Here I would suggest that 'ad hoc' is best: all you want to do is to allocate each independent adult to one of three arbitrarily defined groups: 'low', 'middle' and 'high' income. **Define `Low` and `High` Income Thresholds**

Define thresholds for income categories:

- Low-income threshold: £_____
- High-income threshold: £_____

Task: Create a New Variable Based on Regrouping of Original Variable.

Recode `income_gross` into categories based on the chosen thresholds.

```
# Define thresholds for income categories
LOW_THRESHOLD <- 10000 # Replace with the upper limit for low income
HIGH_THRESHOLD <- 50000 # Replace with the lower limit for high income

# Define income categories based on thresholds
frs_independent <- frs_independent %>%
    mutate(income_category = case_when(
        income_gross <= LOW_THRESHOLD ~ "Low",
        income_gross >= HIGH_THRESHOLD ~ "High",
        TRUE ~ "Middle" ))
```

The `mutate()` function in R, from the **dplyr** package, is used to add or modify columns in a data frame. It allows you to create new variables or transform existing ones by applying calculations or conditional statements directly within the function.

Explanation of the code

- **`frs_independent %>%`**: The pipe operator `%>%` sends `frs_independent` into `mutate()`, allowing us to apply transformations without reassigning it repeatedly.
- **`mutate()`**: Starts the transformation process by defining new or modified columns.
- **`income_category = case_when(...)`**:
  - This creates a new column named `income_category`.
  - The `case_when()` function defines conditions for assigning values to this new column.
- **`case_when()`**:
  - `case_when()` is used here to assign categorical labels based on conditions.
  - `income_gross <= LOW_THRESHOLD ~ "Low"`: If `income_gross` is less than or equal to `LOW_THRESHOLD`, `income_category` will be labeled "Low."
  - `income_gross >= HIGH_THRESHOLD ~ "High"`: If `income_gross` is greater than or equal to `HIGH_THRESHOLD`, `income_category` will be labeled "High."
  - `TRUE ~ "Middle"`: Any values not meeting the previous conditions are labeled "Middle."

Task: Add some Metadata.

Define metadata for the new variable by labeling income categories.

```
# Add metadata by converting to a factor and defining labels

frs_independent$income_category <- factor(frs_independent$income_category,
                        levels = c("Low", "Middle", "High"), labels = c("<= £10,000", "£10,0
```

Task: Check your work.

Examine the frequency distribution of the variable you have just created. Both variables should have the same number of missing cases, unless:

- Missing cases in the old variable have been intentionally converted into valid cases in the new variable.
- You forgot to allocate a new value to one of the old variable categories, in which case the new variable will have more missing cases than the old variable.

```
# Frequency distribution of income categories
table(frs_independent$income_category)
```

```
<= £10,000 £10,001 - £49,999        >= £50,000
      8584             22981              2271
```

After preparing the data, use cross-tabulations to compare income levels across demographic groups.

```
# Cross-tabulate income category by age group, nationality, etc.
table(frs_independent$income_category, frs_independent$age_group)
```

```
                   16-19 20-24 25-29 30-34 35-39 40-44 45-49 50-54 55-59 60-64
  <= £10,000         373   680   492   558   474   511   554   652   781   826
  £10,001 - £49,999  263  1241  1802  2056  2052  1948  1995  1967  1749  1772
  >= £50,000           1     8    59   186   314   331   334   356   237   177

                   65-69 70-74  75+
  <= £10,000         773   744 1166
  £10,001 - £49,999 2073  1554 2509
  >= £50,000         144    56   68
```

Explore income distribution across different regions.

```
# Cross-tabulate income category by region
table(frs_independent$income_category, frs_independent$region)
```

```
                   East Midlands East of England London North East North West
  <= £10,000                 562             665    740        357        878
  £10,001 - £49,999         1550            1855   1850        979       2347
  >= £50,000                 135             245    367         48        174

                   Northern Ireland Scotland South East South West Wales
  <= £10,000                    874     1212        895        588   399
  £10,001 - £49,999            2305     3234       2563       1707   971
  >= £50,000                    123      322        367        149    63

                   West Midlands Yorks and the Humber
  <= £10,000                 744                  670
  £10,001 - £49,999         1892                 1728
  >= £50,000                 164                  114
```

Tips for Cross-Tabulation

- Place the income variable in the columns.
- Add multiple variables in the rows to create simultaneous cross-tabulations.