ENVS225 Assignment 1

Gabriele Filomena and Zi Ye

2025-10-28

Table of contents

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.

- 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, supported by tables or/and charts.
    - Correct interpretation of correlation coefficients, statistical significance, and model fit.
    - 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

Normally you don't need to do so, but if something does not work, you may have to install Quarto. Recent versions of RStudio do not require you to do so. installed.

.html file

You can submit a `.html` 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).

To do so, at the top of your document, include something like this in a YAML cell (see Template, in RStudio, "visual view", `Insert`, `YAML block`)

```
title: "Assignment 1"
author: "Anonymous" # do not change
format: html
```

Once your document is ready with your text and code, render the Quarto document: Click the **Render** button in the RStudio toolbar (blue arrow). This, if everything works (code, existing data, etc), will create an `.html` file in the same location as your `.qmd` file

.pdf file: More complicated

If you really want to render your `.qmd` file into a `.pdf` file, you need to follow a more complicated path.

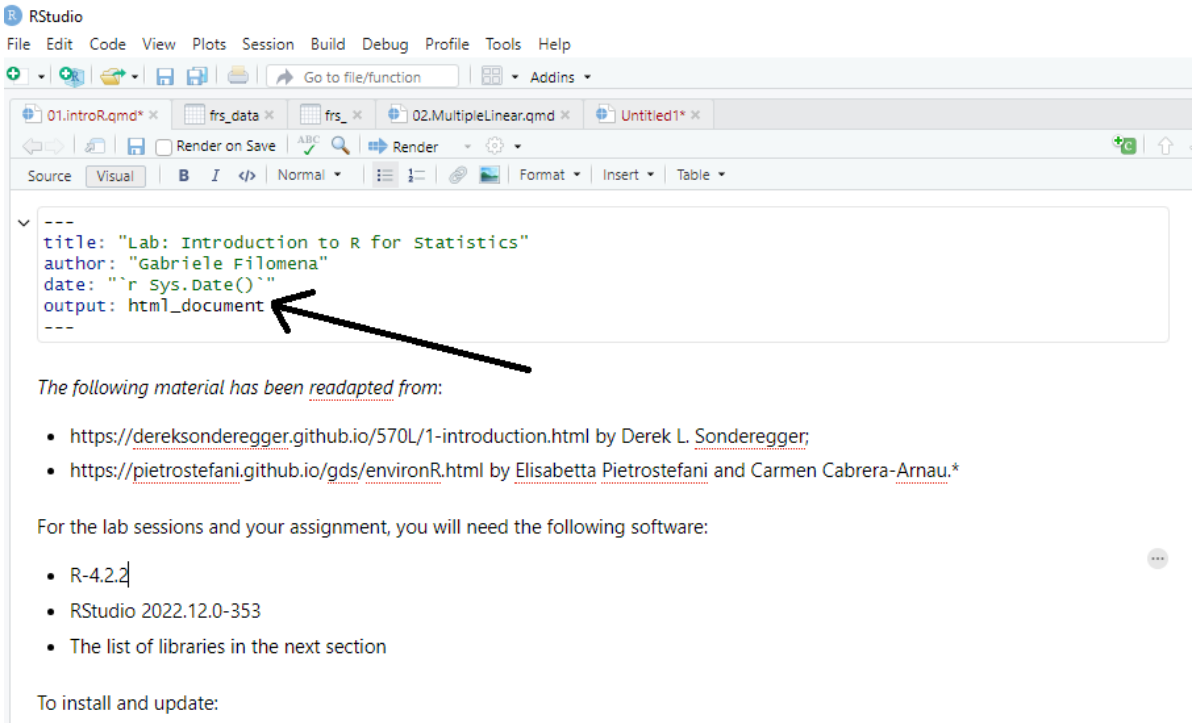
LaTeX Installation: you'll need a LaTeX distribution like **TinyTeX** from R.

1. Execute this in the RStudio **console** (bottom):

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

2. **Open the Quarto File**: Open your `.qmd` file in RStudio.

3. **Set Output Format**: In the YAML header at the top of your Quarto file, specify `pdf` under `format`:




```
title: "Assignment 1"
author: "Anonymous" # do not change
format: pdf
```

4. Click the **Render** button in the RStudio toolbar (blue arrow).

Assessment Template

Suggested Report Structure — This template follows the requested sections. Each section contains guidance text and an empty R code cell for your analysis. Download it from here (you need to be logged in with the UoL account) and work on it with your own Rstudio.

Setup

Load commonly used libraries for statistical analysis and data manipulation.

```
# Core tidyverse
library(tidyverse)    # includes ggplot2, dplyr, tidyr, readr, purrr, tibble, stringr, forcat

# Tables & reporting
library(kableExtra)

# Stats helpers
library(broom)

## These need to be installed on your PC.
```

Once all the chunks execute correctly (no errors when running all the code, `Ctrl +Alt + R`) you can render the `qmd` file. Follow the instructions here to render the document to a PDF. Remove this message and any irrelevant text from the final submission.

Introduction

Aim. State clearly what you want to investigate (e.g., association between X and Y).
Relevance. Explain why the topic matters (theory, policy, practice).
Gap & Research Question (RQ). Identify what is missing in the current knowledge and

pose 1 RQ (avoid yes/no questions; ask *how*, *to what extent*, *which factors*).
Structure. Briefly preview how the rest of the article is organized.

Literature Review

What we know. Summarise key findings from prior studies relevant to your RQ.
Predictor rationale. Justify the inclusion of variables (theory-driven, prior empirical evidence).
Remaining gap. Specify precisely the gap this report addresses (e.g., population, geography, time period, variable, method).
Note: The goal is a clean, sensible analysis grounded in existing ideas—not necessarily novel theory.

Methodology

Data. Briefly describe the dataset (who collected it, when, sample size, key variables).
Transformations. Describe any recoding/aggregation (e.g., bin income into bands; reduce age groups from 11 to 3). Provide justification.
Techniques. Outline and justify the statistical methods used (e.g., GLM/LM, logistic regression, mixed models, matching).

Important: do not include statistics here, or graphs, or tables.

Results and Discussion

```
# summary variables, distribution, plots, etc
```

Present the variables Start with descriptive statistics for your variables and distribution. If relevant, include discuss one-to-one associations (briefly, e.g. correlation plots). You can provide correlation plots and/or boxplots but do not overload the report with graphs.***

```
# run the model, show tables summarising the Regression model.
```

Results + interpretation. Present estimates and interpret them in plain language, tying back to the RQ. The Regression model should the core of this section!! **Link to literature.** Compare/contrast with prior findings. **Selective visuals.** Use clear charts/tables to highlight key results only (avoid clutter). ———————————————————————————

Conclusion

Summary. Recap the main findings vis-à-vis the RQs (do not introduce new results).
Limitations. Offer a brief, honest self-critique (data, measurement, design, external validity).
Implications. Indicate what the findings suggest for practice, policy, or future research.

———————————————————

Working Directories and Paths

Start clean (optional but handy)

```
# Clear the environment (objects in memory)
rm(list = ls())
```

Know where you are (working directory)

Your working directory (WD) is the folder R uses as the starting point for relative paths.

```
# Show current working directory
getwd()
```

```
[1] "C:/Users/gfilo/OneDrive - The University of Liverpool/Teaching/stats/general"
```

A relative path is specified from the working directory (e.g., `data/myfile.csv`).

An absolute path starts at the drive root (e.g., `C:/Users/you/Documents/stats/data/myfile.csv`).

Recommended folder layout (ENVS225)

Download the module materials, unzip them them wherever you prefer. Then un-nest the folder with the material (you will have one folder called `stats-main` containing another folder inside, with the same name. Aim for:

```
stats-main/
   data/
   labs_img/
   labs/
```

You can delete other sub-folders (e.g., docs) if you don't need them. It is strongly advised to move this folder into your M: drive on university machines, so it's accessible from every other PC on campus. Go to `This PC` and access `M:\(UoL userName)`

Set the working directory in RStudio (every time you use a new PC)

Windows: `RStudio > Tools > Global Options > General > Default working directory` > Browse to your `stats-main` folder.

macOS: `RStudio > Preferences > General`(older versions: under `Appearance/Pane Layout`) set Default working directory to `stats-main/`, this is the folder with all the materials inside.

Then restart RStudio and verify:

```
getwd()
```

```
[1] "C:/Users/gfilo/OneDrive - The University of Liverpool/Teaching/stats/general"
```

Tip: In addition always save your projects inside the folder `stats-main`. This keeps paths stable and prevents issues.

Loading data (CSV, Excel) with relative paths

Option a) Your `.qmd` project is in `stats-main`

```
library(readr)
library(readxl)
df <- read_csv("data/survey.csv")
xls <- read_excel("data/survey.xlsx", sheet = 1)
```

Option b) Your `.qmd` project is in `stats-main\subfolder`

```
df <- read_csv("../data/survey.csv")
xls <- read_excel("../data/survey.xlsx", sheet = 1)
```

In summary:

- `read_csv(MyFile.csv)` RStudio looks in the working directory for the file `MyFile.csv`.
- `read_csv(MyFolder/MyFile.csv)` RStudio looks inside the Working Directory (WD), for the folder `MyFolder`, then the file `MyFile.csv`.
- `read_csv(data/survey.csv)` RStudio looks inside the WD, looks inside `data`, then `survey.csv`.

- `read_csv(``../data/survey.csv` RStudio goes up one folder from the WD, then looks for the folder `data`and the file `surve.csv`.

You can always inspect where you are by

```
getwd()
```

```
[1] "C:/Users/gfilo/OneDrive - The University of Liverpool/Teaching/stats/general"
```

```
list.files()        # What's in the working directory?
```

```
 [1] "about.qmd"           "assessment.html"     "assessment.qmd"
 [4] "howSubmit.html"      "howSubmit.qmd"       "overview.html"
 [7] "overview.qmd"        "reportTemplate.html" "reportTemplate.qmd"
[10] "setup.qmd"           "wdPaths.html"        "wdPaths.qmd"
[13] "wdPaths.rmarkdown"   "wdPaths_files"
```

```
list.files("data")  # Do we see the data folder?
```

```
character(0)
```

1 Lab: Introduction to R

The following material has been readapted from:

- https://dereksonderegger.github.io/570L/1-introduction.html by Derek L. Sonderegger;
- 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

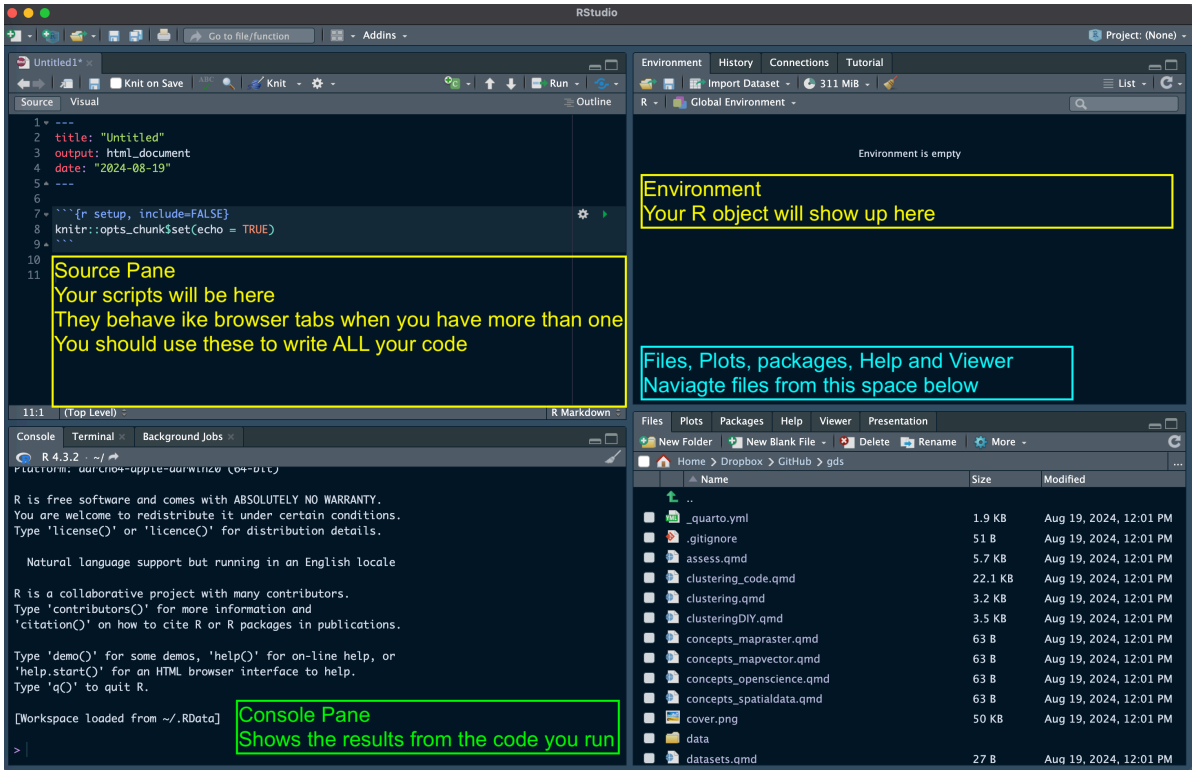
We will be running R through the program RStudio which is located at rstudio.com. When you first open up RStudio the console window gives you some information about the version of R you are running and then it gives the prompt `>`. This prompt is waiting for you to input a command. The prompt + tells 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.

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()
```

1.2.2 Using the console

We can use the console to perform 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.

```r
6*8
```

```
[1] 48
```

```r
4^3
```

```
[1] 64
```

```r
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.

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

```
[1] 3.141593
```

```r
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:

```r
# 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

Normally you would use the Console for quick calculations or executions. **In this module, though, we are going to work with Quarto Markdown Scripts (`.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 code. 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
```

** `.qmd - the type of scripts we use -` are just 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
```

This is just a quick intro to R, now move to the actual practical of week 1.

2 Lab: Exploring a Dataset

The following material has been readapted from:

- https://pietrostefani.github.io/gds/environR.html by Elisabetta Pietrostefani and Carmen Cabrera-Arnau
- https://raw.githubusercontent.com/dereksonderegger/570L/master/07_DataImport.Rmd

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) The Comprehensive R Archive Network (CRAN)
- RStudio 2022.12.0-353 (or higher) Posit

IMPORTANT: Before starting: verify that you have set the directory correctly and familiarise yourself with working with directories and loading files. These are the instructions.

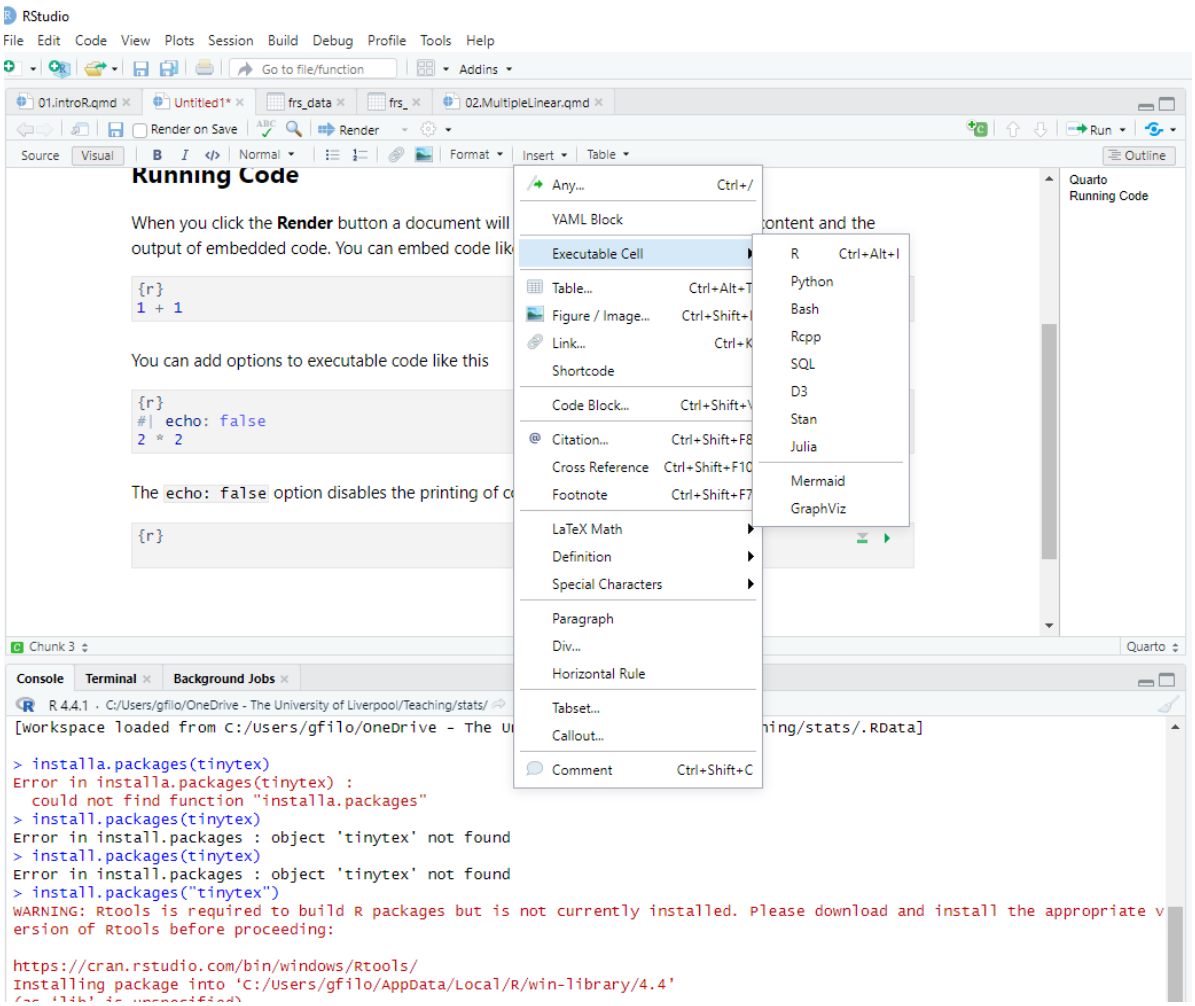
Now create a .qmd project. We use the `File -> New File -> Quarto Document..` dropdown option; a menu will appear asking you for the document title, author, and preferred output type. You can select HTML. Call it "*week1*" and save it in your main folder.

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:



2.1 Practice: Dataset and Dataframes

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 certain 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.

2.1.1 Datasets in R

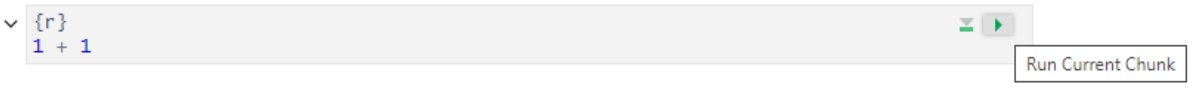
In R, we want a way of storing data where it feels just as if we had an Excel Spreadsheet where each row represents an observation and each column represents some information about that observation. We will call this object a `data.frame`, an R represention of a data set. The easiest way to understand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

```
library(dplyr)

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"
```

2.1.2 Importing Data in R

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 type, 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.

Argument	Default	Description
`sep`	`","`	Specifies the character that separates columns. For `read.csv()`, this is usually a comma.
`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)
```

2.2 Practice: Descriptive Statistics

2.2.1 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
```

2.2.2 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.

2.2.3 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.

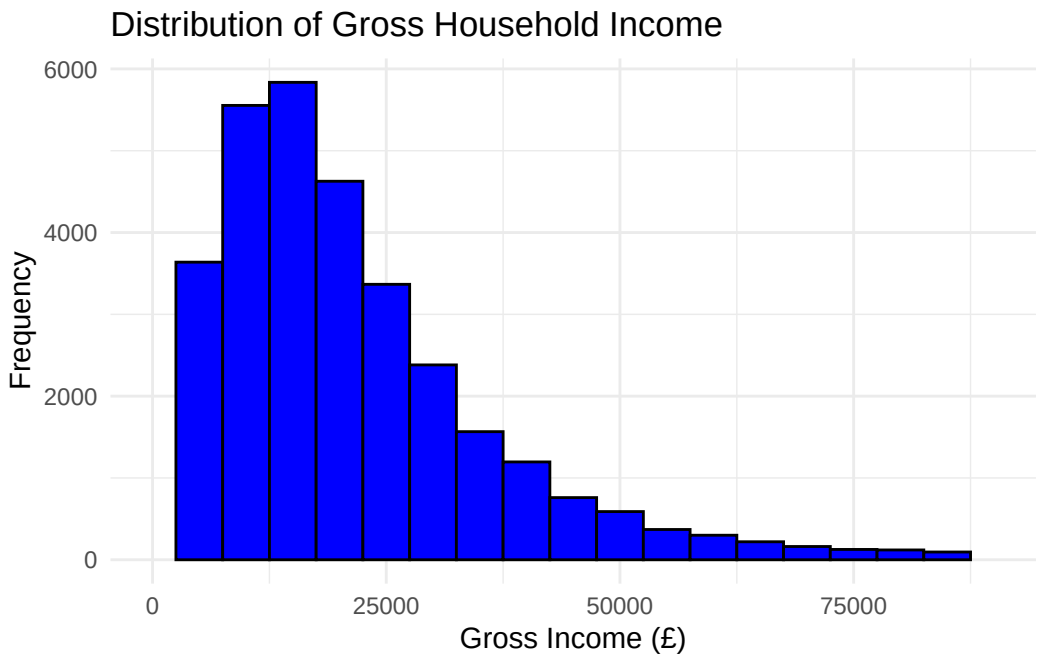
You need to select all cases (persons) that are independent, that is where the variable `dependent` has values different from `!=` "Dependent" or equal `==` "Independent".

```
# 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.

3 Lab: Correlation, Single, and Multiple Linear Regression

IMPORTANT: Before starting: verify that you have set the directory correctly and familiarise yourself with working with directories and loading files. These are the instructions.

Now create a .qmd project. We use the `File -> New File -> Quarto Document..` dropdown option; a menu will appear asking you for the document title, author, and preferred output type. You can select HTML. Call it "*week2*" and save it in your main folder.

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

In this week's practical, we will review how to calculate and visualise correlation coefficients between variables. This practical is split into two parts. The first part focuses on measuring and visualising the relationship between between **continuous variables**. The second part goes through the implementation of a Linear Regression Model, again between continuous variables.

Before getting into it, have a look at this resource, it really helps understand how regression models work.

The lecture's slides can be found here.

Before completing the practical, please take this quiz.

Learning Objectives:

- Visualise the association between two continuous variables using a scatterplot.
- Measure the strength of the association between two variables by calculating their correlation coefficient.
- Build a formal regression model.
- Understand how to estimate and interpret a multiple linear regression model.

Note on file paths: When calling the `read.csv` function, the path will vary depending on the location of the script it is being executed or your working directory (WD):

1. **If the script or your WD is in a sub-folder** (e.g., `labs`), use `"../data/Census2021/EW_DistrictPerc`
   The `..` tells R to go one level up to the main directory (`stats/`) and then access the `data/` folder. **Example**: `read.csv("../data/Census2021/EW_DistrictPercentages.csv")`.
2. **If the script is in the main directory** (e.g. inside `stats/`), you can access the data directly using `"data/Census2021/EW_DistrictPercentages.csv"`. Here, no `..` is necessary as the `data/` folder is directly accessible from the working directory. **Example**: `read.csv("data/Census2021/EW_DistrictPercentages.csv")`

3.1 Part I. Correlation

3.1.1 Data Overview: Descriptive Statistics:

Let's start by picking **one dataset derived from the English-Wales 2021 Census data**. You can choose one dataset that aggregates data either at a) county, b) district, or c) ward-level. Lower Tier Local Authority-, Region-, and Country-level data is also available in the data folder.

see also: https://canvas.liverpool.ac.uk/courses/77895/pages/census-data-2021

```
# Load necessary libraries
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 4.4.3
```

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
options(scipen = 999, digits = 4)  # Avoid scientific notation and round to 4 decimals globa

# load data - you main need to remove "../"
census <- read.csv("../data/Census2021/EW_DistrictPercentages.csv") # District level
```

We're using a (district/ward/etc.) level census dataset that includes:

- % of population with poor health (variable name: `pct_Very_bad_health`).
- % of population with no qualifications (`pct_No_qualifications`).
- % of male population (`pct_Males`).
- % of population in a higher managerial/professional occupation (`pct_Higher_manager_prof`).

First, let's get some descriptive statistics that help identify general trends and distributions in the data.

```r
# Summary statistics
summary_data <- census %>%
  select(pct_Very_bad_health, pct_No_qualifications, pct_Males, pct_Higher_manager_prof) %>%
  summarise_all(list(mean = mean, sd = sd))
summary_data
```

```
  pct_Very_bad_health_mean pct_No_qualifications_mean pct_Males_mean
1                    1.173                      17.9          48.97
  pct_Higher_manager_prof_mean pct_Very_bad_health_sd pct_No_qualifications_sd
1                        13.22                 0.3401                    3.959
  pct_Males_sd pct_Higher_manager_prof_sd
1       0.6605                     4.73
```

Q1. Complete the table below by specifying each variable type (continuous or categorical) and reporting its mean and standard deviation.

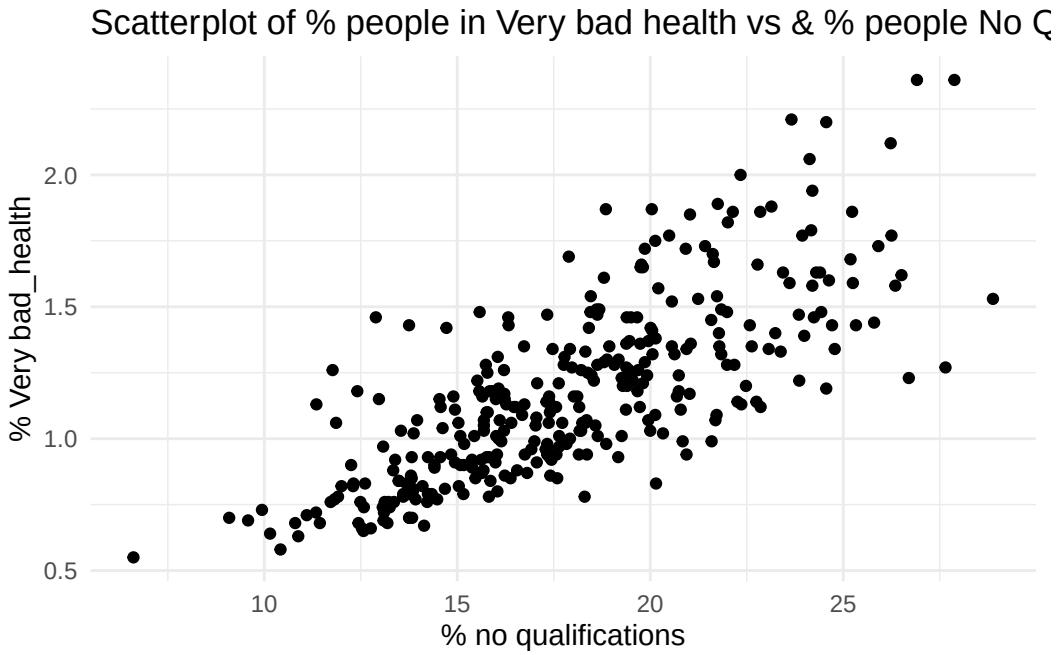
Variable Name	Type (Continuous or Categorical)	Mean	Standard Deviation
pct_Very_bad_health			
pct_No_qualifications			
pct_Males			
pct_Higher_manager_prof			

3.1.2 Simple visualisation for continuous data

You can visualise the relationship between two continuous variables using a scatter plot. Using the chosen census datasets, visualise the association between the % of population with bad health (`pct_Very_bad_health`) and each of the following:

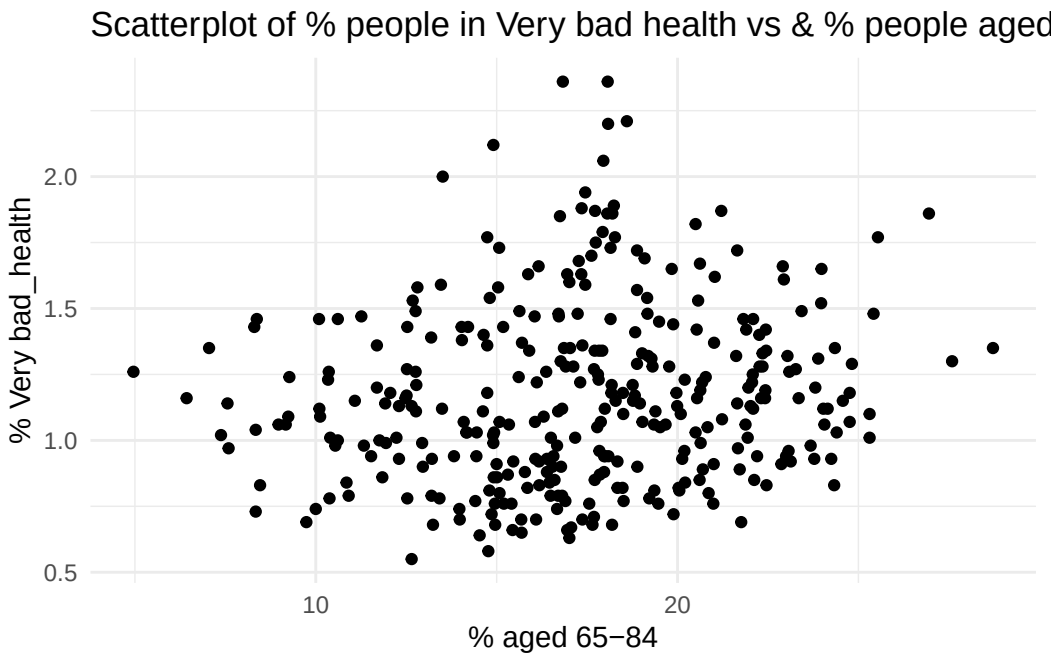
- the % of population with no qualifications (`pct_No_qualifications`);
- the % of population aged 65 to 84 (`pct_Age_65_to_84`);
- the % of population in a married couple (`pct_Married_opposite_sex_couple`);
- the % of population in a Higher Managerial or Professional occupation (`pct_Higher_manager_prof`).

```
# 1
ggplot(census, aes(x = pct_No_qualifications, y = pct_Very_bad_health)) +
    geom_point() +
    labs(title = paste("Scatterplot of % people in Very bad health vs & % people", "No Quali
         x = "% no qualifications", y = "% Very bad_health") +
    theme_minimal()
```

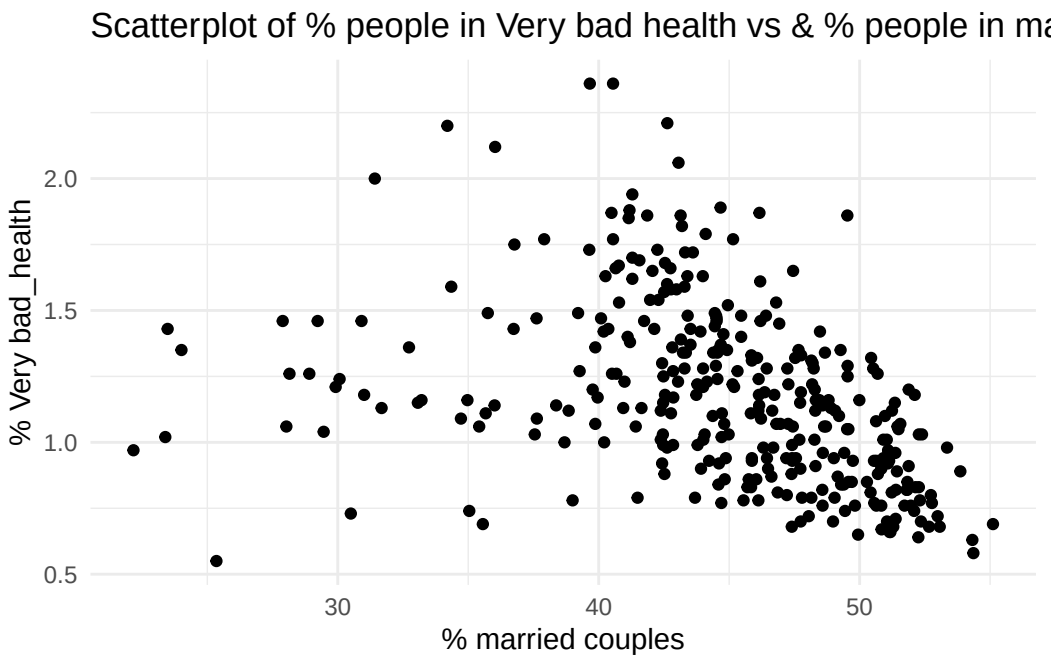



```
# 2
ggplot(census, aes(x = pct_Age_65_to_84, y = pct_Very_bad_health)) +
    geom_point() +
    labs(title = paste("Scatterplot of % people in Very bad health vs & % people", "aged betw
```

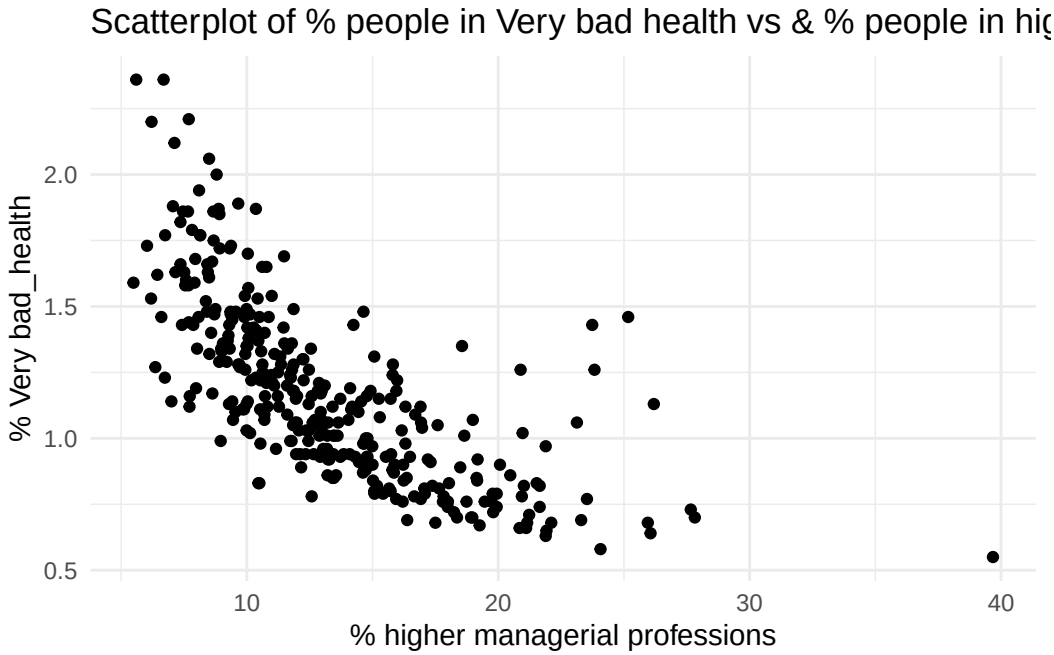
```
         x = "% aged 65-84", y = "% Very bad_health") +
    theme_minimal()
```




```
# 3
ggplot(census, aes(x = pct_Married_opposite_sex_couple, y = pct_Very_bad_health)) +
    geom_point() +
    labs(title = paste("Scatterplot of % people in Very bad health vs & % people", "in marri
         x = "% married couples", y = "% Very bad_health") +
    theme_minimal()
```




```
# 4
ggplot(census, aes(x = pct_Higher_manager_prof, y = pct_Very_bad_health)) +
    geom_point() +
    labs(title = paste("Scatterplot of % people in Very bad health vs & % people", "in highe
         x = "% higher managerial professions", y = "% Very bad_health") +
    theme_minimal()
```



Q2. Which of the associations do you think is strongest, which one is the weakest?

As noted, before, an observed association between two variables is no guarantee of causation. It could be that the observed association is:

- simply a chance one due to sampling uncertainty;
- caused by some third underlying variable which explains the spatial variation of both of the variables in the scatterplot;
- due to the inherent arbitrariness of the boundaries used to define the areas being analysed (the 'Modifiable Area Unit Problem').

Q3. Setting these caveats to one side, are the associations observed in the scatterplots suggestive of any causative mechanisms of bad health?

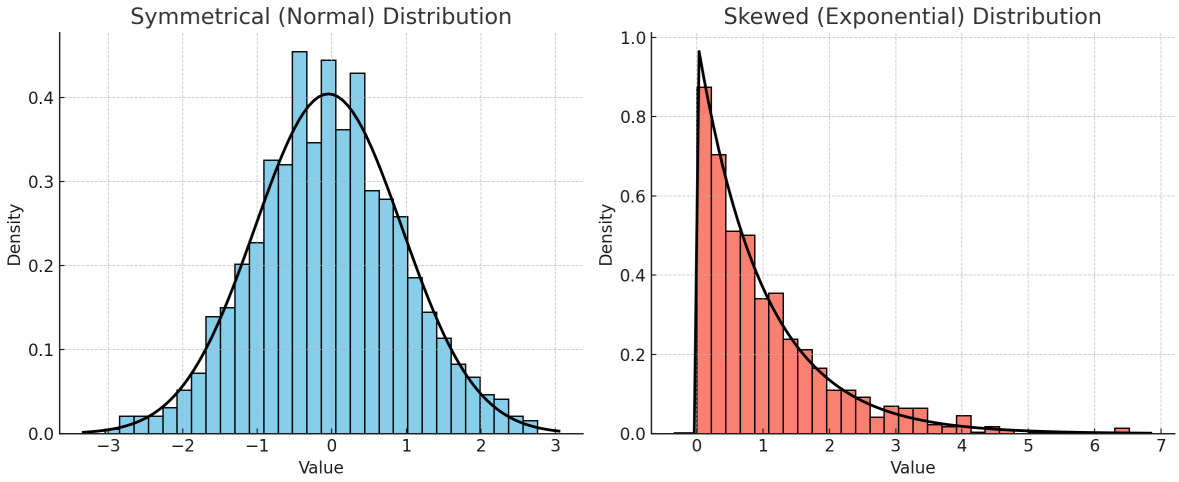
Rather than relying upon an impressionistic view of the strength of the association between two variables, we can measure that association by calculating the relevant correlation coefficient. The Table below identifies the statistically appropriate measure of correlation to use between two continuous variables.

Variable Data Type	Measure of Correlation	Range
Both symmetrically distributed	Pearson's	-1 to +1
One or both with a skewed distribution	Spearman's Rank	-1 to +1

Different Calculation Methods: Pearson's correlation assumes linear relationships and is suitable for symmetrically distributed (normally distributed) variables, measuring the strength of the linear relationship. Spearman's rank correlation, however, works on ranked data, so it's more suitable for skewed data or variables with non-linear relationships, measuring the strength and direction of a monotonic relationship.

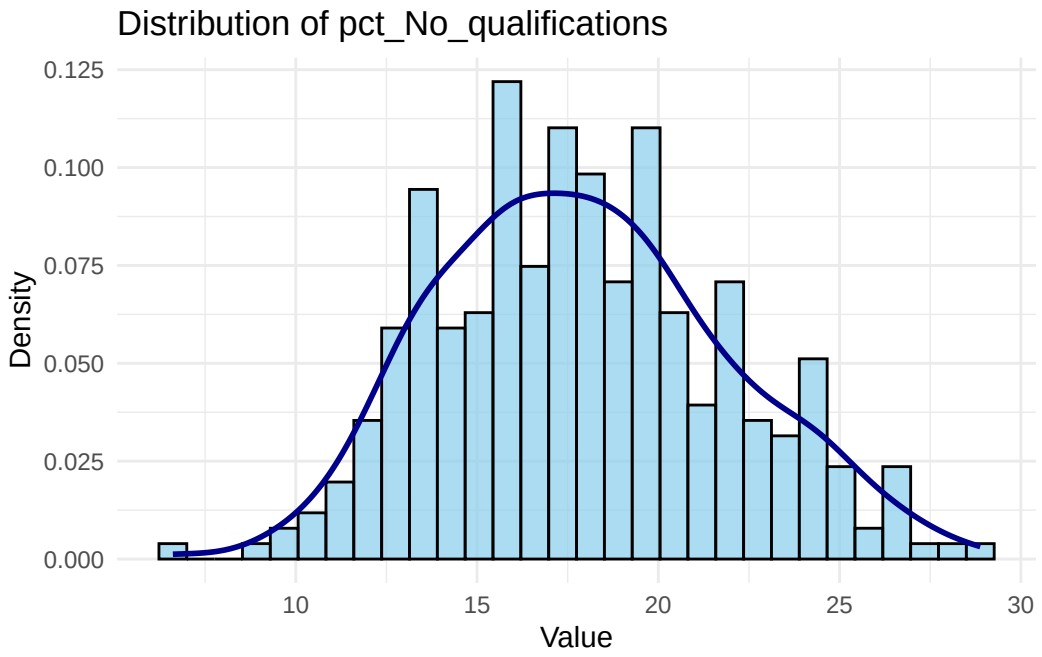
When calculating correlation for a single pair of variables, select the method that best fits their data distribution:

- Use **Pearson's** if both variables are symmetrically distributed.
- Use **Spearman's** if one or both variables are skewed.



You can check the distribution of a variable (e.g. `pct_No_qualifications` like this):

```
# Plot histogram with density overlay for a chosen variable (e.g., 'pct_No_qualifications')
ggplot(census, aes(x = pct_No_qualifications)) +
    geom_histogram(aes(y = after_stat(density)), bins = 30, color = "black", fill = "skyblue
    geom_density(color = "darkblue", linewidth = 1) +
    labs(title = "Distribution of pct_No_qualifications", x = "Value", y =  "Density") +
  theme_minimal()
```



When analyzing multiple pairs of variables, using different measures (Pearson for some pairs, Spearman for others) creates inconsistencies since Pearson and Spearman values aren't directly comparable in size due to their different calculation methods. To maintain consistency across comparisons, calculate **both Pearson's and Spearman's correlations** for each pair, e.g. do the trends align (both showing strong, weak, or moderate correlation in the same direction)? This consistency check can give confidence that the relationships observed are not dependent on the correlation method chosen. While in a report you'd typically include only one set of correlations (usually Pearson's if the relationships appear linear), calculating both can validate that your observations aren't an artifact of the correlation method.

Research Question 1: Which of our selected variables are most strongly correlated with % of population with bad health?

To answer this question, complete the Table below by editing/running this code:.

Pearson correlations

```
pearson_correlation <- cor(census$pct_Very_bad_health,
    census$pct_No_qualifications,use = "complete.obs", method = "pearson")

# Display the results
cat("Pearson Correlation:", pearson_correlation, "\n")
```

```
Pearson Correlation: 0.7619
```

Spearman correlations:

```
spearman_correlation <- cor(census$pct_Very_bad_health,
    census$pct_No_qualifications, use = "complete.obs", method = "spearman")

cat("Spearman Correlation:", spearman_correlation, "\n")
```

```
Spearman Correlation: 0.7781
```

Covariates	Pearson	Spearman
pct_Very_bad_health - pct_No_qualifications		
pct_Very_bad_health - pct_Age_65_to_84		
pct_Very_bad_health - pct_Married_opposite_sex_couple		
pct_Very_bad_health - pct_Higher_manager_prof		

What can you make of this numbers?

If you think you have found a correlation between two variables in our dataset, this doesn't mean that an association exists between these two variables in the population at large. The uncertainty arises because, by chance, the random sample included in our dataset might not be fully representative of the wider population.

For this reason, we need to verify whether the correlation is statistically significant,

```
# significance test for pearson, for example
pearson_test <- cor.test(census$pct_Very_bad_health,
    census$pct_No_qualifications, method = "pearson", use = "complete.obs")
pearson_test
```

```
    Pearson's product-moment correlation

data:  census$pct_Very_bad_health and census$pct_No_qualifications
t = 21, df = 329, p-value <0.0000000000000002
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.7127 0.8037
sample estimates:
   cor
0.7619
```

Look at https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/cor.test for details about the function. But in general, when calculating the correlation between two variables, a **p-value** accompanies the correlation coefficient to indicate the statistical significance of the observed association. This p-value tests the null hypothesis that there is no association between the two variables (i.e., that the correlation is zero).

When interpreting p-values, certain thresholds denote different levels of confidence. A p-value less than 0.05 is generally considered statistically significant at the 95% confidence level, suggesting that we can be 95% confident there is an association between the variables in the broader population. When the p-value is below 0.01, the result is significant at the 99% confidence level, meaning we have even greater confidence (99%) that an association exists. Sometimes, on research papers or tables significance levels are denoted with asterisks: one asterisk (*) typically indicates significance at the 95% level ($p < 0.05$), two asterisks (**) significance at the 99% level ($p < 0.01$), three asterisks (***) significance at the 99.99% level ($p < 0.01$).

Typically, p-values are reported under labels such as "Sig (2-tailed)," where "2-tailed" refers to the fact that the test considers both directions (positive and negative correlations). Reporting the exact p-value (e.g., $p = 0.002$) is more informative than using thresholds alone, as it gives a clearer picture of how strongly the data contradicts the null hypothesis of no association.

In a nutshell, lower p-values suggest a stronger statistical basis for believing that an observed correlation is not due to random chance. A statistically significant p-value reinforces confidence that an association is likely to exist in the wider population, though it does not imply causation.

3.1.3 Part. 2: Implementing a Linear Regression Model

A key goal of data analysis is to explore the potential factors of health at the local district level. So far, we have used cross-tabulations and various bivariate correlation analysis methods to explore the relationships between variables. One key limitation of standard correlation analysis is that it remains hard to look at the associations of an outcome/dependent variable to multiple independent/explanatory variables at the same time. Regression analysis provides a very useful and flexible methodological framework for such a purpose. Therefore, we will investigate how various local factors impact residents' health by building a multiple linear regression model in R.

We use `pct_Very_bad_health` as a proxy for residents' health.

Research Question 2: How do local factors affect residents' health?

Dependent (or Response) Variable:

- % of population with bad health (`pct_Very_bad_health`).

Independent (or Explanatory) Variables:

- % of population with no qualifications (`pct_No_qualifications`).
- % of male population (`pct_Males`).
- % of population in a higher managerial/professional occupation (`pct_Higher_manager_prof`).

Load some other Libraries

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v forcats   1.0.0     v stringr   1.5.1
v lubridate 1.9.3     v tibble    3.2.1
v purrr     1.0.2     v tidyr     1.3.1
v readr     2.1.5
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
library(broom)
```

and the data (if not loaded):

```
# Load dataset
census <- read.csv("../data/Census2021/EW_DistrictPercentages.csv")
```

Regression models are the standard method for constructing predictive and explanatory models. They tell us how changes in one variable (the target variable or independent variable, Y) are *associated with* changes in explanatory variables, or dependent variables, X_1, X_2, X_3 (X_n), etc. Classic linear regression is referred to *Ordinary least squares* (OLS) regression because they estimate the relationship between one or more independent variables and a dependent variable Y using a hyperplane (i.e. a multi-dimensional line) that minimises the sum of the squared difference between the observed values of Y and the values predicted by the model (denoted as $\hat{Y}$, Y-hat).

Having seen **Single Linear Regression** in class - where the relationship between one independent variable and a dependent variable is modeled - we can extend this concept to situations where more than one explanatory variable might influence the outcome. While single linear regression helps us understand the effect of **ONE** variable in isolation, real-world phenomena are often influenced by multiple factors simultaneously. Multiple linear regression addresses

this complexity by allowing us to model the relationship between a dependent variable and multiple independent variables, providing a more comprehensive view of how various explanatory variables contribute to changes in the outcome.

Here, regression allows us to examine the relationship between people's health rates and multiple dependent variables.

Before starting, we define two hypotheses:

- **Null hypothesis** (H_0): For each variable X_n, there is no effect of X_n on Y.
- **Alternative hypothesis** (H_1): There is an effect ofX_n on Y.

We will test if we can reject the null hypothesis.

3.1.4 Model fit

```r
# Linear regression model
model <- lm(pct_Very_bad_health ~ pct_No_qualifications + pct_Males + pct_Higher_manager_pro
summary(model)
```

```
Call:
lm(formula = pct_Very_bad_health ~ pct_No_qualifications + pct_Males +
    pct_Higher_manager_prof, data = census)

Residuals:
    Min      1Q  Median      3Q     Max
-0.4911 -0.1357 -0.0368  0.0985  0.7669

Coefficients:
                        Estimate Std. Error t value             Pr(>|t|)
(Intercept)              4.00293    0.87981    4.55            0.0000076 ***
pct_No_qualifications    0.05283    0.00591    8.94 < 0.0000000000000002 ***
pct_Males               -0.07353    0.01785   -4.12            0.0000479 ***
pct_Higher_manager_prof -0.01318    0.00494   -2.67                0.008 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.213 on 327 degrees of freedom
Multiple R-squared:  0.61,  Adjusted R-squared:  0.607
F-statistic:  171 on 3 and 327 DF,  p-value: <0.0000000000000002
```

Code explanation

`lm()` Function:

- `lm()` stands for "linear model" and is used to fit a linear regression model in R.
- The formula syntax `pct_Very_bad_health ~ pct_No_qualifications + pct_Males + pct_Higher_manager_prof` specifies a relationship between:
  - **Dependent Variable**: `pct_Very_bad_health`.
  - **Independent Variables**: `pct_No_qualifications`, `pct_Males`, and `pct_Higher_manager_prof`. The model is trained on the `data` dataset.

Storing the Model: The `model <-` syntax stores the fitted model in an object called `model`.

`summary(model)` provides a detailed output of the model's results, including:

- **Coefficients**: Estimates of the regression slopes (i.e., how each independent variableaffects `pct_Very_bad_health`).
- **Standard Errors**: The variability of each coefficient estimate.
- **t-values** and **p-values**: Indicate the statistical significance of the effect of each independent (explanatory) variable.

- **R-squared** and **Adjusted R-squared**: Show how well the independent variables explain the variance in the dependent variable.
- **F-statistic**: Tests the overall significance of the model.

We can focus only on certain output metrics:

```
# Regression coefficients
coefficients <- tidy(model)
coefficients
```

```
# A tibble: 4 x 5
  term                    estimate std.error statistic  p.value
  <chr>                      <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)               4.00     0.880        4.55 7.58e- 6
2 pct_No_qualifications     0.0528   0.00591      8.94 2.99e-17
3 pct_Males                -0.0735   0.0178      -4.12 4.79e- 5
4 pct_Higher_manager_prof  -0.0132   0.00494     -2.67 7.97e- 3
```

These are:

- **Regression Coefficient Estimates.**
- **P-values.**
- **Adjusted R-squared**.

3.1.5 How to interpret the output metrics

3.1.5.1 Regression Coefficient Estimates

The `Estimate` column in the output table tells us the rate of change between each dependent variable X_n and Y.

Intercept: In the regression equation, this is β_0 and it indicates the value of Y when X_n are equal to zero.

Slopes: These are the other regression coefficients of an independent variable, e.g. β_1, i.e. estimated average changes in Y for a one unit change in an independent variable, e.g. X_1, when all other dependent or explanatory variables are held constant.

There are two key points worth mentioning:

- **The unit of** X and Y: you need to know what the units are of the independent and dependent variables. For instance, one unit could be one year if you have an age variable, or a one percentage point if the variable is measured in percentages (all the variables in this week's practical).
- **All the other explanatory variables are held constant**. It means that the coefficient of an explanatory variable X_1 (e.g. β_1) should be interpreted as: a one unit change in X_1 is associated with β_1 units change in Y, keeping other values of explanatory variables (e.g. X_2, X_3) constant – for instance, X_2= 0.1 or X_3= 0.4.

For the independent variable X, we can derive how changes of 1 unit for the independent are associated with the changes in `pct_Very_bad_health`, for example:

- The association of `pct_No_qualifications` is positive and strong: each increase in 1% of `pct_No_qualifications` is associated with an increase of 0.05% of very bad health rate.
- The association of `pct_Males` is negative and strong: each decrease in 1% of `pct_Males` is associated with an increase of 0.07% of `pct_Very_bad_health` in the population in England and Wales.
- The association of `pct_Higher_manager_prof` is negative but weak: each decrease in 1% of `pct_Higher_manager_prof` is associated with an increase of 0.013% of `pct_Very_bad_health`.

3.1.5.2 P-values and Significance

The ***t tests*** of regression coefficients are used to judge the statistical inferences on regression coefficients, i.e. associations between independent variables and the outcome variable. For a t-statistic of a dependent variable, there is a corresponding ***p-value*** that indicates different levels of significance in the column `Pr(>|t|)` and the asterisks `*`.

- `***` indicates "changes in X_n are significantly associated with changes in Y at the <0.001 level".
- `**` suggests that "changes in X_n are significantly associated with changes in Y between the 0.001 and (<) 0.01 levels".
- Now you should know what `*` means: The significance is between the 0.01 and 0.05 levels, which means that we observe a less significant (but still significant) relationship between the variables.

P-value provide a measure of how significant the relationship is; it is an indication of whether the relationship between X_n and Y found in this data could have been found by chance. Very small p-values suggest that the level of association found here might **not** have come from a random sample of data.

In this case, we can say:

- Given that the p-value is indicated by `***`, changes in `pct_No_qualifications` and `pct_Males` are significantly associated with changes in `pct_Very_bad_health` at the <0.001 level; the association is highly statistically significant; we can be confident that the observed relationship between these variables and `pct_Very_bad_health` is not due to chance.
- Given that the p-value is indicated by `**`, changes in `pct_Higher_manager_prof` are significantly associated with changes in `pct_Very_bad_health` at the 0.001 level. This means that the association between the independent and dependent variable is not one that would be found by chance in a series of random sample 99.999% of the time.

In both cases we can then confidently reject the **Null** hypothesis (H_0: no association between dependent and independent variables exist).

Remember, If the *p-value* of a coefficient is smaller than 0.05, that coefficient is statistically significant. In this case, you can say that the relationship between this independent variable and the outcome variable is *statistically* significant. Contrarily, if the *p-value* of a coefficient is larger than 0.05 you can conclude that there is no evidence of an association or relationship between the independent variable and the outcome variable.

3.1.5.3 R-squared and Adjusted R-squared

These provide a measure of model fit. They are calculated as the difference between the actual value of Y and the value predicted by the model. The **R-squared** and **Adjusted R-squared** values are statistical measures that indicate how well the independent variables in your model explain the variability of the dependent variable. Both R-squared and Adjusted R-squared help us understand how closely the model's predictions align with the actual data. An R-squared of 0.6, for example, indicates that 60% of the variability in Y is explained by the independent variables in the model. The remaining 40% is due to other factors not captured by the model.

Adjusted R-squared also measures the goodness of fit, but it adjusts for the number of independent variables in the model, accounting for the fact that adding more variables can artificially inflate R-squared without genuinely improving the model. This is especially useful when comparing models with different numbers of independent variables. If Adjusted R-squared is close to or above 0.6, as in your example, it implies that the model has a **strong explanatory power** while not being overfit with unnecessary explanatory variables.

A high R-squared and Adjusted R-squared indicate that the model captures much of the variation in the data, making it more reliable for predictions or for understanding the relationship between Y and the explanatory variables. However Low R-squared values suggest (e.g. 0.15) that the model might be missing important explanatory variables or that the relationship between Y and the selected explanatory variables is not well-captured by a linear approach.

An R-squared and Adjusted R-squared over 0.6 are generally seen as signs of a **well-fitting model** in many fields, though the ideal values can depend on the context and the complexity of the data.

3.1.6 Interpreting the Results

```
coefficients
```

```
# A tibble: 4 x 5
  term                    estimate std.error statistic  p.value
  <chr>                      <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)               4.00     0.880        4.55 7.58e- 6
2 pct_No_qualifications     0.0528   0.00591      8.94 2.99e-17
3 pct_Males                -0.0735   0.0178      -4.12 4.79e- 5
4 pct_Higher_manager_prof  -0.0132   0.00494     -2.67 7.97e- 3
```

Q4. Complete the table above by filling in the coefficients, t-values, p-values, and indicating if each variable is statistically significant.

Variable Name	Coefficients	t-values	p-values	Significant?
pct_No_qualifications				
pct_Males				
pct_Higher_manager_prof				

From the lecture notes, you know that the Intercept or Constant represents the estimated average value of the outcome variable when the values of all independent variables are equal to zero.

Q5. When values of `pct_Males`, `pct_No_qualifications` and `pct_Higher_manager_prof` are all *zero*, what is the % of population with very bad health? Is the intercept term meaningful? Are there any districts (or zones, depending on the dataset you chose) with zero percentages of persons with no qualification in your data set?

Q6. Interpret the regression coefficients of `pct_Males`, `pct_No_qualifications` and `pct_Higher_manager_prof`. Do they make sense?

3.1.7 Identify factors of % bad health

Now combine the above two sections and identify factors affecting the percentage of population with very bad health. Fill in each row for the direction (positive or negative) and significance level of each variable.

Variable Name	Positive or Negative	Statistical Significance
pct_No_qualifications		
pct_Higher_manager_prof		
pct_Males		

Q7. Think about the potential conclusions that can be drawn from the above analyses. Try to answer the research question of this practical: How do local factors affect residents' health? Think about causation *vs* association and consider potential confounders when interpreting the results. How could these findings influence local health policies?

3.2 Part C: Practice and Extension

If you haven't understood something, if you have doubts, even if they seem silly, ask.

1. Finish working through the practical.
2. Revise the material.
3. Extension activities (optional): Think about other potential factors of very bad health and test your ideas with new linear regression models.

4 Lab: Correlation and Multiple Linear Regression with Qualitative Variables

The lecture's slides can be found here.

In last week, we introduced Multiple Linear Regression (MLR) - a statistical method that models the relationship between a dependent variable and two or more independent variables, allowing researchers to examine how various predictors jointly influence an outcome. By using the following R, we create and interpret the model:

```
model <- lm(pct_Very_bad_health ~ pct_No_qualifications + pct_Males + pct_Higher_manager_pro
data = census)

summary(model)
```

In a regression model, independent/predictor variables could be continuous or categorical (or qualitative). While continuous variables capture quantitative effects, **categorical variables provide insights into differences across groups**. When we say categorical variables, we normally mean:

- Nominal Data: categorical data without natural order. E.g. Gender, Colour, Country…
- Ordinal Data: categorical data with a meaningful order. E.g. Education level, Customer satisfaction, Grade…

By blending continuous and categorical predictors, MLR with categorical variables enhances the model's ability to reflect real-world complexities and improves interpretability, as it allows analysts to assess how each category or group within a independent variable influences the dependent variable.

For most categorical (especially the *nominal*) variables, they cannot be included in the regression model directly as a continuous independent variable. Instead, these qualitative independent variables should be included in regression models by using the **dummy variable** approach, transforming categorical information into a numerical format suitable for regression analysis.

However, R provides a powerful way, by automatively handling with such process when the categorical variable is designated as a factor and to be included in the regression model. This makes it much easier for you to use categorical variables in the regression model to assess the effects of categorical groupings on the dependent variable alongside continuous predictors.

Learning Objectives:

In this week's practical we are going to

- Analysis of categorical/qualitative variables
- Estimate and interpret a multiple linear regression model with categorical variables
- Make predictions using a regression model

4.1 Analysis categorical variables

Recall in Week 2, you get familiar to R by using the Census data. Today we will explore both the Family Resource Survey (FRS) and the Census data by using their categorical variables. You should already have your Census data in your local drive folder under the path of '/data/Census2021/'; for the FRS datasets, please click the links to download the datasets from Canvas. Please download both the datasets of 'FRS16-17_labels.csv' and 'FRS_dictionary.xlxs' for today's practical. You may create a new folder named 'FRS' under the '/data/' along with the Census2021 folder, and put the newly downloaded .csv files at the '/data/FRS/' folder for later use.

To start today's practical session, we first will use 'FRS16-17_labels.csv'. You can open the .csv file in Excel and find itsdifferent from the Census dataset, the FRS dataset contains many qualitative/categorical variables, such as hh_tenure (Housing Tenure), happy (How happy did you feel yesterday?), health (How is your health in general) and etc.. To know the meaning of all the column names and the values in cell, you may need to reference to the 'FRS_dictionary.xlsx' to help your understanding. Recall our lecture, these categorical variable will need to be treated as Dummy variable in the regression model. In R, the variable type of qualitative/categorical variables is called '**factor**'.

As usual we first load the necessary libraries.

Some tips to avoid R returning can't find data errors:

Check your working directory by

```
getwd()
```

Check the relative path of your data folder on your PC/laptop, make sure you know the relative path of your data from your working directory, returned by `getwd()`.

Library knowledge used in today:

- `dplyr`: a basic library provides a suite of functions for data manipulation

- **ggplot2**: a widely-used data visualisation library to help you create nice plots through layered plotting.
- **tidyverse**: a collection of R packages designed for data science, offering a cohesive framework for data manipulation, visualization, and analysis. Containing dyplyr, ggplot2 and other basic libraries.
- **broom**: a part of the tidyverse and is designed to convert statistical analysis results into tidy data frames.
- **forcats**: designed to work with factors, which are used to represent categorical data. It simplifies the process of creating, modifying, and ordering factors.
- **vcd**: visualise and analyse categorical data.

A useful shortcut to format your code: select all your code lines, use Ctrl+Shift+A for automatically format them in a tidy way.

4.1.1 Data overview

```
if(!require("dplyr"))
  install.packages("dplyr",dependencies = T)
```

```
Loading required package: dplyr
```

```
Attaching package: 'dplyr'
```

```
The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
# Load necessary libraries
if(!require("ggplot2"))
  install.packages("ggplot2",dependencies = T)
```

```
Loading required package: ggplot2
```

```
if(!require("broom"))
  install.packages("broom",dependencies = T)
```

```
Loading required package: broom
```

```
library(dplyr)
library(ggplot2)
library(broom)
```

Or we can use library `tidyverse` which includes `ggplot2`, `dplyr`,`broom` and other foundamental libraries together already, remember you need first install the package if you haven't by using `install.packages("tidyverse")`.

```
if(!require("tidyverse"))
  install.packages("tidyverse",dependencies = T)
```

```
Loading required package: tidyverse
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v forcats   1.0.0     v stringr   1.5.1
v lubridate 1.9.3     v tibble    3.2.1
v purrr     1.0.2     v tidyr     1.3.1
v readr     2.1.5
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
library(tidyverse)
```

We will also use forcat library, so

```
if(!require("forcats"))
  install.packages("forcats")

library(forcats)
```

Exactly as you did in previous weeks, we first load in the dataset:

```
frs_data <- read.csv("../data/FRS/FRS16-17_labels.csv")
```

Recall in previous weeks, we used the following code to overview the dataset. Familiar yourself again by using them:

```
View(frs_data)
```

and also `summary()` to produce summaries of each variable

```
summary(frs_data)
```

You may notice that for the numeric variables such as *hh_income_gross* (household gross income) and *work_hours*(worked hours per week), the `summary()` offers useful descriptive statistics. While for the qualitative information, such as *age_group* (age group), *highest_qual (*Highest educational qualification), *marital_status (*Marital status) and *nssec (*Socio-economic status), the `summary()` function is not that useful by providing mean or median values.

Performing descriptive analysis for categorical variables or qualitative variables, we focus on summarising the frequency and distribution of categories within the variable. This analysis helps understand the composition and diversity of categories in the data, which is especially useful for identifying patterns, common categories, or potential data imbalances.

```
# Frequency count
table(frs_data$age_group)
```

```
  0-4 05-10 11-15 16-19 20-24 25-29 30-34 35-39 40-44 45-49 50-54 55-59 60-64
 2914  3575  2599  1858  1929  2353  2800  2840  2790  2883  2975  2767  2775
65-69 70-74   75+
 2990  2354  3743
```

```
table(frs_data$highest_qual)
```

```
A-level or equivalent       Degree or above       Dependent child
                 5260                  9156                 10298
   GCSE or equivalent            Not known                 Other
                 9729                  6820                  2882
```

```
table(frs_data$marital_status)
```

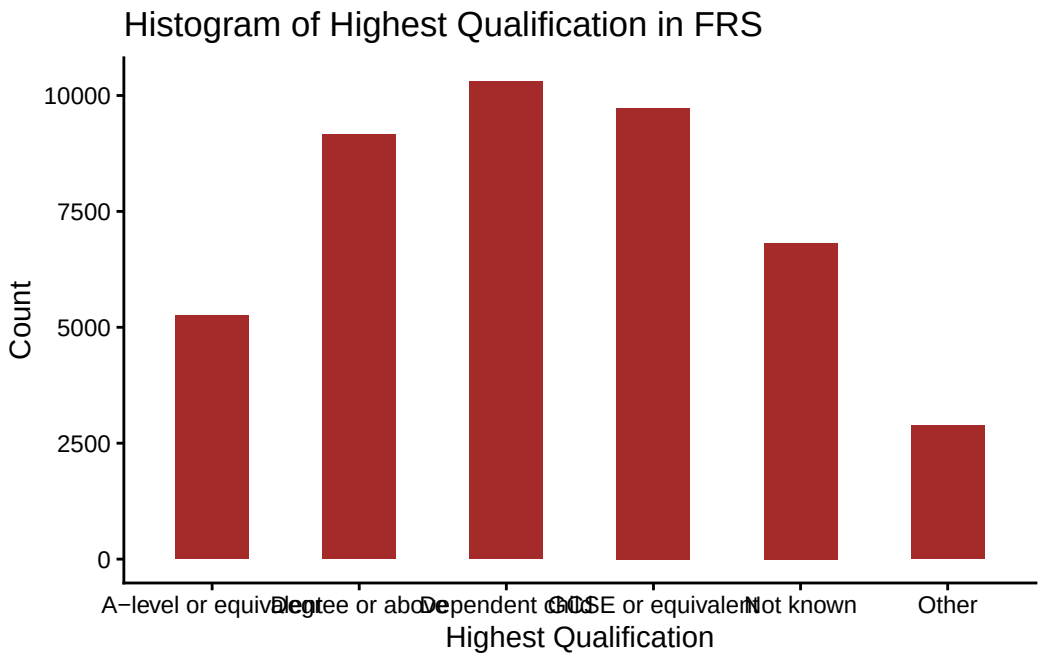
```
                        Cohabiting Divorced/civil partnership dissolved
                              4015                                  2199
          Married/Civil partnership                             Separated
                             18195                                   747
                            Single                               Widowed
                             16663                                  2326
```

```
table(frs_data$nssec)
```

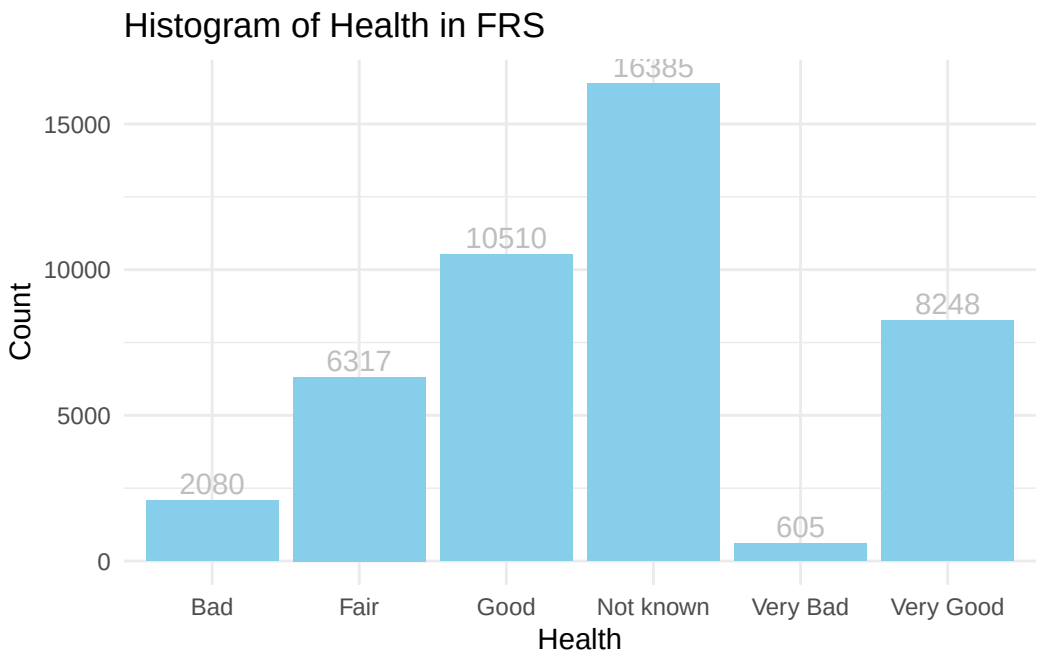
```
                             Dependent child
                                       10299
                           Full-time student
                                         963
              Higher professional occupations
                                        3004
                    Intermediate occupations
                                        4372
         Large employers and higher managers
                                        1025
Lower managerial and professional occupations
                                        8129
  Lower supervisory and technical occupations
                                        2400
        Never worked or long-term unemployed
                                        1516
                          Not classifiable
                                         107
                         Routine occupations
                                        4205
                    Semi-routine occupations
                                        5226
      Small employers and own account workers
                                        2899
```

By using ggplot2, it is easy to create some nice descriptive charts for the categorical variables, such like what you did for the continuous variables last week.

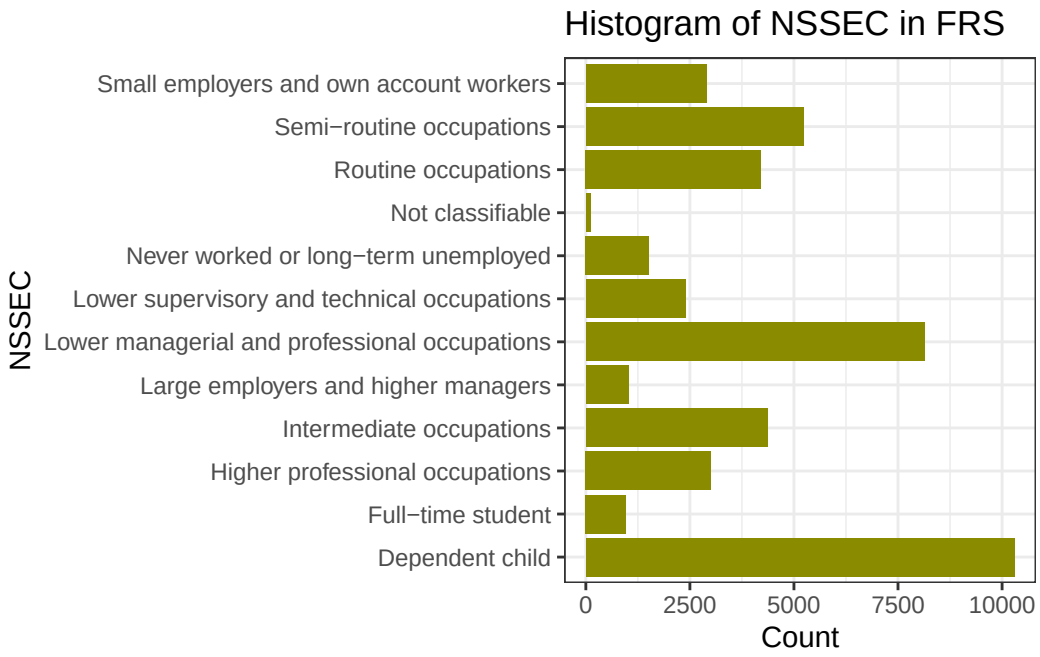
```
ggplot(frs_data, aes(x = highest_qual)) +
  geom_bar(fill="brown",width=0.5) +
  labs(title = "Histogram of Highest Qualification in FRS", x = "Highest Qualification", y =
  theme_classic()#choose theme type, try theme_bw(), theme_minimal() see differences
```




```
ggplot(frs_data, aes(x = health)) +
  geom_bar(fill="skyblue") +
  geom_text(stat = "count", aes(label = ..count..),vjust = -0.3,colour = "grey")+ #add text
  labs(title = "Histogram of Health in FRS", x = "Health", y = "Count")+#set text info
  theme_minimal()
```

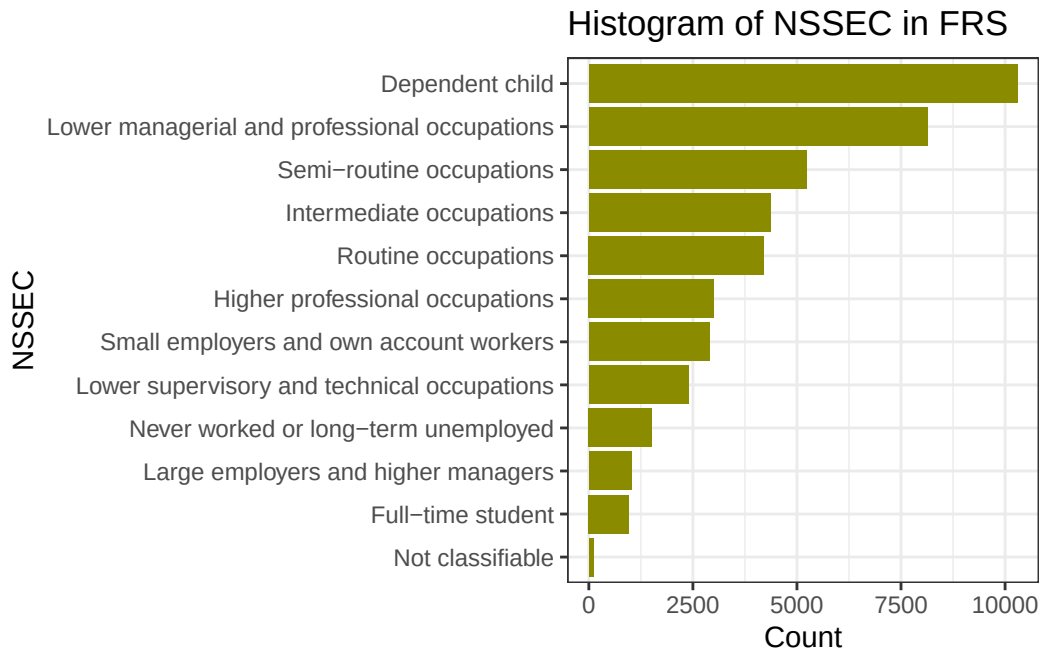



```
ggplot(frs_data, aes(x = nssec)) +
  geom_bar(fill = "yellow4") +
  labs(title = "Histogram of NSSEC in FRS", x = "NSSEC", y = "Count") +
  coord_flip()+ #Flip the Axes, add a # in front of this line, to make the code in gray and y
  theme_bw()
```



If we want to reorder the Y axis by from highest to lowest, we use the functions in `forcats` library. `fct_infreq()`: orders by the value's frequency of the variable `nssec`. `fct_rev()`: reverses the order to go from highest to lowest.

```
ggplot(frs_data, aes(x = fct_rev(fct_infreq(nssec)))) +
  geom_bar(fill = "yellow4") +
  labs(title = "Histogram of NSSEC in FRS", x = "NSSEC", y = "Count") +
  coord_flip()+ #Flip the Axes, add a # in front of this line, to make the code in gray and y
  theme_bw()
```



You can change the variables in ggplot() to make your own histogram chart for the variables you are interested in. You will learn more of visualisation methods in Week 5's practical.

4.1.2 Correlation

Q1. Which of the associations do you think is strongest? Which is the weakest?

As before, rather than relying upon an impressionistic view of the strength of the association between two variables, we can measure that association by calculating the relevant correlation coefficient.

To calculate the correlation between categorical data, we first use Chi-squared test to assess the independence between pairs of categorical variables, then we use Cramer's V to measures the strength of association - the correlation coefficents in R.

Pearson's chi-squared test (2) is a statistical test applied to sets of categorical data to evaluate how likely it is that any observed difference between the sets arose by chance. If the p-value is low (typically < 0.05), it suggests a significant association between the two variables.

```
chisq.test(frs_data$health,frs_data$happy)
```

```
Warning in chisq.test(frs_data$health, frs_data$happy): Chi-squared
approximation may be incorrect
```

```
    Pearson's Chi-squared test

data:  frs_data$health and frs_data$happy
X-squared = 45594, df = 60, p-value < 2.2e-16
```

If you see a warning message of Chi-squared approximation may be incorrect. This is because some expected frequencies in one or more cells of the cross-tabular (health * happy) are too low. The df means degrees of freedom and it related to the size of the table and the number of categories in each variable. The most important message from the output is the estimated p-value, which shows as p-value $<$ 2.2e-16 (2.2 with 16 decimals move to the left, it is a very small number so written in scientific notation). P-value of the chi-squared test is far smaller than 0.05, so we can say the correlation is statistically significant.

Cramér's V is a measure of association for categorical (nominal or ordinal) data. It ranges from 0 (no association) to 1 (strong association). The main downside of using Cramer's V is that no information is provided on whether the correlation is positive or negative. This is not a problem if the variable pair includes a nominal variable but represents an information loss if the both variables being correlated are ordinal.

```r
# Install the 'vcd' package if not installed
if(!require("vcd"))
install.packages("vcd", repos = "https://cran.r-project.org", dependencies = T)
```

```
Loading required package: vcd
```

```
Warning: package 'vcd' was built under R version 4.4.2
```

```
Loading required package: grid
```

```r
library(vcd)

# creat the crosstable
crosstab <- table(frs_data$health, frs_data$happy)

# Calculate Cramér's V
assocstats(crosstab)
```

```
                    X^2 df P(> X^2)
Likelihood Ratio 54036 60        0
Pearson          45594 60        0
```

```
Phi-Coefficient   : NA
Contingency Coeff.: 0.713
Cramer's V        : 0.454
```

```
#you can also directly calculate the assoication between variables
assocstats(table(frs_data$health, frs_data$age_group))
```

```
                   X^2 df P(> X^2)
Likelihood Ratio 26557 75        0
Pearson          23854 75        0

Phi-Coefficient   : NA
Contingency Coeff.: 0.592
Cramer's V        : 0.329
```

Research Question 1. Which of our selected person-level variables is most strongly correlated with an individual's health status?

Use the codes of Chi-test and Cramer's V to answer this question by completing Table 1.

Table 1 Person-level correlations with health status

Covariates		**Correlation Coefficient** *Cramer's V*	**Statistical Significance** *p-value*
health	*age_group*		
Health	*highest_qual*		
health	*marital_status*		
Health	*nssec*		

4.2 Income inequality with respect to gender and health status

In this section, we will work with individual-level data ("FRS 2016-17_label.csv") to explore income inequality with respect to gender and health status.

To explore income inequality, we need to work with a data set excluding dependent children. In addition, we look at individuals who are the representative persons of households. Therefore, we will select cases (or samples) that meet both conditions.

We want R to select persons only if they are the representative persons of households and they are not dependent children. The involved variables are `hrp` and `Dependent` for the categories

"Household Reference Person" and "independent", you can select the appropriate cases. We also want to exclude the health variable reported as "Not known".

```
frs_df <- frs_data %>% filter(hrp == "HRP" &
                                dependent == "Independent" &
                                health != "Not known")
```

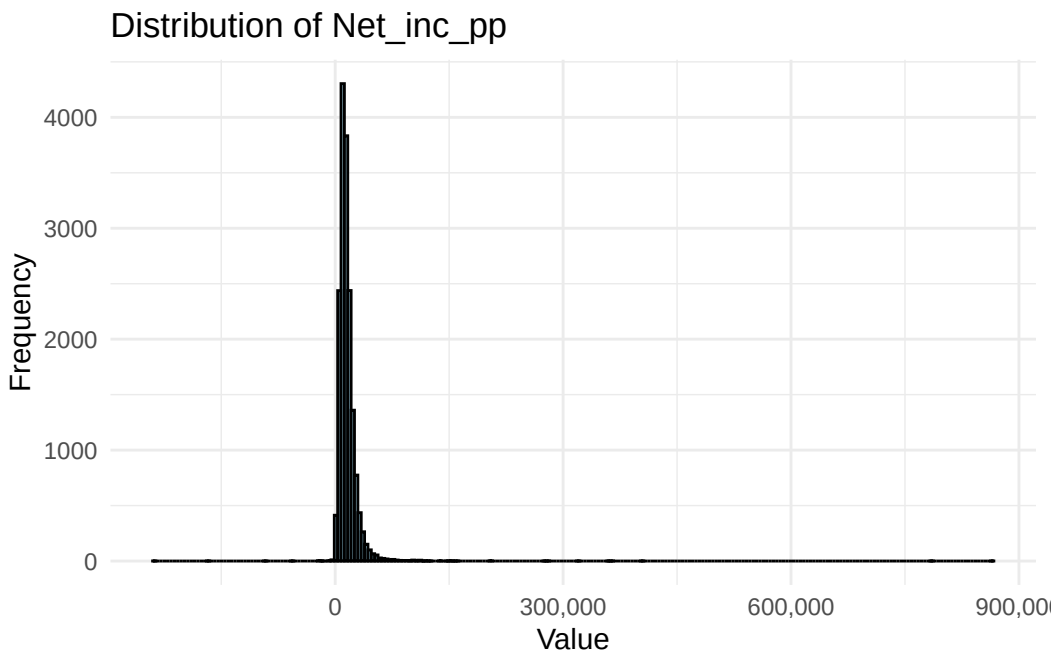
Then, we create a new numeric variable `Net_inc_perc` indicate net income per capita as our dependent variable:

```
frs_df$Net_inc_pp = frs_df$hh_income_net / frs_df$hh_size
summary(frs_df$Net_inc_pp)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-238160    9074   13347   15834   19136  864812
```

The distribution of the net household income per capita can be visualised by using `ggplot()`

```
ggplot(frs_df, aes(x = Net_inc_pp)) +
  geom_histogram(
    bins = 250, #A higher bins value means more, narrower bars, covers smaller range of value
    color = "black",
    fill = "skyblue",
    alpha = 0.7
  ) +   labs(title = "Distribution of Net_inc_pp", x = "Value", y =  "Frequency") +   scale_x
  theme_minimal()
```



Our two qualitative independent variables "`sex`" and "`health`". Let's first know what they look like:

```
table(frs_df$sex)
```

```
Female   Male
  7647   9180
```

```
table(frs_df$health)
```

```
      Bad      Fair      Good  Very Bad Very Good
     1472      4253      6277       426      4399
```

Remember in the lecture, what we did in the Region long-term illness before we put the categorical variable Region into the regression model? Yes. First, make sure they are in factor type and Second, decide the reference category. Here, I will use Female and Very Bad health status as my base categories. You can decide what you wish to use. This time, I use the following codes to combine these two steps in one line.

```r
frs_df$sex <- fct_relevel(as.factor(frs_df$sex), "Female")
frs_df$health <- fct_relevel(as.factor(frs_df$health), "Very Bad")
```

Implement the regression model with the two qualitative independent variables.

```r
model_frs <- lm(Net_inc_pp ~ sex + health, data = frs_df)
summary(model_frs)
```

```
Call:
lm(formula = Net_inc_pp ~ sex + health, data = frs_df)

Residuals:
    Min      1Q  Median      3Q     Max
-255133   -6547   -2213    3515  845673

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      12115.5      762.9  15.881  < 2e-16 ***
sexMale           2091.2      240.6   8.691  < 2e-16 ***
healthBad         -102.8      854.3  -0.120 0.904205
healthFair        1051.3      789.0   1.332 0.182751
healthGood        2766.0      777.4   3.558 0.000375 ***
healthVery Good   4931.8      787.8   6.260 3.95e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15530 on 16821 degrees of freedom
Multiple R-squared:  0.01646,   Adjusted R-squared:  0.01616
F-statistic: 56.29 on 5 and 16821 DF,  p-value: < 2.2e-16
```

Same of what you have learnt in Week 2, the code explanation:

`lm()` Function:

- `lm()` stands for "linear model" and is used to fit a linear regression model in R.
- The formula syntax `Net_inc_pp ~ Sex + health` specifies a relationship between:
  - **Dependent Variable**: `Net_inc_pp`.
  - **Independent Variables**: `Sex`, and `health`. The model is trained on the `data = frs_df` dataset.

Storing the Model: The `model <-` syntax stores the fitted model in an object called `model`.

`summary(model)` provides a detailed output of the model's results, including:

- **Coefficients**: Estimates of the regression slopes (i.e., how each independent variableaffects `Net_inc_pp`).
- **Standard Errors**: The variability of each coefficient estimate.
- **t-values** and **p-values**: Indicate the statistical significance of the effect of each independent (explanatory) variable.
- **R-squared** and **Adjusted R-squared**: Show how well the independent variables explain the variance in the dependent variable.
- **F-statistic**: Tests the overall significance of the model.

The result can be formatted by:

```
library(broom)
tidy(model_frs)
```

```
# A tibble: 6 x 5
  term            estimate std.error statistic  p.value
  <chr>              <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)       12116.      763.    15.9   2.21e-56
2 sexMale            2091.      241.     8.69  3.90e-18
3 healthBad          -103.      854.    -0.120 9.04e- 1
4 healthFair         1051.      789.     1.33  1.83e- 1
5 healthGood         2766.      777.     3.56  3.75e- 4
6 healthVery Good    4932.      788.     6.26  3.95e-10
```

Q2. What conclusions could be drawn in terms of income inequalities with respect to gender and health status? Also think about the statistical significance of these differences.

The interpretation of this table is again similar to what you have learnt in Week2, but you may find that R has automatically treat your categorical/qualitative variable Sex and Health as dummy variables, so in the output, you see sexMale, the reason there is no sexFemale is because I requested R to use sex=Female as the reference category. You also wee healthBad, healthFair, healthGood, and healthVery Good, but you didn't see health Very Bad because agaion I request `frs_df$health <- fct_relevel(as.factor(frs_df$health), "Very Bad")` .

For the independent variable, the coefficient estimates of them need to be interpreted by comparing to the reference category: Male and Very Bad health. We can derive how the differences between different independent variable categories are associated with the changes in `Net_inc_pp`, for example:

- The association of `sexMale`is positive and strong: compare to sex=Female, `sexMale` is associated with an increase of £2,091 net income per capita, here we see some gender inequality;
- The association of `healthBad` and `healthFair` arenot statistically significant, which means we cannot draw any conclusion of the relationships between `healthBad` and `Net_inc_pp` or `healthFair` and `Net_inc_pp` from the model ;
- Then, the association of `healthGood` and `healthVeryGood` are both positive and strong: `healthGood` is associated with an increase of £2,766 net income per capita compared to `healthVeryBad`, and `healthVeryGood` is associated with an increase of £4,931 net income per capita compared to `healthVeryBad` . Here you may draw some useful conclusion on who the health condition impact the inequality of income.

We then read R-squared and Adjusted R-squared to evaluate the model fit. We see both values are only around 0.016, which means in the model `Net_inc_pp ~ Sex + health` , the independent variable `Sex + health` can only explain 1.6% of the variation of our dependent variable `Net_inc_pp`. This is a relatively poor performance, and the model cannot be used to do any prediction of `Net_inc_pp` by using just sex and health. This also suggests us that to fully explain the `Net_inc_pp` , we may need to add in more variables, such as education level, occupation, age, etc. Although the model is not solid for any prediction, the coefficients and significant conclusion from the model are still very useful.

4.3 Implementing a linear regression model with a qualitative independent variable

To gain further practice on using linear regression model with qualitative variables as independent variables, we follow the model you created last week, to add in the qualitative/categorical variable Region from the Census 2021 dataset. You should already have the dataset **EW_DistrictPercentages.csv**, if not, you can download them from Canvas. Please put the .csv file in you '/data/Census2021' folder.

Research Question 2: How does health vary across regions in the UK?

The practical is split into two main parts. The first focuses on implementing a linear regression model with a qualitative independent variable. **Note that** you need first to set the reference category (baseline) as the outcomes of the model reflects the **differences** between categories with the baseline. The second part focuses prediction based the estimated linear regression model.

First we load the UK district-level census dataset.

```
# load data
LAcensus <- read.csv("../data/Census2021/EW_DistrictPercentages.csv") # Local authority level
```

Using the district-level census dataset "**EW___DistrictPercentages.csv**". the variable "Region" (labelled as Government Office Region) is used to explore regional inequality in health.

Familiar yourself with the dataset by using the same codes as last week:

```
#view the data
View(LAcensus)
glimpse(LAcensus)
```

The names() function returns all the column names.

```
names(LAcensus)
```

The dim() function can merely returns the number of rows and number of columns.

```
dim(LAcensus)
```

```
[1] 331 161
```

There are 331 rows and 151 columns in the dataset. It would be very hard to scan throught the data if we use so many variables altogether. Therefore, we can select several columns to tailor for this practical. You can of course include other variables you are interested in also by their names:

```
df <- LAcensus %>% select(c("pct_Longterm_sick_or_disabled",
                            "pct_No_qualifications",
                            "pct_Males",
                            "pct_Higher_manager_prof",
                            "Region"))
```

Simply descriptive of this new data

```
summary(df)
```

```
 pct_Longterm_sick_or_disabled pct_No_qualifications   pct_Males
 Min.   :1.330                 Min.   : 6.61         Min.   :46.77
 1st Qu.:2.865                 1st Qu.:15.06         1st Qu.:48.62
 Median :3.810                 Median :17.63         Median :48.98
 Mean   :4.013                 Mean   :17.90         Mean   :48.97
 3rd Qu.:4.800                 3rd Qu.:20.41         3rd Qu.:49.30
```

```
 Max.   :9.110                 Max.   :28.88         Max.   :55.02
 pct_Higher_manager_prof    Region
 Min.   : 5.49           Length:331
 1st Qu.: 9.79           Class :character
 Median :12.34           Mode  :character
 Mean   :13.22
 3rd Qu.:15.80
 Max.   :39.68
```

Now we can retrieve the "Region" column from the data frame by simply use **df$Region**. But what if we want to understand the data better, like the following questions?

Q3. How many categories do the variable "Region" entail? How many local authority districts does each region include?

Simply use the function table() would return you the answer.

```
table(df$Region)
```

```
                    East            East Midlands                   London
                      45                       35                       33
              North East              North West              South East
                      12                       39                       64
              South West                   Wales            West Midlands
                      30                       22                       30
Yorkshire and The Humber
                      21
```

The `table()` function tells us that this data frame contains 10 regions, and the number of LAs belongs to each region.

R can only include the categorical variables in the **factor type, so we set the column *Region* in `factor()`

```
df$Region<- factor(df$Region)
```

4.3.1 Include the categorical variables into a regression model

We will continue with a very similar regression model fitted in last week that relates Percentages long-term illness (*pct_Long_term_sick_or_disabled*) to Percentages no-qualification

(*pct_No_qualifications*), Percentage Males (*pct_Males*) and Percentages Higher Managerial or Professional occupation (*pct_Higher_manager_prof*).

Decide which region to be set as the baseline category. The principle is that if you want to compare the (average) long term illness outcome of Region A to those of other regions, Region A should be chosen as the baseline category. For example, if you want to compare the (average) long term illness outcome of London to rest of regions in the England and Wales, London should be selected as the baseline category.

Implement the regression model with the categorical variables - *Region* in our case. R will automatically handle the qualitative variable as dummy variables so you don't need to concern any of that. But you need to let R knows which category of your qualitative variable is your reference category or the baseline. Here we will use London as our first go. **Note:** We choose London as the baseline category so the London region will be **excluded** in the independent variable list.

Therefore, first, we set London as the reference:

```
df$Region <- fct_relevel(df$Region,  "London")
```

Similar to last week, we build our linear regression model, but also include the *Region* variable into the model.

```
model <- lm(pct_Longterm_sick_or_disabled ~ pct_Males + pct_No_qualifications + pct_Higher_ma

summary(model)
```

```
Call:
lm(formula = pct_Longterm_sick_or_disabled ~ pct_Males + pct_No_qualifications +
    pct_Higher_manager_prof + Region, data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-1.73799 -0.42623 -0.08528  0.41308  2.20676

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)
(Intercept)                  7.615375   3.018425   2.523  0.01212 *
pct_Males                   -0.167291   0.061638  -2.714  0.00701 **
pct_No_qualifications        0.251334   0.023179  10.843  < 2e-16 ***
pct_Higher_manager_prof      0.007145   0.020244   0.353  0.72438
RegionEast                  -0.760329   0.173260  -4.388 1.56e-05 ***
```

```
RegionEast Midlands            -0.254826   0.194419  -1.311  0.19090
RegionNorth East                1.164153   0.260808   4.464 1.12e-05 ***
RegionNorth West                0.784897   0.192449   4.078 5.74e-05 ***
RegionSouth East               -0.328575   0.165728  -1.983  0.04827 *
RegionSouth West                0.096484   0.214507   0.450  0.65317
RegionWales                     1.382217   0.220643   6.264 1.21e-09 ***
RegionWest Midlands            -0.415570   0.196345  -2.117  0.03508 *
RegionYorkshire and The Humber -0.118795   0.217473  -0.546  0.58528
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7086 on 318 degrees of freedom
Multiple R-squared:  0.7593,    Adjusted R-squared:  0.7502
F-statistic:  83.6 on 12 and 318 DF,  p-value: < 2.2e-16
```

You have already learnt how to interpret the output of regression model last week: **Significance** (p-value), **Coefficient Estimates, and Model fit** (R squared and Adjusted R-squared).

Q4. Relating back to this week's lecture notes, indicate what regions have statistically significant differences in the percentage of long-term illness, compared to London?

First, the **Significance** and the **Coefficient Estimates**. By examining the P-value, which is the last column in the output table, we can see that most of the independent variables are significant predictor of `pct_Longterm_sick_or_disabled`.

- Similarly to last week, we learn that the changes in `pct_No_qualifications` are significantly associated with changes in `pct_Longterm_sick_or_disabled` at the <0.001 level (with the three asterisks ***), which is actually an indicator of highly statistically significant, while we are less confident that the observed relationship between `pct_No_qualifications` and `pct_Longterm_sick_or_disabled` are statistically significant (with the two asterisks **). Through their coefficient estimates, we learn that:
  - The association of `pct_Males` is negative and mild: each decrease in 1% of `pct_Males` is associated with an increase of 0.17% of long term sick/disable rate in the population of EW.
  - The association of `pct_No_qualifications` is positive and strong: each increase in 1% of `pct_No_qualifications` is associated with an increase of 0.25% of long term sick/disable rate.
  - The association of `pct_Higher_manager_prof` is not statistically significant.

- Now comes to the dummy variables (all the items starts with Region) created by R for our qualitative variable *Region*: `RegionEast`, `RegionNorth Eest`, `RegionNorth West` and `RegionWales` are also statistically significant at the <0.001 level. The changes in `RegionSouth East` and `RegionWest Midlands` are significantly associated with changes in `pct_Longterm_sick_or_disabled` at the 0.05 level. The 0.05 level suggests that it is a mild likelihood that the relationship between these independent variables and the dependent variable is not due to random change. They are just mildly statistically significant.
- The coefficient estimates of them need to be interpreted by comparing to the reference category London. The Estimate column tells us: North East region is associated with a 1.16% higher rate of long term sick/disable than London when the other predictors remain the same. Similarly, Wales is 1.38% higher rate of long term illness than London when the other predictors remain the same. You can draw the conclusion for the other regions in this way by using their coefficient estimate values.

  Reminder: You **cannot** draw conclusion between North East and Wales, nor comparison between any regions beyond London. It is because the regression model is built for the comparison between regions to your reference category London. If we want to compare between North East and Wales, we need to set either of them as the reference category by using `df$Region <- fct_relevel(df$Region, "North East")` or `df$Region <- fct_relevel(df$Region, "Wales")`.
- `RegionEast Midland`, `RegionSouth West`, and `RegionlYorkshire and The Humber` were not found to be significantly associated with `pct_Longterm_sick_or_disabled`.

Last but not least, the **Measure of Model Fit**. The model output suggests the R-squared and Adjusted R-squared are of greater than 0.75 indicate a reasonably well fitting model. The model explains 75.9 % of the variance in the dependent variable. After adjusting for the number of independent variable, the model explains 75.0% of the variance. They two suggest a strong fit of the model.

4.3.2 Change the baseline category

If you would like to learn about differences in long-term illness between North East and other regions in the EW, you need to change the baseline category (from London) to the North East region (with variable name "Region_2").

```
df$Region <- fct_relevel(df$Region, "North West")
```

The regression model is specified again as follows:

```r
model1 <- lm(
  pct_Longterm_sick_or_disabled ~ pct_Males + pct_No_qualifications + pct_Higher_manager_pro
  data = df
)

summary(model1)
```

```
Call:
lm(formula = pct_Longterm_sick_or_disabled ~ pct_Males + pct_No_qualifications +
    pct_Higher_manager_prof + Region, data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-1.73799 -0.42623 -0.08528  0.41308  2.20676

Coefficients:
                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                     8.400272   3.042017   2.761 0.006090 **
pct_Males                      -0.167291   0.061638  -2.714 0.007009 **
pct_No_qualifications           0.251334   0.023179  10.843  < 2e-16 ***
pct_Higher_manager_prof         0.007145   0.020244   0.353 0.724376
RegionLondon                   -0.784897   0.192449  -4.078 5.74e-05 ***
RegionEast                     -1.545226   0.158845  -9.728  < 2e-16 ***
RegionEast Midlands            -1.039723   0.165225  -6.293 1.03e-09 ***
RegionNorth East                0.379256   0.235090   1.613 0.107685
RegionSouth East               -1.113472   0.152463  -7.303 2.27e-12 ***
RegionSouth West               -0.688413   0.182460  -3.773 0.000192 ***
RegionWales                     0.597320   0.189607   3.150 0.001786 **
RegionWest Midlands            -1.200467   0.174076  -6.896 2.89e-11 ***
RegionYorkshire and The Humber -0.903692   0.191943  -4.708 3.74e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7086 on 318 degrees of freedom
Multiple R-squared:  0.7593,    Adjusted R-squared:  0.7502
F-statistic:  83.6 on 12 and 318 DF,  p-value: < 2.2e-16
```

Examining R-squared to measure the model fit. How robust is the new model? What % of the variance in the dependent variable has been explained?

Now, complete the following table.

Region names	Higher or lower than North West	Whether the difference is statistically significant (Yes or No)
East Midlands		
East of England		
North East		
North West		
South East		
London		
West Midlands		
Yorkshire and The Humber		
Wales		

Now it is very easy to use your model to estimate the results Y (dependent variable - `pct_Longterm_sick_or_disabled`) by setting all the input independent variable X (`pct_Males` `pct_No_qualifications` and `pct_Higher_manager_prof`).

```
obj_London <- data.frame(
  pct_Males = 49.7,
  pct_No_qualifications = 24.3,
  pct_Higher_manager_prof = 14.7,
  Region = "London"
)
obj_WM <- data.frame(
  pct_Males = 49.8,
  pct_No_qualifications = 23.3,
  pct_Higher_manager_prof = 11.2,
  Region = "West Midlands"
)
obj_NE <- data.frame(
  pct_Males = 49.8,
  pct_No_qualifications = 23.3,
  pct_Higher_manager_prof = 11.2,
  Region = "North East"

)

predict(model1, obj_London)
```

```
       1
5.513467
```

```
predict(model1, obj_WM)
```

```
       1
4.804829
```

```
predict(model1, obj_NE)
```

```
       1
6.384552
```

4.3.3 Recode the Region variable and explore regional inequality in health

In many real-word studies, we might not be interested in health inequality across all regions. For example, in this case study, we are interested in health inequality between *London*, *South, North, Midland and Wales*. We can achieve this by re-grouping regions in the UK based on the variable "Region". That said, we need to have a new grouping of regions as follows:

Original region labels	**New region labels**
East Midlands	Midlands
East	South
London	London
North East	North
North West	North
South East	South
South West	South
West Midlands	Midlands
Yorkshire and The Humber	North
Wales	Wales

Here we use mutate() function in R to make it happen:

```
df <- df %>%
  mutate(
    New_region_label = fct_collapse(
      Region,
      North = c("North East", "North West", "Yorkshire and The Humber"),
      Midlands = c("East Midlands", "West Midlands"),
      South = c("East", "South East", "South West"),
      London = "London",
```

```
      Wales = "Wales"
    )
  )
```

This code may looks a bit complex. You can simply type ?mutate in your console. Now in your right hand Help window, the R studio offers your the explanation of the mutate function. This is a common way you can use R studio to help you learn what the function ca`ate()` creates new columns that are functions of existing variables. Therefore, the `df %>% mutate()` means add a new column into the current dataframe `df`; the `New_region_label` in the `mutate()` function indicates the name of this new column is `New_region_label`. The right side of the `New_region_label =` indicates the value we want to assign to the `New_region_label` in each row.

The right side of `New_region_label` is

`fct_collapse(Region, North = c("North East", "North West", "Yorkshire and The Humber"), Midlands = c("East Midlands", "West Midlands"), South = c("East", "South East", "South West"), London = "London", Wales = "Wales")`

By using the code, the `fct_collapse()` function recodes or group each value in the `Region` column into one of the broader categories: "London", "South", "North","Midlands" or "Wales". Specifically, it takes the existing factor levels in `Region` and collapses multiple detailed regions into fewer, aggregated groups. For example, regions such as *"North East"*, *"North West"*, and *"Yorkshire and The Humber"* are all grouped under **"North"**, while *"East Midlands"* and *"West Midlands"* are combined into **"Midlands"**, and so on.

Now we use the same way to examine our new column `New_region_label`:

```
table(df$New_region_label)
```

```
North  London   South Midlands   Wales
   72      33     139      65      22
```

Comparing with the `Region_label`, we now can see the mutate worked:

```
df[,c("Region","New_region_label")]
```

Now you will have a new qualitative variable named `New_region_label` in which the UK is divided into four regions: London, South, North and Midlands.

Based on the newly generated qualitative variable `New_region_label`, we can now build our new linear regression model. Don't forget:

(1) R need to deal with the categorical variables in regression model in the factor type;

```
class(df$New_region_label)
```

```
[1] "factor"
```

The `class()` returns the type of the variable. The `New_region_label` is already a factor variable. If not, we need to convert it by the `as.factor()`, as we used above.

```
df$New_region_label = as.factor(df$New_region_label)
```

2) Let R know which region you want to use as the baseline category. Here I will use London again, but of course you can choose other regions.

```
df$New_region_label <- fct_relevel(df$New_region_label, "London")
```

The linear regression window is set up below. This time we include `New_region_label` rather than `Region_label` as the region variable:

```
model2 <- lm(
  pct_Longterm_sick_or_disabled ~ pct_Males + pct_No_qualifications + pct_Higher_manager_pro
  data = df
)

summary(model2)
```

```
Call:
lm(formula = pct_Longterm_sick_or_disabled ~ pct_Males + pct_No_qualifications +
    pct_Higher_manager_prof + New_region_label, data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-1.83271 -0.48362 -0.06293  0.40581  2.11961

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)               9.98363    3.20535   3.115  0.00201 **
pct_Males                -0.18418    0.06585  -2.797  0.00547 **
pct_No_qualifications     0.20143    0.02257   8.924  < 2e-16 ***
pct_Higher_manager_prof  -0.03410    0.01986  -1.718  0.08684 .
New_region_labelNorth     0.47416    0.18706   2.535  0.01172 *
```

```
New_region_labelSouth    -0.50635    0.16433  -3.081  0.00224 **
New_region_labelMidlands -0.41031    0.18517  -2.216  0.02740 *
New_region_labelWales     1.25183    0.23431   5.343 1.73e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.76 on 323 degrees of freedom
Multiple R-squared:  0.7188,    Adjusted R-squared:  0.7127
F-statistic:   118 on 7 and 323 DF,  p-value: < 2.2e-16
```

Q5. Are there statistically significant differences in the percentage of people with long-term sick/disable between London and North, and between South and Wales, controlling for other variables? What conclusions could be drawn in terms of regional differences in health outcome?

4.4 Predictions using fitted regression model

Write down the % long term sick or disabled regression model with the new region label categorical variables.

Relating to this week's lecture, the % `pct_Longterm_sick_or_disabled` is equal to:

[**write down the model**]

Q6. Now imagine that the values of variables *pct_Males, pct_No_qualifications, and pct_Higher_manager_prof* are 49, 23 and 11, respectively, what would the percentage of persons with long-term sick/disable in Wales and London be?

Check the answer at the end of this practical page.

4.5 Extension activities

The extension activities are designed to get yourself prepared for the Assignment 1 in progress. For this week, try whether you can:

- Present descriptive statistics for independent variable and the dependent variable: counts, percentages, a centrality measure, a spread measure, histograms or any relevant statistic
- Report the observed association between the dependent and independent variables: correlation plus a graphic or tabular visualisation
- Briefly describe and critically discuss the results

- Think about other potential factors of long-term illness and income, and then test your ideas with linear regression models
- Summaries your model outputs and interpret the results.

4.6 Answer of the written down model and Q6

The model of the new region label is: *pct_Longterm_sick_or_disabled* (%) = *9.98+ (-0.1842)* pct_Males* (%) *+ 0.2014 * pct_No_qualifications* (%) *+ (-0.0341) * pct_Higher_manager_prof + (0.4742) * North + (-0.5063)* * South + (-0.4103) * Midlands + *1.2518 * Wales.* But should be aware that the relation between *pct_Higher_manager_prof* and *pct_Longterm_sick_or_disabled* is not statistically significant.

So if the values of variables *pct_Males, pct_No_qualifications, and pct_Higher_manager_prof* are 49, 23 and 11,

the model of Wales will be: *pct_Longterm_sick_or_disabled* (%) = *9.98+ (-0.1842)* 49 + 0.2014 * 23 + (-0.0341) * 11 + (0.4742) * 0+ (-0.5063)* * 0+ (-0.4103) * 0+ *1.2518 * 1*(you can direct paste the number sentence into your R studio Console and the result will be returned)

the model of London will be: *pct_Long_term_ill* (%) = *9.98+ (-0.1842)* 49 + 0.2014 * 23 + (-0.0341) * 11 + (0.4742) * 0+ (-0.5063)* * 0+ (-0.4103) * 0+ *1.2518 * 0*

You can also make a new object like

```
obj_Wales <- data.frame(
  pct_Males = 49,
  pct_No_qualifications = 23,
  pct_Higher_manager_prof = 11,
  New_region_label = "Wales"
)

obj_London <- data.frame(
  pct_Males = 49,
  pct_No_qualifications = 23,
  pct_Higher_manager_prof = 11,
  New_region_label = "London"
)

predict(model2, obj_Wales)
```

```
       1
6.468668
```

```
predict(model2, obj_London)
```

```
       1
5.216834
```

Therefore, the percentage of persons with long-term illness in Wales and London be 6.47% and 5.22 % and separately. If you got the right answers, then congratulations you can now use regression model to make prediction.

5 Lab: Logistic Regression

Last week, we learnt how to use qualitative variables in multiple linear regression model to understand the relationship between independent variables X1 … Xn and the dependent variable Y. Today we will learn to use logistic regression for binary dependent variable. A **logistic regression model** is a type of regression analysis used when the dependent variable is binary (e.g., "success/failure" or "0/1"). It estimates the probability of one outcome relative to the other using a logistic function. This model is commonly used in situations like predicting disease presence (yes/no) or customer churn (stay/leave). The independent variables can be continuous, categorical, or a mix of both. The model's output is in the form of odds ratios, showing how predictors affect the likelihood of the outcome.

The lecture's slides can be found here.

Learning objectives

An understanding of, and ability to:

- Estimate and interpret a logistic regression model
- Assess the model fit

The application context of a binomial logistic regression model is when the dependent variable under investigation is a binary variable. Usually, a value of 1 for this dependent variable means the occurrence of an event; and, 0 otherwise. For example, the dependent variable for this practical is whether a person is a long-distance commuter i.e. 1, and 0 otherwise.

In this week's practical, we are going to apply logistic regression analysis in an attempt to answer the following research question:

RESEARCH QUESTION: Who is willing to commute long distances?

The practical is split into two main parts. The first focuses on implementing a binary logistic regression model with R. The second part focuses the interpretation of the resulting estimates.

5.1 Knowing the dataset and descriptive analysis

Prepare the data for implementing a logistic regression model. The data set used in this practical is the "sar_sample_label.csv" and "sar_sample_code.csv". The SAR is a snapshot of census microdata, which are individual level data. The data sample has been drawn and anonymised from census and known as the Samples of Anonymised Records (SARs).

You may need to download the two datasets and also the data dictionary from Canvas if you haven't. Double click to open both .csv files you may find they have exactly same column names - yes, they are in fact the same spreadsheet but 'sar_sample_label.csv' is more readable for the cell contents as they are in text format but cells in the 'sar_sample_code.csv' is in coding format. The 'SAR_dictionary' is create to help you understand what the columns mean, just like the FRS dictionary for the FRS datasets last week. Yes, the two SAR csv files are actually the same dataframe, only one uses the label as the value but the other uses the code. This is quite normal as when we doing surveys, we use text as the options of multiple choice questions for respondents to choose from, and therefore the collected survey results are in the 'sar_sample_label' format. The label format is usually more readable and is good to do descriptive analysis as we will show in today's content. However, we may find the coding format is more concise for writing the code and thus will be used during regression analysis. **Please notice that coding the labels into numbers, doesn't mean the categorical/qualitative variable has been converted into numeric/continuous numbers.**

Let's first read in both for the data overview.

```
if(!require("tidyverse"))
  install.packages("tidyverse",dependencies = T, repos = "https://cloud.r-project.org/")
```

```
Warning: package 'ggplot2' was built under R version 4.4.3
```

```
if(!require("broom"))
  install.packages("broom",dependencies = T, repos = "https://cloud.r-project.org/")
if(!require("forcats"))
  install.packages("forcats")

library(tidyverse)
library(broom)
library(forcats)
```

```
sar_label <- read.csv("../data/SAR/sar_sample_label.csv")
sar_code <- read.csv("../data/SAR/sar_sample_code.csv")
```

Run the following codes to view both dataframes:

```
#View the sar_label dataset
View(sar_label)
```

```
#View the sar_code dataset
View(sar_code)
```

To know the dataset better, we can run dim() to know who many individuals (rows) in this sample data and how many variables (columns) have been recorded.

```
dim(sar_label)
```

```
[1] 50000    39
```

```
#or
dim(sar_code)
```

```
[1] 50000    39
```

So there are 50,000 respondents in the sample dataset and 39 variables are included.

After browsing both data sheets and also the data dictionary (sometimes we call it meta data), you may notice that all the columns in the SAR dataset is categorical/qualitative type. Therefore, if we run the `summary()` function for the dataframes, as we did in previous weeks for the Census data or the FRS dataset, the outputs of the `summary()` may not be very useful. Again, please notice that the figures in the sar_code.csv is not mean actual numbers, but the codes of different categories of the qualitative variables. Therefore, the `summary()` results at there is not useful. Please reflect what we've learnt in last week's, when the variables if categorical/qualitative type, how we do the descriptive analysis for them?

```
summary(sar_code)
summary(sar_label)
```

So, the answer is we focus on summarising the frequency and distribution of categories within the variable and the function in R for doing this is `table()`. For example, the variable "work_distance" captures a person's commuting distance. We can run the table() for both label and code csv to understand the distribution, composition and diversity of categories of the ''work_distance" in the data. Try to repeat what we have learnt in Week 3 to finish the **descriptive analysis** for the categorical variables in SAR (by ggplot2).

```
table(sar_label$work_distance)
```

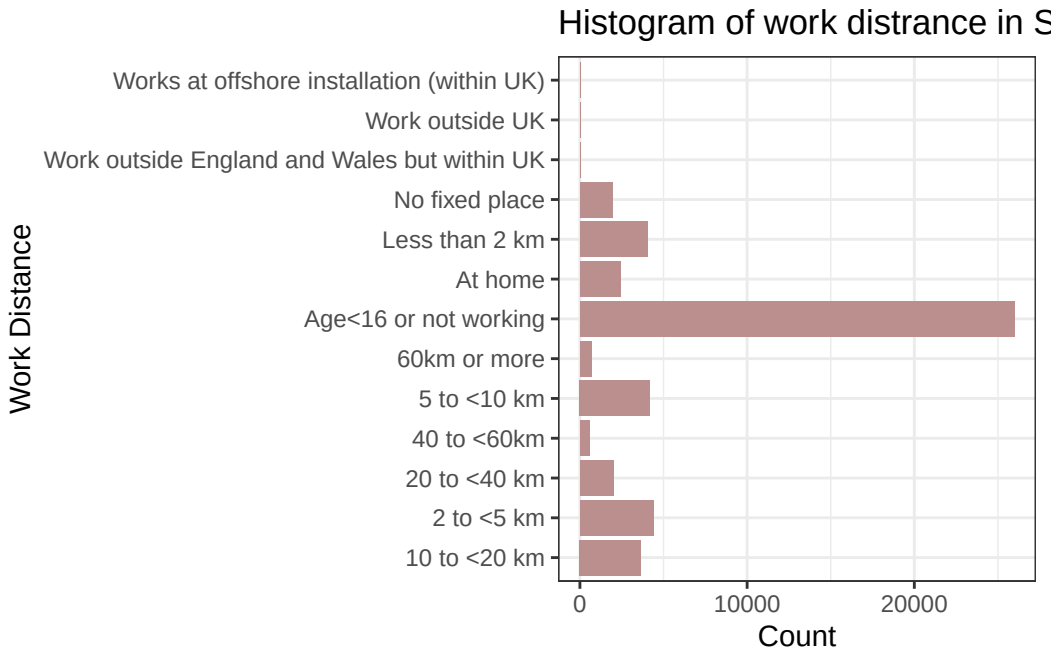
```
                                    10 to <20 km
                                            3650
                                     2 to <5 km
                                            4414
                                    20 to <40 km
                                            2014
                                     40 to <60km
                                             572
                                     5 to <10 km
                                            4190
                                    60km or more
                                             703
                             Age<16 or not working
                                           25975
                                         At home
                                            2427
                                  Less than 2 km
                                            4028
                                  No fixed place
                                            1943
Work outside England and Wales but within UK
                                              29
                                Work outside UK
                                              21
  Works at offshore installation (within UK)
                                              34
```

```
table(sar_code$work_distance)
```

```
   -9     1     2     3     4     5     6     7     8     9    10    11    12
25975  4028  4414  4190  3650  2014   572   703  2427  1943    29    21    34
```

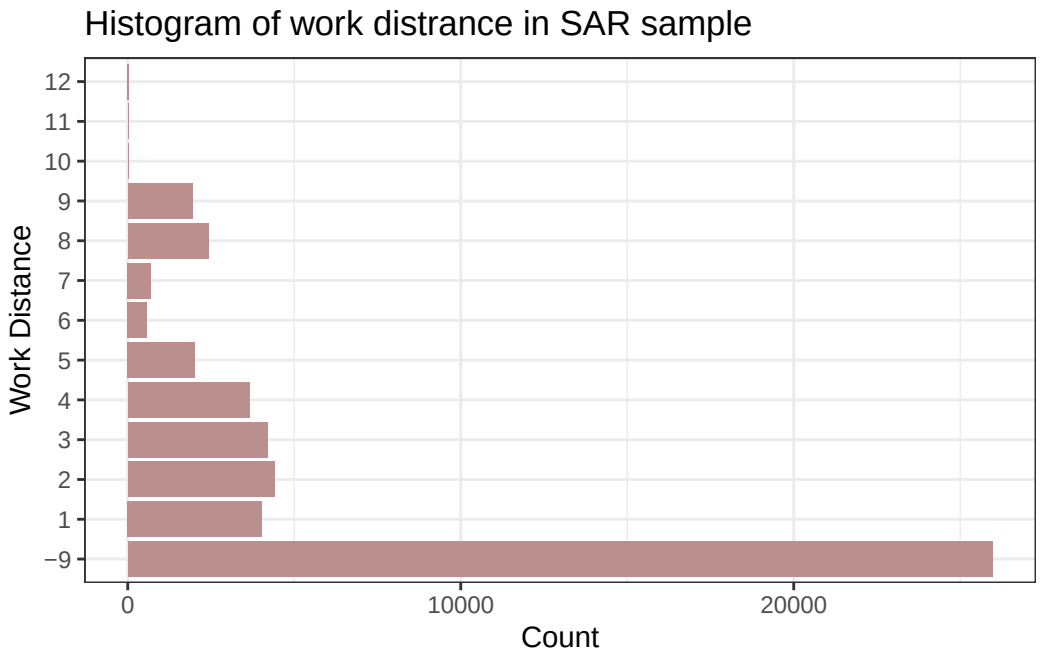
To create a chart as part of the descriptive analysis, exactly as what we did in Week 3, we use the library ggplots and the codes:

```
library(ggplot2)
ggplot(sar_label, aes(x = work_distance)) +
  geom_bar(fill = "rosybrown") +
  labs(title = "Histogram of work distrance in SAR sample", x = "Work Distance", y = "Count")
  coord_flip()+ #Flip the Axes, add a # in front of this line, to make the code in gray and y
  theme_bw()
```



We have learnt that for the categorical/qualitative variables, we need R to treat them as `factor` type. Therefore, if we want to create descriptive analysis charts by using `sar_code` , we need first convert the code numbers from numeric to factor type.

```
library(ggplot2)
ggplot(sar_code, aes(x = as.factor(work_distance))) +
  geom_bar(fill = "rosybrown") +
  labs(title = "Histogram of work distrance in SAR sample", x = "Work Distance", y = "Count")
  coord_flip()+ #Flip the Axes, add a # in front of this line, to make the code in gray and y
  theme_bw()
```



You may also realise that this chart is not that readable to be used as a descriptive analysis chart to put in the report as the Y-axis are all in codes, therefore the first chart created by labels is a better choice. Based on 'SAR_dictionary.xlsx', we learn how the codes map to the categories for the variable work_distance in both csv.

Code for Work_distance	Categories
-9	Age<16 or not working
1	Less than 2 km
2	2 to <5 km
3	5 to <10 km
4	10 to <20 km
5	20 to <40 km
6	40 to <60 km
7	60km or more
8	At home
9	No fixed place
10	Work outside England and Wales but within UK
11	Work outside UK
12	Works at offshore installation (within UK)

There are a variety of categories in the variable, however, we are only interested in commuting distance and therefore in people reporting their commuting distance. **Thus, we will explore**

the numeric codes of the variable ranging from 1 to 8.

As we are also interested in exploring whether people with different socio-economic statuses (or occupations) tend to be associated with varying probabilities of commuting over long distances, we further filter or select cases.

```
table(sar_label$nssec)
```

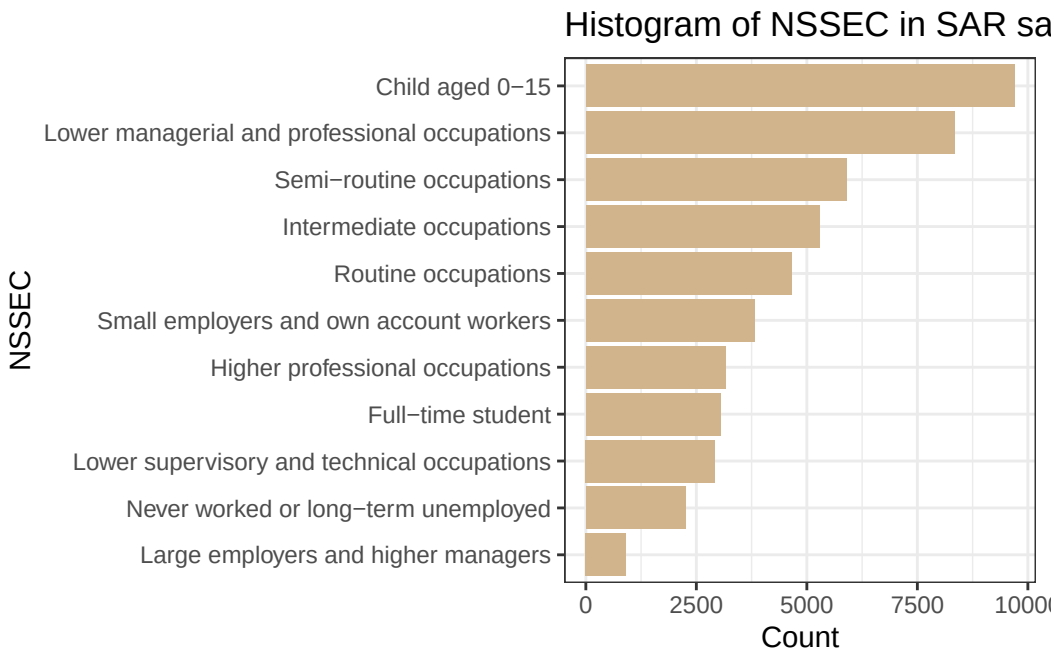
```
                              Child aged 0-15
                                         9698
                             Full-time student
                                         3041
                Higher professional occupations
                                         3162
                       Intermediate occupations
                                         5288
           Large employers and higher managers
                                          909
Lower managerial and professional occupations
                                         8345
  Lower supervisory and technical occupations
                                         2924
          Never worked or long-term unemployed
                                         2261
                           Routine occupations
                                         4660
                      Semi-routine occupations
                                         5893
      Small employers and own account workers
                                         3819
```

```
table(sar_code$nssec)
```

```
   1    2    3    4    5    6    7    8    9   10   12
 909 3162 8345 5288 3819 2924 5893 4660 2261 3041 9698
```

```
ggplot(sar_label, aes(x = fct_rev(fct_infreq(nssec)))) +
  geom_bar(fill = "tan") +
  labs(title = "Histogram of NSSEC in SAR sample", x = "NSSEC", y = "Count") +
```

```
  coord_flip()+ #Flip the Axes, add a # in front of this line, to make the code in gray and y
  theme_bw()
```



From 'SAR_dictionary.xlsx', we learn what each code indicate corresponding category for variable `nssec`. For the following regression model, we select people who reported an occupation, and delete cases with numeric codes from 9 to 12, who are *unemployed*, *full-time students*, *children* and *not classifiable*.

Code for nssec	Category labels
1	Large employers and higher managers
2	Higher professional occupations
3	Lower managerial and professional occupations
4	Intermediate occupations
5	Small employers and own account workers
6	Lower supervisory and technical occupations
7	Semi-routine occupations
8	Routine occupations
9	Never worked or long-term employed
10	Full-time student
11	Not classifiable
12	Child aged 0-15

Now, similar to next week, we use the `filter()` to prepare our dataframe today. You may already realise that using `sar_code` is easier to do the filtering.

```
sar_df <- sar_code %>% filter(work_distance<=8 & nssec <=8 )
```

Q1. Create descriptive analysis for the two variables "work_distance", "nssec" and "sex" with the new data, including (1) summarise the frequencies and (2) create histogram charts for the three variables.

5.2 Preparing the input variables

For a logistic regression model, we first need to **recode the "work_distance" variable into a binary dependent variable** as our independent variable.

A simple way to create a binary dependent variable representing long-distance commuting is to use the mutate() function as discussed in last week's practical session. Before creating the binary variables from the "work_distance" variable, we need to define *what counts as a long-distance commuting move*. Such definition can vary. Here we define a long-distance commuting move as any commuting move over a distance above 60km (the category of "60km or more").

```
sar_df <- sar_df %>% mutate(
  New_work_distance = if_else(work_distance >6, 1,0))
```

Q2. Do descripitve analysis to the new `sar_df` dataframe with new column named `New_work_distance` by using the codes you have learnt.

The independent variables are gender and socio-economic status. The gender variable `sex` is a categorical variable. Therefore, as we've learnt last week, before adding the categorical variables into the regression model, we need first make it a factor and then identify the reference category. For gender, we use male as the basline. Prepare your "sex" variable before the regression model:

```
sar_df$sex <- relevel(as.factor(sar_df$sex),ref="1")
```

Then, we prepare socio-economic status variable `nssec` for the regression model. We are interested in whether people with occupations being "Higher professional occupations" are associated with a lower probability of commuting over long distances when comparing to people in other occupations. Therefore by using the "Higher professional occupations" code 2, we run the code as below:

```r
sar_df$nssec <- relevel(as.factor(sar_df$nssec), ref = "2")
```

5.3 Implementing a logistic regression model

The binary dependent variable is long-distance commuting, variable name `New_work_distance`.

```r
#create the model
m.glm = glm(New_work_distance~sex + nssec,
            data = sar_df,
            family= "binomial")
# inspect the results
summary(m.glm)
```

```
Call:
glm(formula = New_work_distance ~ sex + nssec, family = "binomial",
    data = sar_df)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.67337    0.05329 -31.401  < 2e-16 ***
sex2        -0.36678    0.04196  -8.742  < 2e-16 ***
nssec1      -0.12881    0.11306  -1.139    0.255
nssec3      -0.38761    0.06467  -5.994 2.05e-09 ***
nssec4      -1.03079    0.08439 -12.214  < 2e-16 ***
nssec5       1.22639    0.06489  18.898  < 2e-16 ***
nssec6      -1.38992    0.10919 -12.730  < 2e-16 ***
nssec7      -1.43909    0.09002 -15.986  < 2e-16 ***
nssec8      -1.48534    0.09646 -15.398  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 20441  on 33025  degrees of freedom
Residual deviance: 17968  on 33017  degrees of freedom
AIC: 17986

Number of Fisher Scoring iterations: 6
```

```r
# odds ratios
exp(coef(m.glm))
```

```
(Intercept)        sex2      nssec1      nssec3      nssec4      nssec5
  0.1876138   0.6929649   0.8791416   0.6786766   0.3567267   3.4088847
     nssec6      nssec7      nssec8
  0.2490946   0.2371432   0.2264258
```

```r
# confidence intervals
exp(confint(m.glm, level = 0.95))
```

```
Waiting for profiling to be done...
```

```
                2.5 %    97.5 %
(Intercept) 0.1688060 0.2080319
sex2        0.6381810 0.7522773
nssec1      0.7017990 1.0935602
nssec3      0.5981911 0.7708192
nssec4      0.3020431 0.4205270
nssec5      3.0037298 3.8739884
nssec6      0.2002766 0.3073830
nssec7      0.1984396 0.2824629
nssec8      0.1869397 0.2729172
```

5.3.1 Interpreting estimated regression coefficients

Compare to the statistical inference in a multiple linear regression model context (you have done in Week 2 and 3), the interpretation of coefficients (B) and odds ratios (Exp(B)) for the logistic regression model have some similarity and differences:

It is the same that we read p-values of regression coefficients to assess significances of coefficients; for instance, by comparing p-values to the conventional level of significance of 0.05:

· If the p-value of a coefficient is smaller than 0.05, the coefficient is statistically significant. In this case, you can say that the relationship between an independent variable and the outcome variable is *statistically* significant.

· If the p-value of a coefficient is larger than 0.05, the coefficient is statistically insignificant. In this case, you can say or conclude that there is no statistically significant association or relationship between an independent variable and the outcome variable.

It is different in the way we interpret the regression coefficients (B) as we need to read the odds ratios (Exp(B)) from `exp(coef(m.glm))`

o For the variable **`sex`**, a negative sign and the odds ratio estimate indicate that the probability of commuting over long distances for female is 0.693 times less likely than male (the reference group), with the confidence intervals (CI) or likely range between 0.63 to 0.75, holding all other variables constant (the socio-economic classification variable). Put it differently, being females reduces the probability of long-distance commuting by 30.7% (1-0.693).

o For variable **`nssec`**, a positive significant and the odds ratio estimate indicate that the probability of long-distance commuting for those whose socio-economic classification as:

- the p-value of Large employers and higher managers (nssec=1) is > 0.05, so there is no statistically significant relationship between large employers and higher managers and long-distance commuting;
- Lower managerial and professional occupations (nssec=3) are 0.679 times likely for long-distance commuting when comparing to our reference category (nssec=2, higher prof occupations), holding all other variables constant (the Sex variable), with a likely range (CI) between 0.60 to 0.77. Thefore we can also say that the workers in lower managerial and professional occupations has 32.1% (1-0.679) less probability than the higher professional workers to travel longer than 60km for work when the gender is the same.
- Small employers and own account workers (nssec=5) are 3.409 times more likely than the reference category (nssec=2, higher prof occupations), when holding the gender variable the same, with a likely range (CI) of between 3.00 to 3.87.
- Compare to the higher professional occupations (nssec=2), the worker in Routine occupations (nssec=8) are 0.226 times (or 22.6%) likely to travel more than 60km to work, with the CI between 0.19 to 0.27. when other variable constant. Or, we can see being routine occupations decreases the probability of long-distance commuting by 77.4% (1-0.226).

Q3. Can you write down the findings you learnt from the model outcomes to other occupation groups (nssec = 4, nssec = 6 and nssec =7)?

5.3.2 Model fit

We include the R library `pscl` for calculate the measures of fit.

```
if(!require("pscl"))
  install.packages("pscl", repos = "https://cloud.r-project.org/")
```

```
Loading required package: pscl
```

```
Warning: package 'pscl' was built under R version 4.4.2
```

```
Classes and Methods for R originally developed in the
Political Science Computational Laboratory
Department of Political Science
Stanford University (2002-2015),
by and under the direction of Simon Jackman.
hurdle and zeroinfl functions by Achim Zeileis.
```

```
library(pscl)
```

Relating back to this week's lecture notes, what is the Pseudo R^2 of the fitted logistic model (from the Model Summary table below)?

```
# Pseudo R-squared
pR2(m.glm)
```

```
fitting null model for pseudo-r2
```

```
          llh       llhNull            G2      McFadden          r2ML
-8.983928e+03 -1.022037e+04  2.472890e+03  1.209785e-01  7.214246e-02
         r2CU
 1.563288e-01
```

```
# or in better format
pR2(m.glm) %>% round(4) %>% tidy()
```

```
fitting null model for pseudo-r2
```

```
# A tibble: 6 x 2
  names              x
  <chr>          <dbl>
1 llh       -8984.
2 llhNull  -10220.
3 G2         2473.
4 McFadden      0.121
5 r2ML          0.0721
6 r2CU          0.156
```

- **llh**: The log-likelihood of the fitted model.

- **llhNull**: The log-likelihood of the null model (without predictors).
- **G2**: The likelihood ratio statistic, showing the model's improvement over the null model.
- **McFadden**: McFadden's pseudo R-squared (a common measure of model fit).
- **r2ML**: Maximum likelihood pseudo R-squared.
- **r2CU**: Cox & Snell pseudo R-squared.

Different from the multiple linear regression, whose R-squared indicates % of the variance in the dependent variables that is explained by the independent variable. In logistic regression model, R-squared is not directly applicable. Instead, we use pseudo R-squared measures, such as McFadden's pseudo R-squared, or Cox & Snell pseudo R-squared to provide an indication of model fit. For the individual level dataset like SAR, value around 0.3 is considered good for well-fitting. Therefore, our model is not that robust for prediction but still explain the association between variables and the categories. To improve the model, we may need more variables as the independent variables, you may identify some based on the related literature or debates on this topic.

5.3.3 Recode Socio-economic status variable and explore commuting differences

This time, we want to know whether "Lower supervisory and technical occupations", "Semi-routine occupations" and "Routine occupations" are associated with higher probability of commuting over long distance when comparing to people in other occupation.

We can use mutate() to create a new column, set the value of "Lower supervisory and technical occupations", "Semi-routine occupations" and "Routine occupations" as original, while the rest as "Other occupations". Here by using the SAR in code format, we can make this more easier by using:

```
sar_df <- sar_df %>% mutate(New_nssec = fct_other(nssec,   keep = c("6", "7", "8"),   other_l
```

Or by using `if_else` and `%in%` in R, we can achieve the same result. `%in%` is an operator used to test if elements of one vector are present in another. It returns `TRUE` for elements found and `FALSE` otherwise.

```
sar_df <- sar_df %>% mutate(New_nssec = if_else(!nssec %in% c(6,7,8), "0" ,nssec))
```

Use "Other occupations" (code: 0) as the reference category by `relevel(as.factor())` and then create the regression model: `glm(New_work_distance~sex + New_nssec, data = sar_df, family= "binomial")`. Can you now run the model by yourself? Find the answer at the end of the practical.

Q4. If we want to explore whether people as independent employers show lower probability of commuting longer than 60km compared with other occupations, how will we prepare the input independent variables and what will be the specified regression model?

5.3.4 Prediction using fitted regression model

Relating to this week's lecture, the log odds of the person who is will to long-distance commuting is equal to:

Log odds of long-distance commuting = 0.188 + 0.693 * sexFemale + 0.679 * nssec3 + 0.357*nssec4 + 3.409*nssec5 + 0.249*nssec6 + 0.237*nssec7 + 0.226*nssec8. We don't include nssec1 as the previous result shows that it is not statistically significant, so we won't use the model to predict individuals that as nssec=1 occupation.

By using R, you can create the object you would like to predict. Here we created three person, see whether you can interpret their gender and socio-economic classification?

```
objs <- data.frame(sex=c("1","2","1"),nssec=c("7","3","5"))
```

Then we can predict by using our model `m.glm`:

```
predict(m.glm, objs,type = "response")
```

```
         1          2          3
0.04259618 0.08108050 0.39007797
```

So let us look at these three people. The first one, for a male who classified as Semi-routine occupation in NSSEC, the probability of he travel over 60km to work is only 4.26%. For the second one, a female who is in Lower managerial and professional occupation, the probability of long-distance commuting is 8.11%. Now you know the prediction outcomes for our last person. Remind: the model fitting result shows the model is not very robust, therefore the prediction may not very solid as well.

5.4 Extension activities

The extension activities are designed to get yourself prepared for the Assignment 2 in progress. For this week, try whether you can:

- Select a regression strategy and explain why a linear or logistic model is appropriate
- Perform one or a series of regression models, including different combinations of your chosen independent variables to explain and/or predict your dependent variable

5.5 Answers for Qs

Answer for Q1

For Q1, we have already had the codes for `work_distance` and `nssec`, so the only missing variable is `sex`:

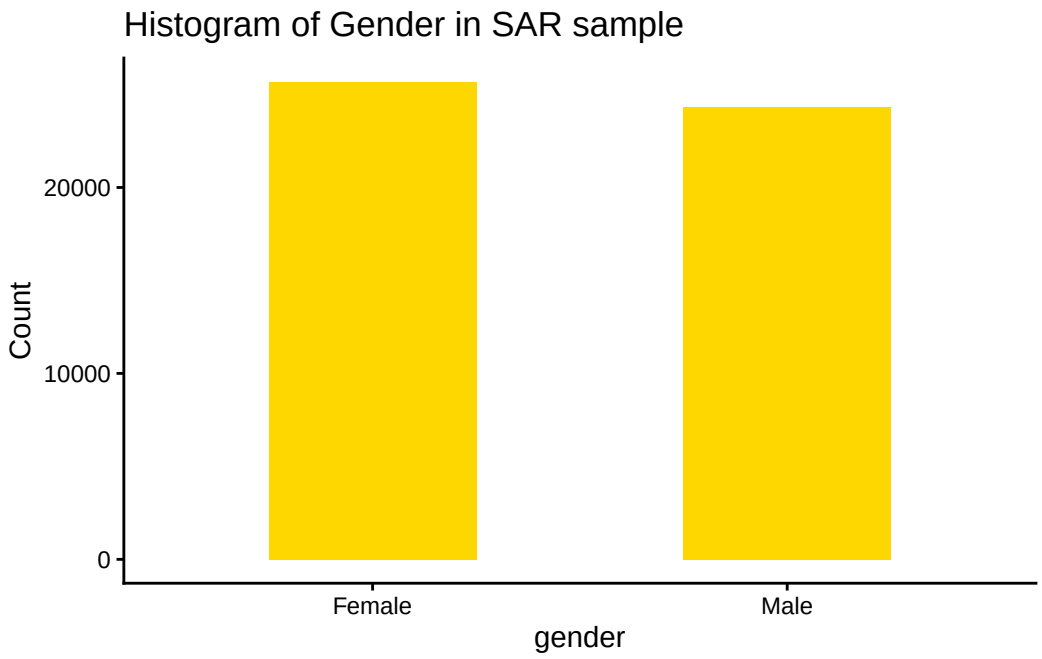
To summaries the frequency of each category, we use code:

```
table(sar_label$sex)
```

```
Female   Male
 25677  24323
```

To create bar charts for the distribution and composition of the gender variable in the SAR sample, we can run:

```
ggplot(sar_label, aes(x = sex)) +
  geom_bar(fill="gold",width=0.5) +
  labs(title = "Histogram of Gender in SAR sample", x = "gender", y = "Count")+#set text inf
  theme_classic()#choose theme type, try theme_bw(), theme_minimal() see differences
```



Answer for Q2

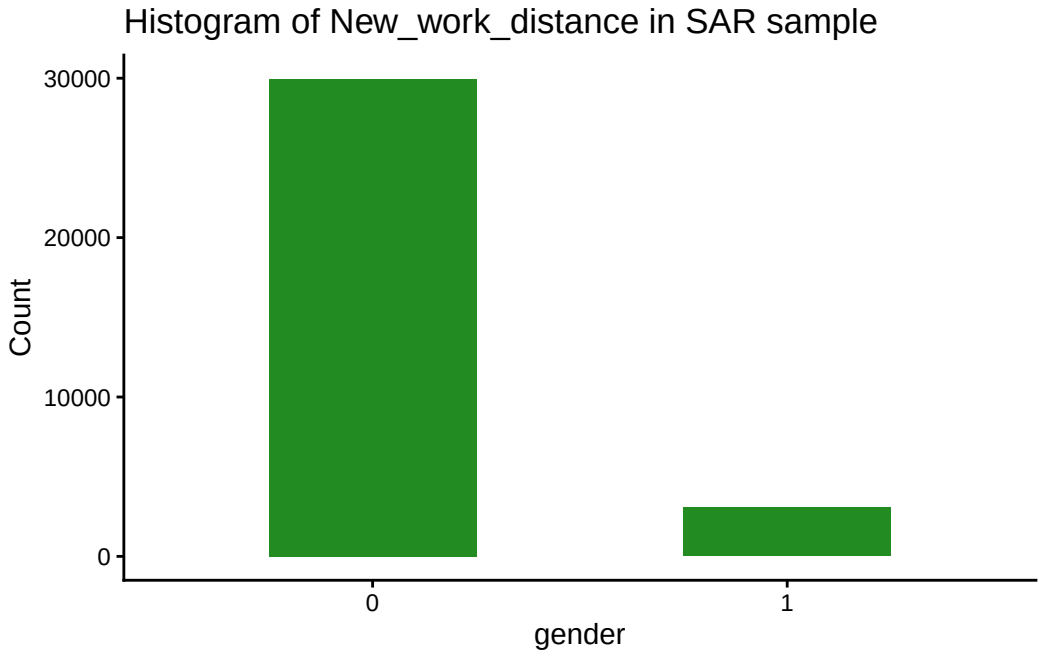
Similarly, the descriptive analysis for the newly created variable `New_work_distance` in `sar_df` should include:

```
table(sar_df$New_work_distance)
```

```
    0     1
29954  3072
```

Here we see the categories of `New_work_distance` are in numeric type 0 or 1 (as we used code `sar_df <- sar_df %>% mutate( New_work_distance = if_else(work_distance >6, 1,0))` to do so), therefore in the histogram code, we use as.factor() to convert it as factor type.

```
ggplot(sar_df, aes(x = as.factor(New_work_distance))) +
  geom_bar(fill="forestgreen",width=0.5) +
  labs(title = "Histogram of New_work_distance in SAR sample", x = "gender", y = "Count")+#s
  theme_classic()#choose theme type, try theme_bw(), theme_minimal() see differences
```



Now we can use the frequency and the bar chart to report that according to our definition of long-distance travelling to work (over 60 km), there are 3072 individual are long-distance commuters in the SAR sample, which makes up 6.14% (3072/50000) of the sample.

Answer for the model in Q4

In Q4, we we want to explore whether people with occupation being independent employers are associated with higher probability of commuting over long distance when comparing to people in other occupation. So we set the nssec= 5 (Small employers and own account workers) as the baseline category.

```
table(sar_df$New_nssec)
```

```
    0     6     7     8
20063  2790  5704  4469
```

Then we set the reference categories: `sex` as `1` (male) and `New_nssec` as 5, which is 'Small employers and own account workers':

```
sar_df$sex <- relevel(as.factor(sar_df$sex),ref="1")
sar_df$nssec <- relevel(as.factor(sar_df$nssec),ref="5")
```

Now, we build the logistic regression model and check out the outcomes:

```
model_new = glm(New_work_distance~sex + nssec, data = sar_df, family= "binomial")

summary(model_new)
```

```
Call:
glm(formula = New_work_distance ~ sex + nssec, family = "binomial",
    data = sar_df)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.44698    0.04138 -10.801   <2e-16 ***
sex2        -0.36678    0.04196  -8.742   <2e-16 ***
nssec2      -1.22639    0.06489 -18.898   <2e-16 ***
nssec1      -1.35519    0.10782 -12.569   <2e-16 ***
nssec3      -1.61400    0.05482 -29.442   <2e-16 ***
nssec4      -2.25717    0.07696 -29.329   <2e-16 ***
```

```
nssec6      -2.61631    0.10377 -25.212   <2e-16 ***
nssec7      -2.66548    0.08317 -32.047   <2e-16 ***
nssec8      -2.71172    0.09021 -30.059   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 20441  on 33025  degrees of freedom
Residual deviance: 17968  on 33017  degrees of freedom
AIC: 17986

Number of Fisher Scoring iterations: 6
```

For the model interpretation, we need:

```
# odds ratios
exp(coef(model_new))
```

```
(Intercept)        sex2      nssec2      nssec1      nssec3      nssec4
 0.63955383  0.69296493  0.29335107  0.25789714  0.19909052  0.10464615
     nssec6      nssec7      nssec8
 0.07307217  0.06956621  0.06642224
```

```
# confidence intervals
exp(confint(model_new, level = 0.95))
```

```
Waiting for profiling to be done...

                2.5 %     97.5 %
(Intercept) 0.58959085 0.69343772
sex2        0.63818099 0.75227729
nssec2      0.25813190 0.33291942
nssec1      0.20787318 0.31733101
nssec3      0.17876868 0.22162844
nssec4      0.08984430 0.12149367
nssec6      0.05933796 0.08915692
nssec7      0.05895858 0.08169756
nssec8      0.05547661 0.07902917
```

And don't forget measure the model fit:

```
# model fit
pR2(model_new) %>% round(4) %>% tidy()
```

```
fitting null model for pseudo-r2
```

```
Warning in tidy.numeric(.): 'tidy.numeric' is deprecated.
See help("Deprecated")
```

```
# A tibble: 6 x 2
  names              x
  <chr>          <dbl>
1 llh       -8984.
2 llhNull   -10220.
3 G2         2473.
4 McFadden      0.121
5 r2ML          0.0721
6 r2CU          0.156
```

Q5. Now what conclusion you can draw when comparing the median NSSEC groups to the higher groups?

6 Lab: Data Visualisation with ggplot

The lecture's slides can be found here.

6.1 Part 1: Towards the Assignment (30 min, or till when you feel you are good to go)

1. Identify a possible topic:

- Load your dataset(s) in Rstudio.
- Define your Research Question.
- Identify the related variables to analyse.

2. Discuss ideas:

- Get feedback on feasibility and the clarity of research questions.

6.2 Part 2 - Visualisation: `ggplot2` Functions and Arguments

For this session you need the following libraries:

```
install.packages(c("ggplot2", "dplyr", "tidyr", "kableExtra", "ggridges",
"RColorBrewer", "broom", "scales", "corrplot", "vtable"))
```

If necessary install them by moving the line into a code chunk.

The **`ggplot2`** package in R is one of the most powerful tools for creating publication-quality visualisations. It uses a layered approach to building plots, starting with data, then adding mappings, geometries, and other components.

A `ggplot2` plot is built step by step:

```
ggplot(data, aes(x = <X-axis variable>, y = <Y-axis variable>, <other aesthetics>)) +
 <geom_function()> +
 <scales/themes/other layers>
```

6.2.1 `ggplot()`

- Initializes the plotting system.
- Main arguments:
- `data`: A data frame containing the variables to be plotted.
- `aes()`: Aesthetic mappings to connect data variables to visual properties like `x`, `y`, `color`, `fill`, `size`, etc.

```
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 4.4.3
```

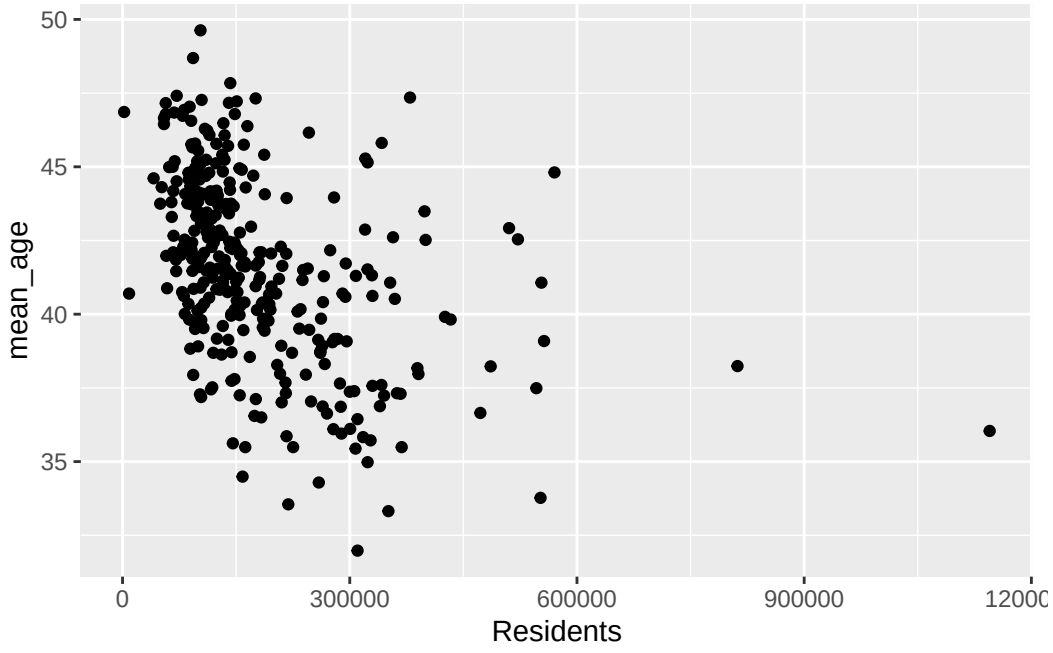
Example:

Let's load the data first:

```
census_data <- read.csv("../data/Census2021/EW_DistrictPercentages.csv")
```

Plotting Number of residents vs age (mean).

```
ggplot(data = census_data, aes(x = Residents, y = mean_age)) +
 geom_point()
```



6.2.2 Geometries in ggplot2

You can plot the following geometries in ggplot:

Geometry	Function	Use Case
Point	`geom_point()`	Scatterplots
Line	`geom_line()`	Line plots
Bar	`geom_bar()`	Bar charts
Histogram	`geom_histogram()`	Histograms
Boxplot	`geom_boxplot()`	Boxplots
Density	`geom_density()`	Density plots
Smooth	`geom_smooth()`	Add regression or smoothing lines

Example: Scatterplot with `geom_point()`

```
ggplot(census_data, aes(x = Residents, y = mean_age, color = factor(Region))) +
 geom_point(size = 3)
```

6.2.3 Aesthetics: `aes()`

- Maps data variables to visual properties.

- Common aesthetics:
- `x`: X-axis variable.
- `y`: Y-axis variable.
- `color`: Changes point/line colors based on a variable.
- `fill`: Fills bars/areas based on a variable.
- `size`: Controls the size of points/lines.

Example: Adding color and size aesthetics

```
ggplot(census_data, aes(x = Residents, y = mean_age, color = factor(Region), size = pct_Very_
 geom_point()
```

What's missing in the plot above?How can this be improved?

6.2.4 Faceting: `facet_*`

Faceting splits data into subsets and creates multiple small plots.

Function	Description
`facet_wrap(~var)`	Wraps plots across rows/columns.
`facet_grid(row ~ col)`	Creates a grid layout for plots.

Faceting in is a technique to create multiple plots based on subsets of your data. It is particularly useful for comparing relationships or distributions across categories. By breaking down a dataset into smaller pieces according to a categorical variable, faceting helps visualise data trends or differences within subgroups. For instance, `facet_wrap(~ Region)` creates individual plots for each category in the Region variable, aligning them into rows and columns. Alternatively, `facet_grid(row ~ col)` creates a more structured layout, where two categorical variables determine the rows and columns of the grid.

Faceting is essential when dealing with data that needs to be compared across multiple dimensions. For example, a scatterplot faceted by sex can show how a relationship differs for males and females. Similarly, faceted histograms can reveal differences in the distribution of a variable across categories.

```
ggplot(census_data, aes(x = Residents, y = mean_age)) +
 geom_point() +
 facet_wrap(~Region)
```

What are the issues here?

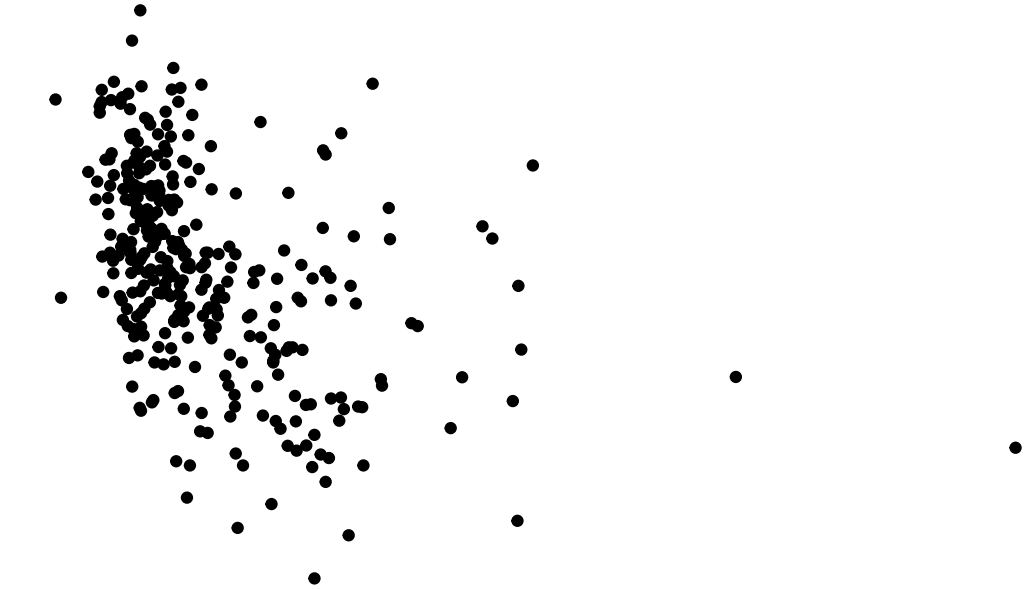
6.2.5 Themes (`theme_*`)

Themes control the overall appearance of the plot.

Function	Effect
`theme_minimal()`	Simple and clean theme.
`theme_classic()`	Classic-style plots.
`theme_dark()`	Dark background.
`theme_void()`	Minimal with no axes or grids.

Applying a theme

```
ggplot(data = census_data, aes(x = Residents, y = mean_age)) +
 geom_point() +
 theme_void() +
 labs(title = "Visualisation with Void Theme")
```



What do you think? Is this any good?

6.2.6 Scales

Scales adjust color, size, and axis properties.

Scale	Description
`scale_color_manual()`	Customizes colors for lines/points.
`scale_fill_brewer()`	Predefined color palettes for fills.
`scale_x_continuous()`	Modifies X-axis properties.
`scale_y_continuous()`	Modifies Y-axis properties.

`scale_color` functions modify the outline color of elements like points, lines, or the border of shapes, while `scale_fill` functions modify the interior fill color of shapes like bars, tiles, or boxes. Both are used to control aesthetics based on a variable mapped in `aes()` but apply to different visual aspects of the plot.

Example: Customizing axis and colors

The `RColorBrewer` library provides pre-defined color palettes specifically designed for data visualisation, ensuring clarity and accessibility. It includes sequential, diverging, and qualitative palettes suitable for various data types and visualisation needs.

Have a look at: https://r-graph-gallery.com/38-rcolorbrewers-palettes.html

```
# Automatically determine the number of colors needed

num_levels <- length(unique(census_data$Region))
```

The `num_levels` variable is a dynamic way to determine the number of unique levels in a categorical variable, such as `Region`. This approach is particularly helpful when you don't know in advance how many categories exist in the dataset.

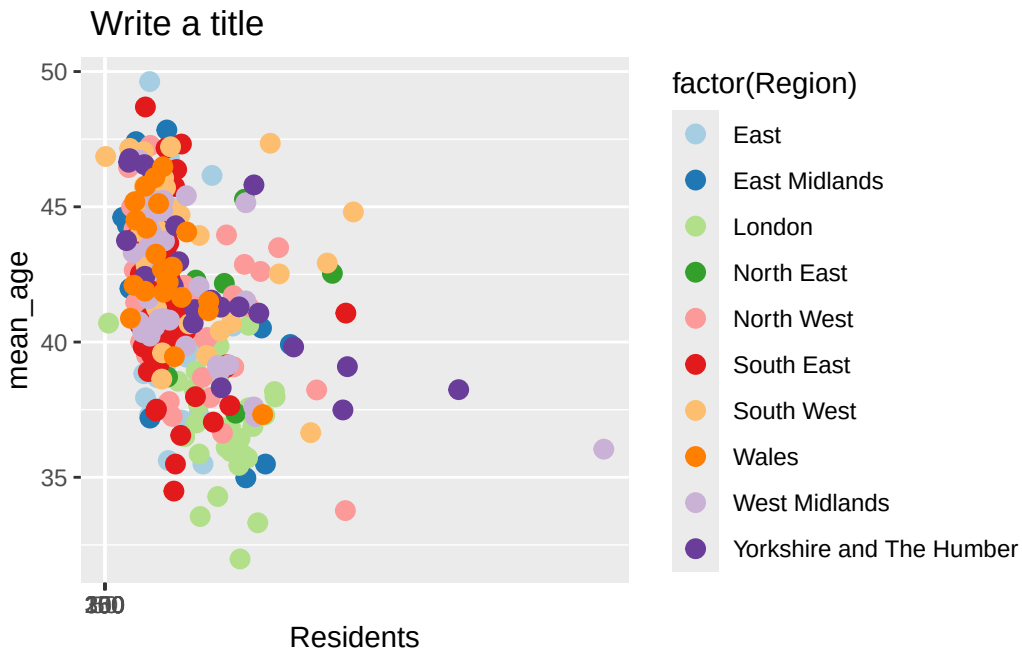
The code above calculates the number of unique categories in the `Region` variable, ensuring that your plot's color palette matches the data's requirements. Based on the value of `num_levels`, you can choose an appropriate color palette from libraries like `RColorBrewer`. For example:

```
library(RColorBrewer)
# Use an appropriate Brewer palette based on the number of levels
palette <- if (num_levels <= 8) "Set2" else "Paired" # Choose a palette with enough colors
```

This dynamically selects a palette suitable for the number of categories. If there are 8 or fewer levels, the `Set2` palette is used; for more levels, `Paired` ensures enough distinct colors (there's several palettes, check the link above for more).

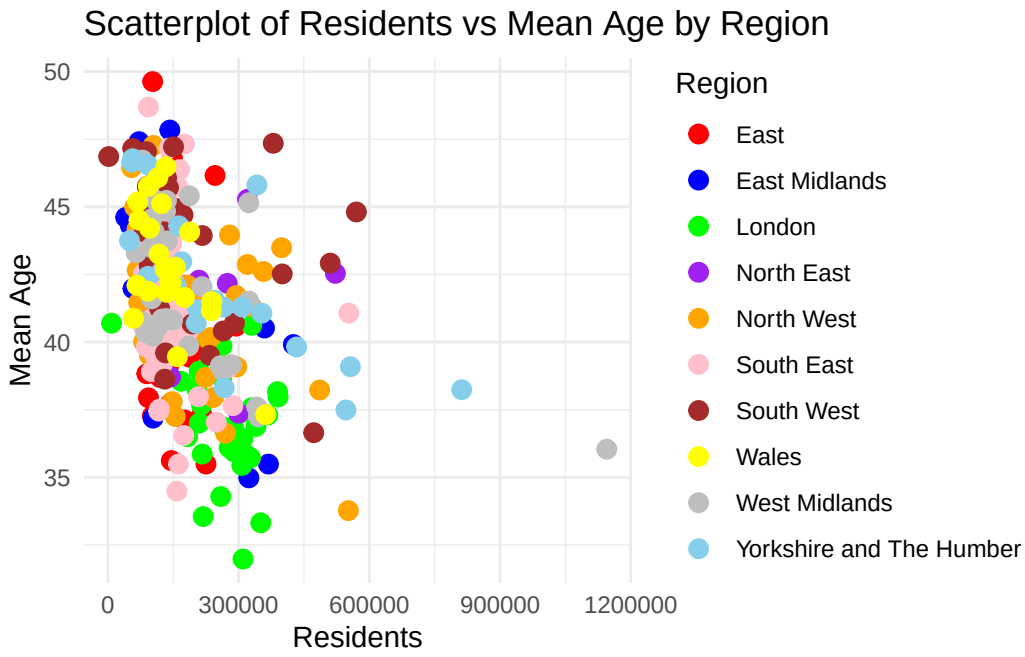
When applying the palette in `ggplot2`, the function`scale_color_brewer(palette = palette)` ensures that the plot assigns colors based on the selected scheme. This process avoids hardcoding colors or palettes and ensures the plot adapts to changes in the data, such as when categories are added or removed.

```
ggplot(census_data, aes(x = Residents, y = mean_age, color = factor(Region))) +
 geom_point(size = 3) +
 scale_color_brewer(palette = palette) + # Dynamically chosen palette
 scale_x_continuous(breaks = seq(50, 350, 50)) +
 labs(title = " Write a title")
```



You can also create your own palette of colours.

```
ggplot(census_data, aes(x = Residents, y = mean_age, color = factor(Region))) +
 geom_point(size = 3) +
 scale_color_manual(values = c("red", "blue", "green", "purple", "orange", "pink", "brown",
 labs(
  title = "Scatterplot of Residents vs Mean Age by Region",
  x = "Residents",
  y = "Mean Age",
  color = "Region"
 ) +
 theme_minimal()
```



Consider in general, that there's several functions associated to scales

Category	Function	Description	Example Usage
Discrete Color Scales	`scale_color_brewer()`	Use pre-defined discrete palettes from RColorBrewer.	`scale_color_brewer(palette = "Set1")`
	`scale_fill_brewer()`	Fill colors for discrete variables from RColorBrewer.	`scale_fill_brewer(palette = "Pastel2")`
	`scale_color_manual()`	Manually assign colors for discrete variables.	`scale_color_manual(values = c("red", "blue", "green"))`
	`scale_fill_manual()`	Manually assign fill colors for discrete variables.	`scale_fill_manual(values = c("purple", "orange", "yellow"))`
	`scale_color_viridis_d()`	Use discrete palettes from the viridis package (colorblind-friendly).	`scale_color_viridis_d(option = "plasma")`
	`scale_fill_viridis_d()`	Fill discrete variables with viridis palettes.	`scale_fill_viridis_d(option = "cividis")`

Category	Function	Description	Example Usage
Continuous Color Scales	`scale_color_gradient()`	Two-color gradient for continuous variables.	`scale_color_gradient(low = "blue", high = "red")`
	`scale_fill_gradient()`	Two-color gradient for continuous fill variables.	`scale_fill_gradient(low = "green", high = "yellow")`
	`scale_color_gradient2()`	Diverging gradient with a midpoint for continuous variables.	`scale_color_gradient2(low = "blue", mid = "white", high = "red", midpoint = 0)`
	`scale_fill_gradient2()`	Diverging gradient with a midpoint for continuous fill variables.	`scale_fill_gradient2(low = "purple", mid = "gray", high = "orange", midpoint = 50)`
	`scale_color_gradientn()`	Multi-color gradient for continuous variables.	`scale_color_gradientn(colors = c("blue", "green", "yellow", "red"))`
	`scale_fill_gradientn()`	Multi-color gradient for continuous fill variables.	`scale_fill_gradientn(colors = c("lightblue", "white", "pink"))`
	`scale_color_viridis_c()`	Continuous color scales from viridis.	`scale_color_viridis_c(option = "magma")`
	`scale_fill_viridis_c()`	Continuous fill scales from viridis.	`scale_fill_viridis_c(option = "inferno")`
Axis Scales	`scale_x_continuous()`	Customize numeric X-axis with limits and breaks.	`scale_x_continuous(limits = c(0, 100), breaks = seq(0, 100, 10))`
	`scale_y_continuous()`	Customize numeric Y-axis with limits and labels.	`scale_y_continuous(labels = scales::percent_format())`
	`scale_x_log10()`	Logarithmic transformation for X-axis.	`scale_x_log10()`
	`scale_y_log10()`	Logarithmic transformation for Y-axis.	`scale_y_log10()`

Category	Function	Description	Example Usage
	`scale_x_reverse()`	Reverse the X-axis direction.	`scale_x_reverse()`
	`scale_y_reverse()`	Reverse the Y-axis direction.	`scale_y_reverse()`
Shape and Size Scales	`scale_shape_manual()`	Manually assign shapes to categories for points.	`scale_shape_manual(values = c(19, 17, 15))`
	`scale_size_continuous()`	Scale the size of points based on a continuous variable.	`scale_size_continuous(range = c(1, 10))`
	`scale_size_manual()`	Manually specify point sizes for categorical variables.	`scale_size_manual(values = c(2, 4, 6))`

6.2.7 Additional Functions for Customization

- **`labs()`**: Adds labels for axes, title, and legend.
- **`coord_flip()`**: Flips X and Y axes for horizontal plots.

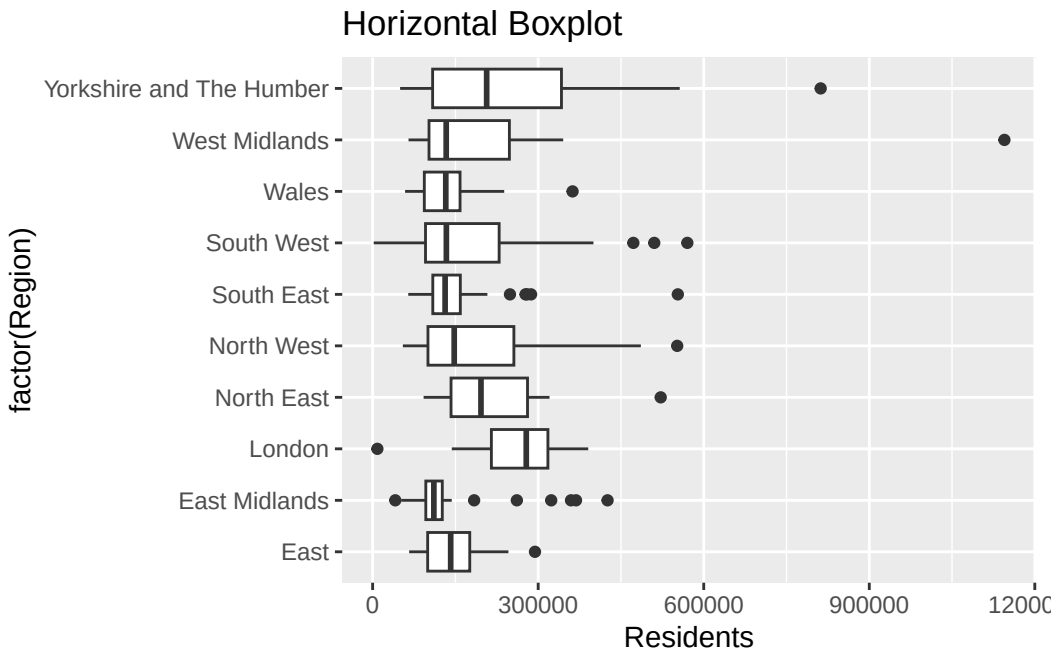
Labeling:

```
# labels
labs(
 title = "Plot Title",
 x = "X-Axis Label",
 y = "Y-Axis Label",
 color = "Legend Title"
)
```

```
<ggplot2::labels> List of 4
 $ x     : chr "X-Axis Label"
 $ y     : chr "Y-Axis Label"
 $ colour: chr "Legend Title"
 $ title : chr "Plot Title"
```

Flipping:

```r
# flipping
ggplot(census_data, aes(x = factor(Region), y = Residents)) +
 geom_boxplot() +
 coord_flip() +
 labs(title = "Horizontal Boxplot")
```



Key Tips for `ggplot2`:

- Start with simple plots and incrementally add layers (`+`).
- Use themes (`theme_minimal()`, `theme_classic()`) to clean up your plots.
- Explore palettes with `RColorBrewer` or `viridis` for colorblind-friendly options.

6.3 Part 3 - Visualisation: Making decent graphs (1h)

This section demonstrates how to create **publication-quality visualisations** in R using `ggplot2`.

Learning goals Developing an understanding of, and ability to create academic standard data visualisations.

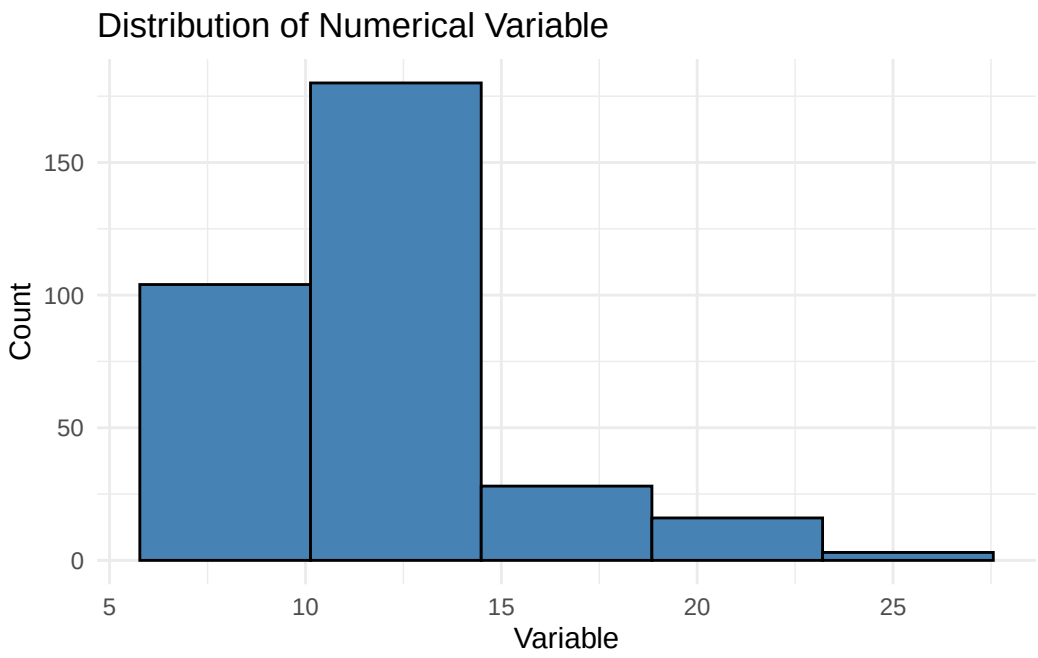
Let's load the datasets first Check the paths, you may need to remove the ".."

```r
frs_data <- read.csv("../data/FRS/FRS16-17_labels.csv")
census_data <- read.csv("../data/Census2021/EW_DistrictPercentages.csv")
```

6.3.1 Distribution of 1 Numerical variable:

Histogram

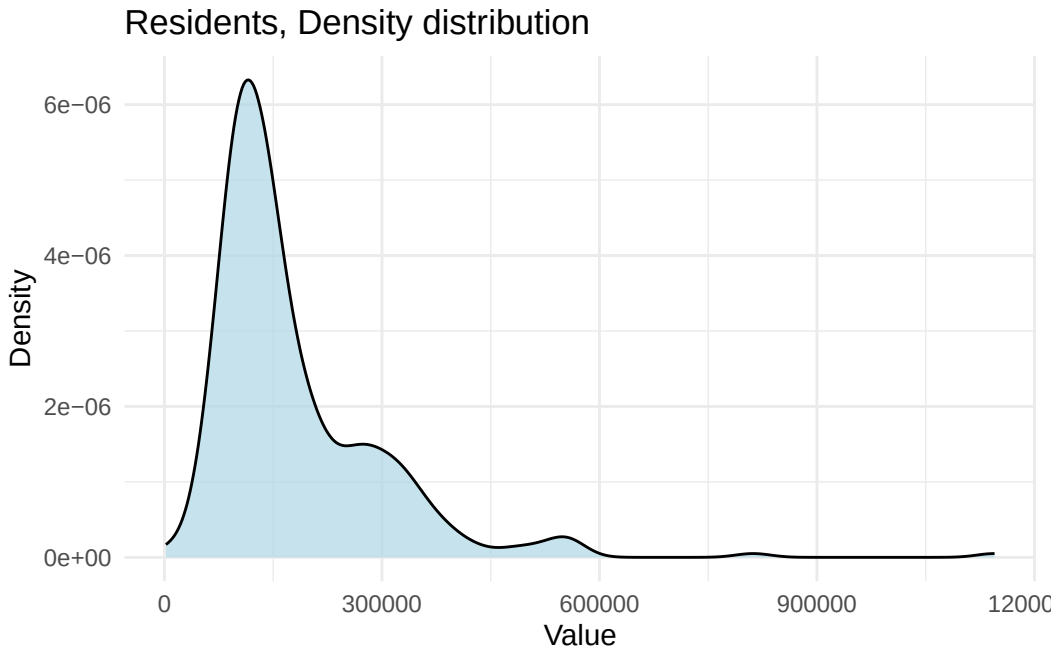
```r
ggplot(census_data, aes(x = pct_Age_20_to_29)) +
 geom_histogram(bins = 5, fill = "steelblue", color = "black") +
 labs(title = "Distribution of Numerical Variable", x = "Variable", y = "Count") +
 theme_minimal()
```



The `bins` parameter in a histogram determines the number of intervals (bins) into which the data range is divided. Each bin represents a range of values along the x-axis, and the height of each bar shows the count of observations within that range. For example, if bins = 5, the data is divided into five equal-width intervals. Using fewer bins results in a coarser view of the data, while more bins provide a finer, more detailed representation. However, too many bins can make the histogram appear cluttered and difficult to interpret. The width of each bin is automatically calculated by dividing the range of the data by the number of bins.

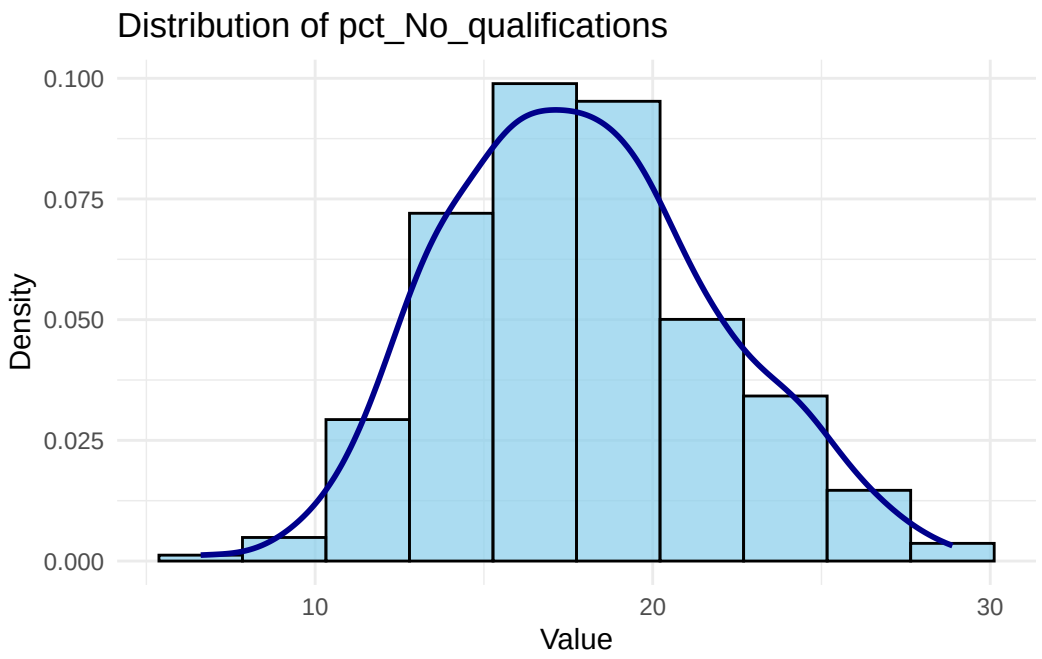
Density Plot

```r
ggplot(census_data, aes(x = Residents)) +
 geom_density(fill = "lightblue", alpha = 0.7) +
 labs(title = "Residents, Density distribution", x = "Value", y = "Density") +
 theme_minimal()
```



Histogram + Density Distribution

```r
# Plot histogram with density overlay for a chosen variable (e.g., 'pct_No_qualifications')
ggplot(census_data, aes(x = pct_No_qualifications)) +
    geom_histogram(aes(y = after_stat(density)), bins = 10, color = "black", fill = "skyblue"
    geom_density(color = "darkblue", linewidth = 1) +
    labs(title = "Distribution of pct_No_qualifications", x = "Value", y =  "Density") +
 theme_minimal()
```



Box Plot

A box plot, also known as a box-and-whisker plot, is a graphical representation of the distribution of a dataset. It displays the minimum, first quartile (Q1), median (Q2), third quartile (Q3), and maximum values, with the central box representing the interquartile range (IQR) from Q1 to Q3. The whiskers extend from the box to show the range of the data, excluding outliers. Outliers are typically represented as individual points outside the whiskers.

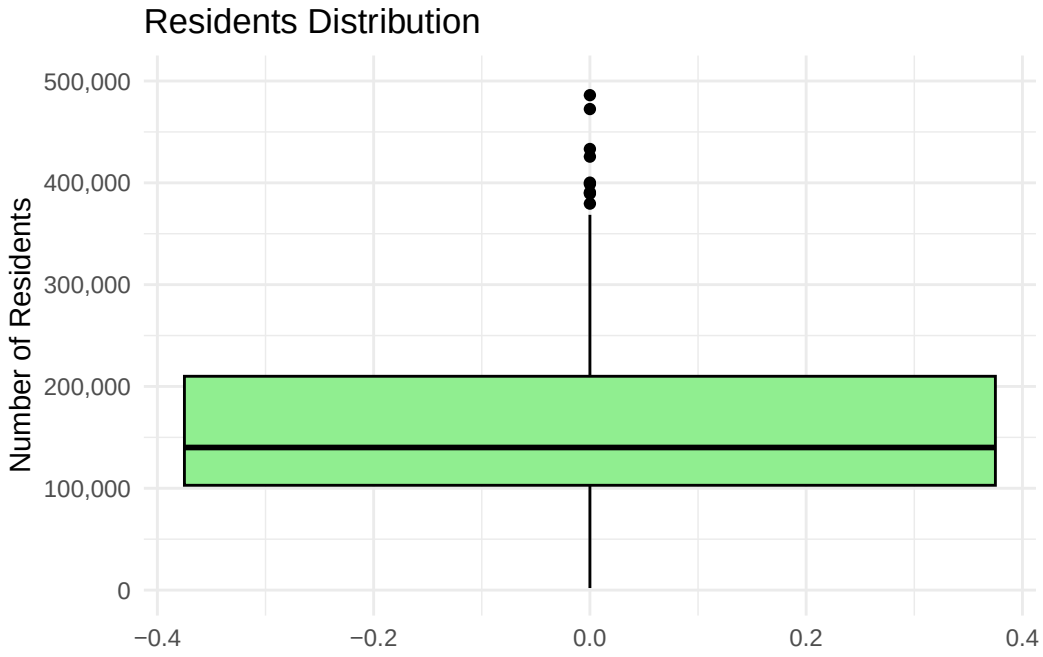
The box plot provides a concise summary of the data distribution, highlighting the spread, center, and potential skewness.

```
library(scales) # For label_comma()
```

```
Warning: package 'scales' was built under R version 4.4.3
```

```
ggplot(census_data, aes(y = Residents)) +
 geom_boxplot(fill = "lightgreen", color = "black") +
 labs(title = "Residents Distribution", y = "Number of Residents") +
 theme_minimal() +
 scale_y_continuous(
  limits = c(0, 500000),
  labels = label_comma() # Properly placed within scale_y_continuous()
 )
```

```
Warning: Removed 9 rows containing non-finite outside the scale range
(`stat_boxplot()`).
```




```
# Setting axis limits for better readability and applying comma-separated format to large nu
```

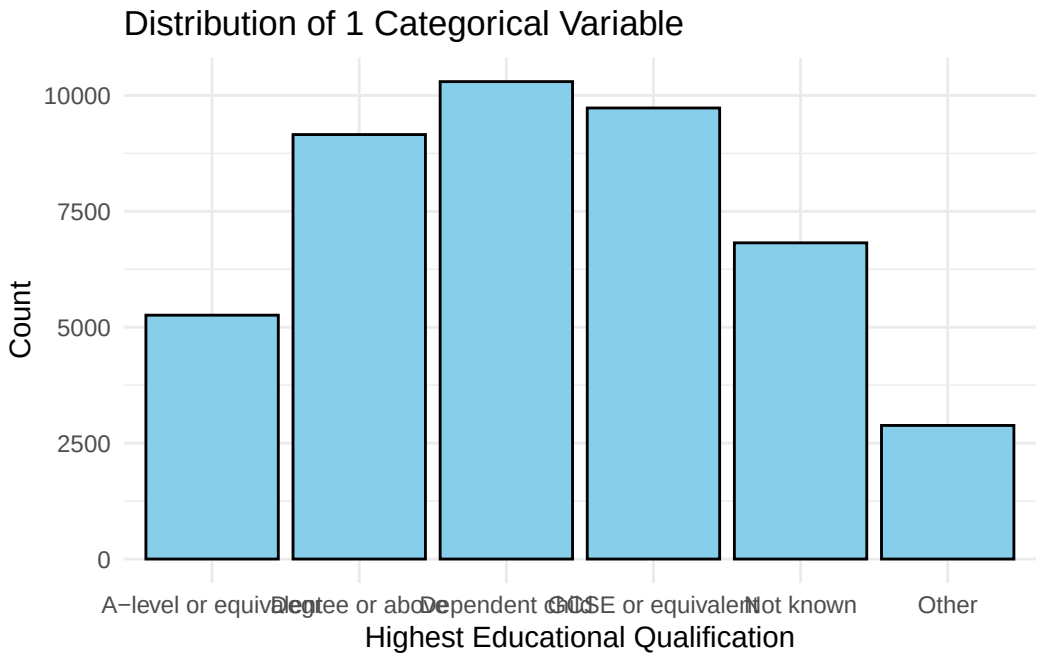
This looks kind of sad with just one variable, but it's the best way, usually, to plot distributions for numerical variables.

Notice the **outliers**. Outliers are data points that deviate significantly from the overall pattern of a dataset. They can arise due to measurement errors, variability in the data, or the presence of extreme values. Outliers can heavily influence statistical analyses, such as means or regression models, potentially leading to misleading conclusions. Identifying and addressing outliers is crucial to ensure robust and accurate results. Boxplots allow spotting them, along with interquartile range (IQR) analysis.

6.3.2 Distribution of 1 Categorical variable:

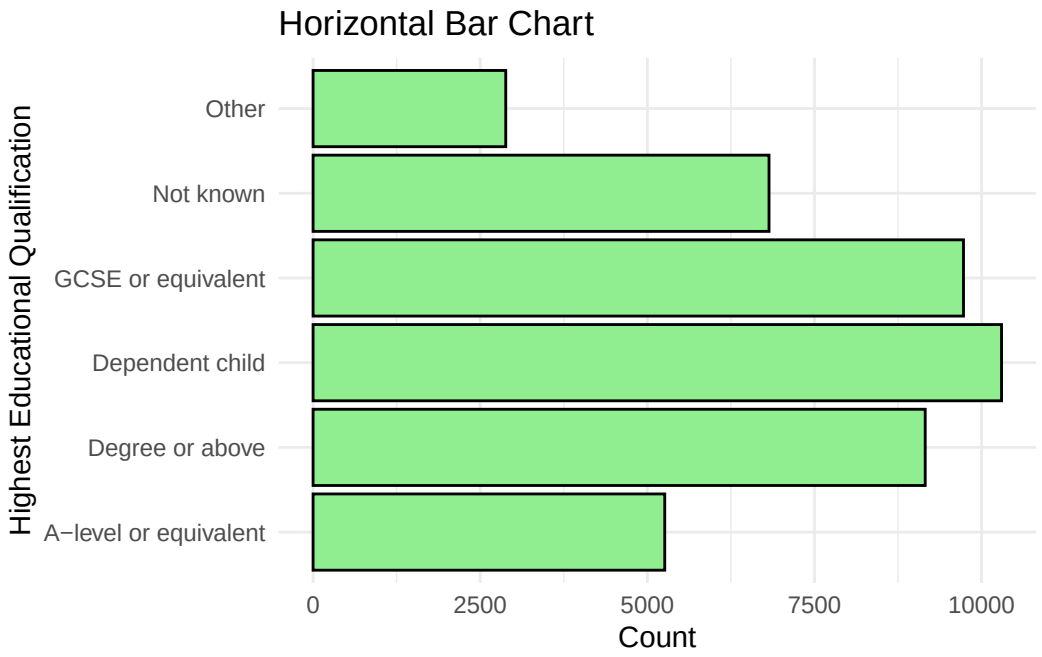
Bar Chart

```
ggplot(frs_data, aes(x = highest_qual)) +
 geom_bar(fill = "skyblue", color = "black") +
 labs(title = "Distribution of 1 Categorical Variable", x = "Highest Educational Qualificati
 theme_minimal()
```



Horizontal Bar Chart

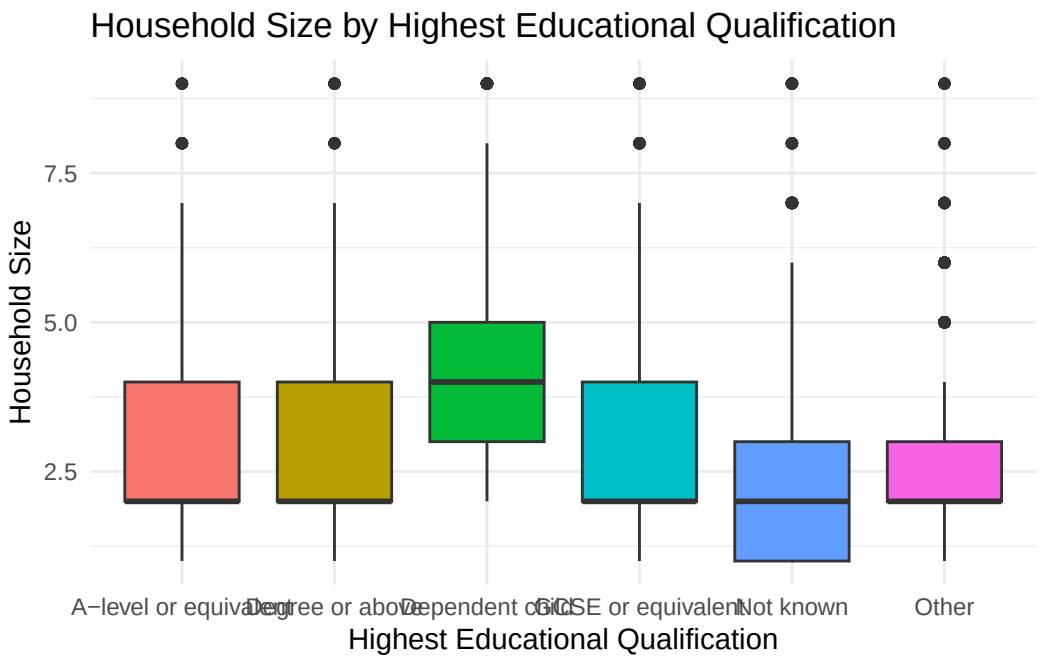
```
ggplot(frs_data, aes(x = highest_qual)) +
 geom_bar(fill = "lightgreen", color = "black") +
 labs(title = "Horizontal Bar Chart", x = "Highest Educational Qualification", y = "Count",
 coord_flip() + #just flipping
 theme_minimal()
```



6.3.3 Comparing variables

1 numerical, 1 categorical: Boxplot

```
ggplot(frs_data, aes(x = highest_qual, y = hh_size, fill = highest_qual)) +
 geom_boxplot() +
 labs(
  title = "Household Size by Highest Educational Qualification",
  x = "Highest Educational Qualification",
  y = "Household Size"
 ) +
 theme_minimal() +
 theme(legend.position = "none") # don't need this
```



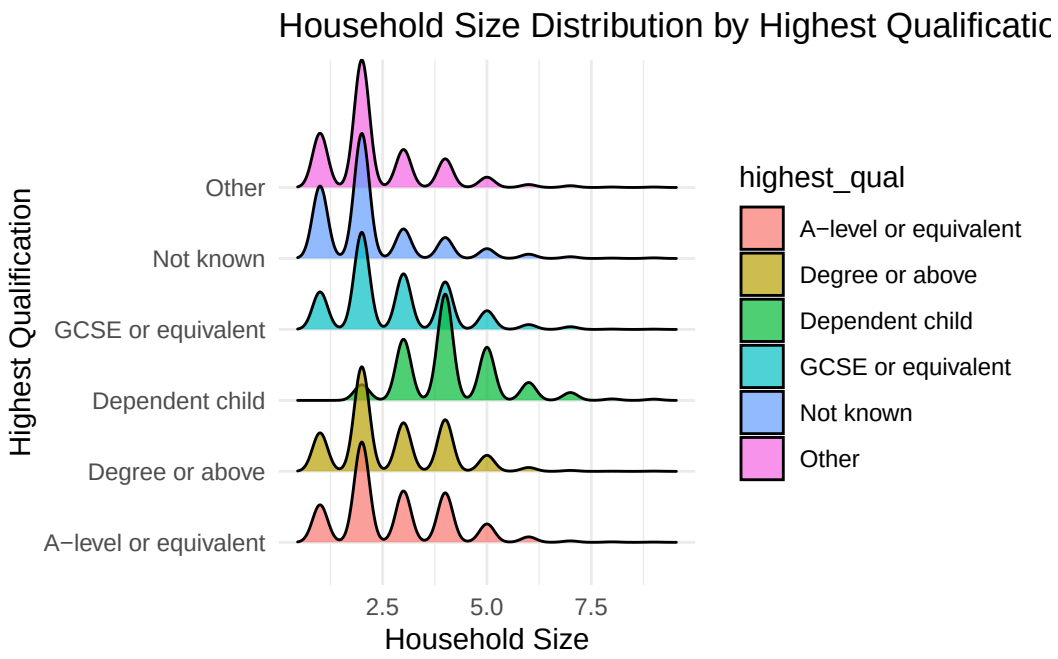
1 numerical, 1 categorical: Density Ridges

```
library(ggridges) # need a library for this
```

```
Warning: package 'ggridges' was built under R version 4.4.2
```

```
ggplot(frs_data, aes(x = hh_size, y = highest_qual, fill = highest_qual)) +
 geom_density_ridges(alpha = 0.7) +
 labs(
  title = "Household Size Distribution by Highest Qualification",
  x = "Household Size",
  y = "Highest Qualification"
 ) +
 theme_minimal()
```

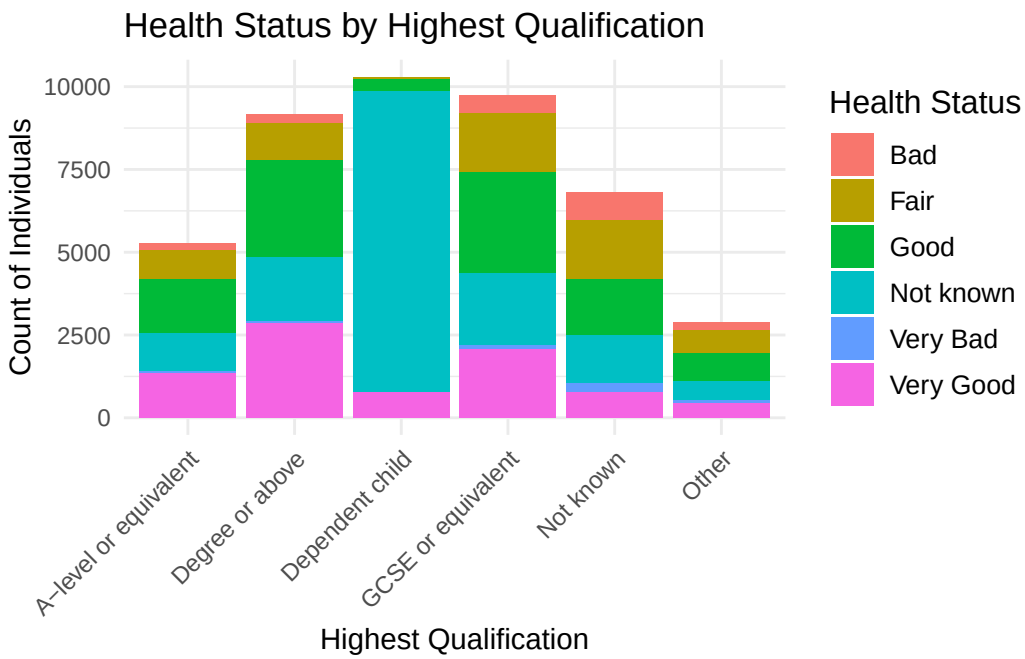
```
Picking joint bandwidth of 0.18
```



2 Categorical Variables: Stacked Bar Chart

Note the adjustments made for making the labels look better

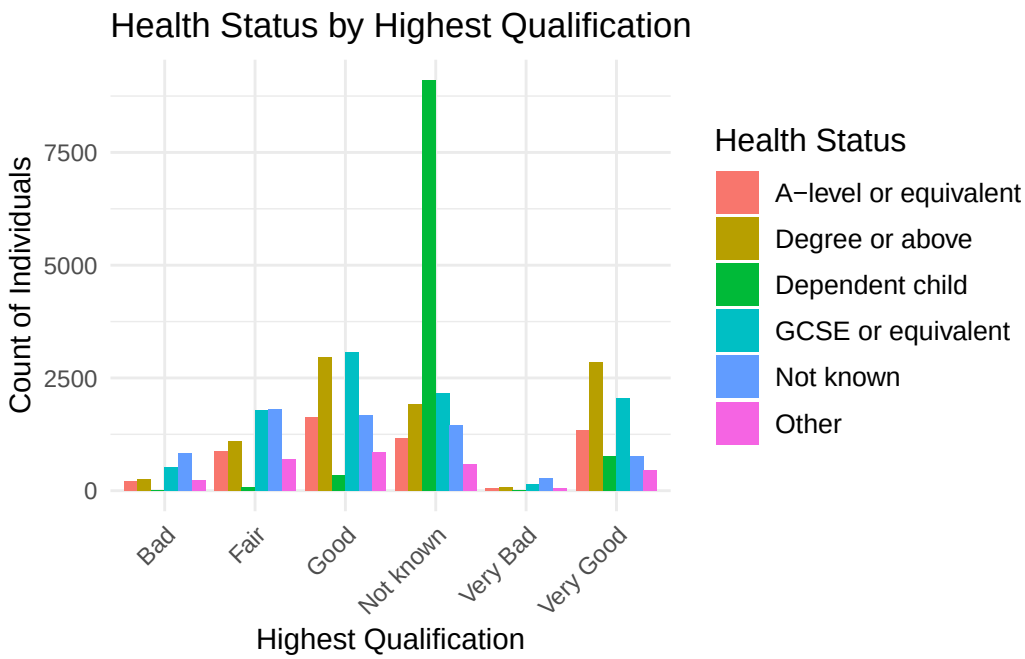
```
ggplot(frs_data, aes(x = highest_qual, fill = health)) +
 geom_bar(position = "stack") +
 labs(
  title = "Health Status by Highest Qualification",
  x = "Highest Qualification",
  y = "Count of Individuals",
  fill = "Health Status"
 ) +
 theme_minimal() +
 theme(
  axis.text.x = element_text(angle = 45, hjust = 1), # Rotate X-axis labels for readability
  legend.title = element_text(size = 12),      # Adjust legend title size
  legend.text = element_text(size = 10)       # Adjust legend text size
 )
```



Side-by-Side Bar Chart (Two-Way Frequency Distribution)

This type of chart can get messy.

```
ggplot(frs_data, aes(x = health, fill = highest_qual)) +
 geom_bar(position = "dodge") +
 labs(
  title = "Health Status by Highest Qualification",
  x = "Highest Qualification",
  y = "Count of Individuals",
  fill = "Health Status"
 ) +
 theme_minimal() +
 theme(
  axis.text.x = element_text(angle = 45, hjust = 1),
  legend.title = element_text(size = 12),
  legend.text = element_text(size = 10)
 )
```



2 Categorical Variable counts to percentages: Stacked Percentage Graph

One can also convert the counts of two categorical variables to percentages. In this case, we aim to visualise the proportional distribution of one categorical variable (`health`) across levels of another categorical variable (`highest_qual`) using a stacked percentage bar chart. First, we create a cross-tabulation to count the occurrences of each combination of Health and Qualification categories. These counts are then converted into a percentage format relative to the total counts for each Qualification category. This transformation allows us to represent the relative proportions rather than absolute counts, facilitating comparison across categories.

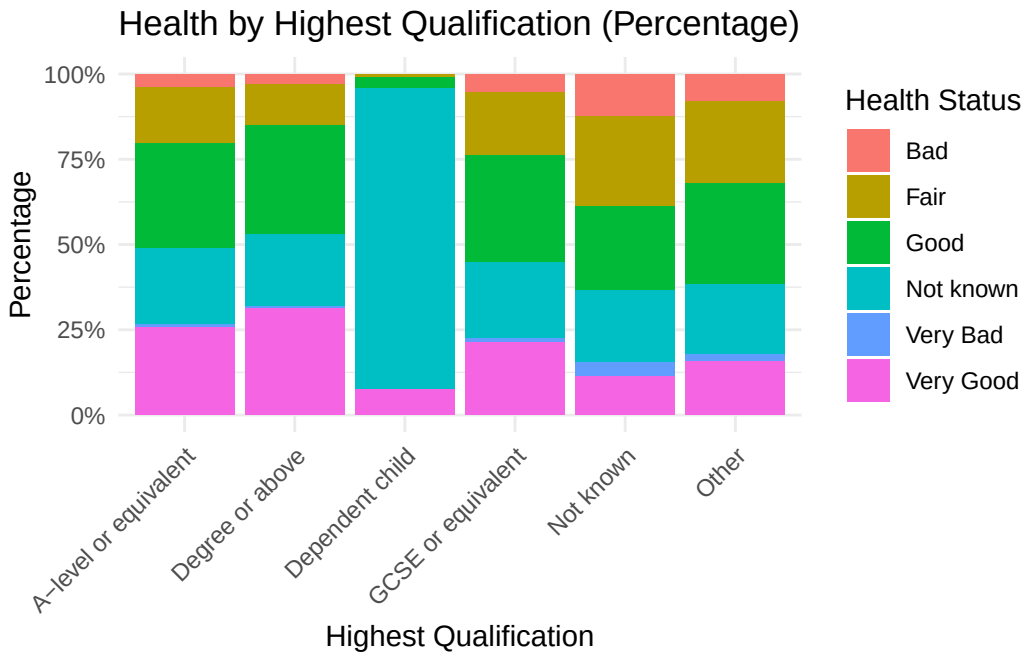
A *cross tabulation* (or contingency table) is a statistical tool used to analyze the relationship between two or more categorical variables. It organizes data into a table format where: - Rows represent the categories of one variable. - Columns represent the categories of another variable. - Cells display the counts or frequencies of data points that fall into each combination of the row and column categories.

```
# Create a cross-tabulation of observed counts
cross_tab <- table(frs_data$health, frs_data$highest_qual)

# Convert the cross-tabulation to a data frame
cross_tab_df <- as.data.frame(cross_tab)
colnames(cross_tab_df) <- c("Health", "Qualification", "Percentage")
```

The data is reshaped into a data frame for compatibility with `ggplot2`, where a stacked bar

chart with percentage scaling (`position = "fill"`) is generated. The result is a plot that shows how Health statuses are distributed proportionally within each Qualification level.

```
# Create a stacked percentage bar chart
ggplot(cross_tab_df, aes(x = Qualification, y = Percentage, fill = Health)) +
 geom_bar(stat = "identity", position = "fill") +
 scale_y_continuous(labels = scales::percent_format()) +
 labs(
  title = "Health by Highest Qualification (Percentage)",
  x = "Highest Qualification",
  y = "Percentage",
  fill = "Health Status"
 ) +
 theme_minimal() +
 theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



3+ Numerical Variables: Boxplot

To compare three numerical variables using boxplots in ggplot2, one needs to reshape the data to a long format so that each numerical variable is treated as a category in a single column.

Here's how you can do it, using `tidyverse`

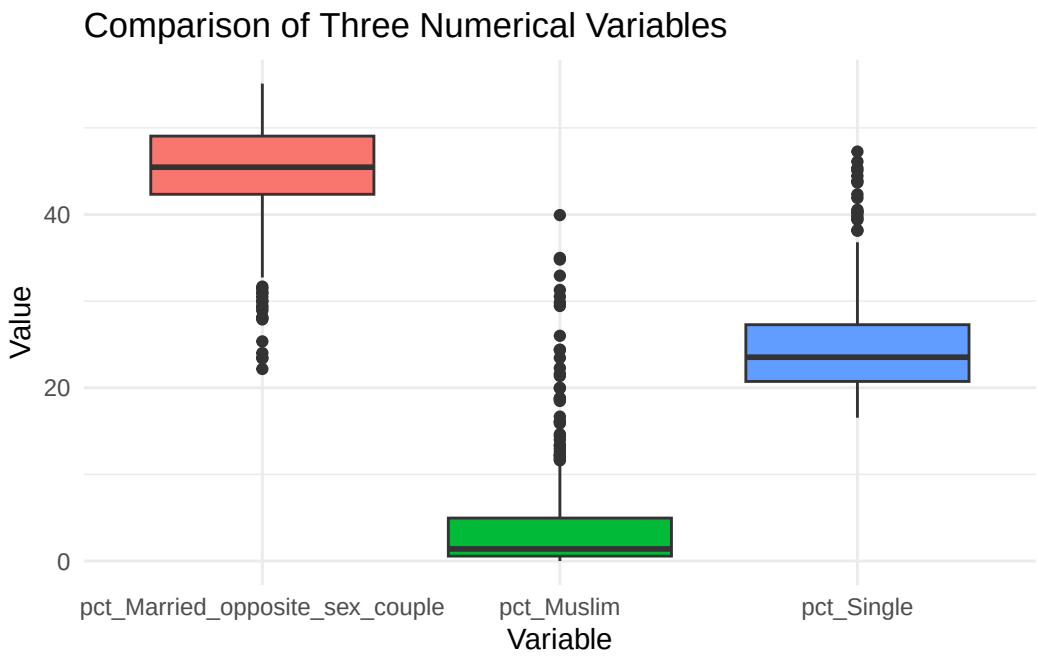
```
library(tidyverse) # we need tidyverse for this
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v lubridate 1.9.3     v tibble    3.2.1
v purrr     1.0.2     v tidyr     1.3.1
-- Conflicts ---------------------------------------- tidyverse_conflicts() --
x readr::col_factor() masks scales::col_factor()
x purrr::discard()    masks scales::discard()
x dplyr::filter()     masks stats::filter()
x dplyr::lag()        masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
# Reshape the data into long format
long_data <- census_data %>%
 pivot_longer(
  cols = c(pct_Single, pct_Muslim, pct_Married_opposite_sex_couple),
  names_to = "Variable",
  values_to = "Value"
 )
```

The code above reshapes the data from a wide format to a long format using the `pivot_longer()` function from the `tidyverse` package. In the original dataset, each column represents a separate variable (e.g., `pct_Single`, `pct_Muslim`, ..), and their values are stored in individual columns. The transformation collapses these columns into two new columns: one for the variable names (Variable) and another for their corresponding values (Value). This format is useful for plotting or analysis where variables are treated uniformly, such as when creating boxplots or facet grids in ggplot2.

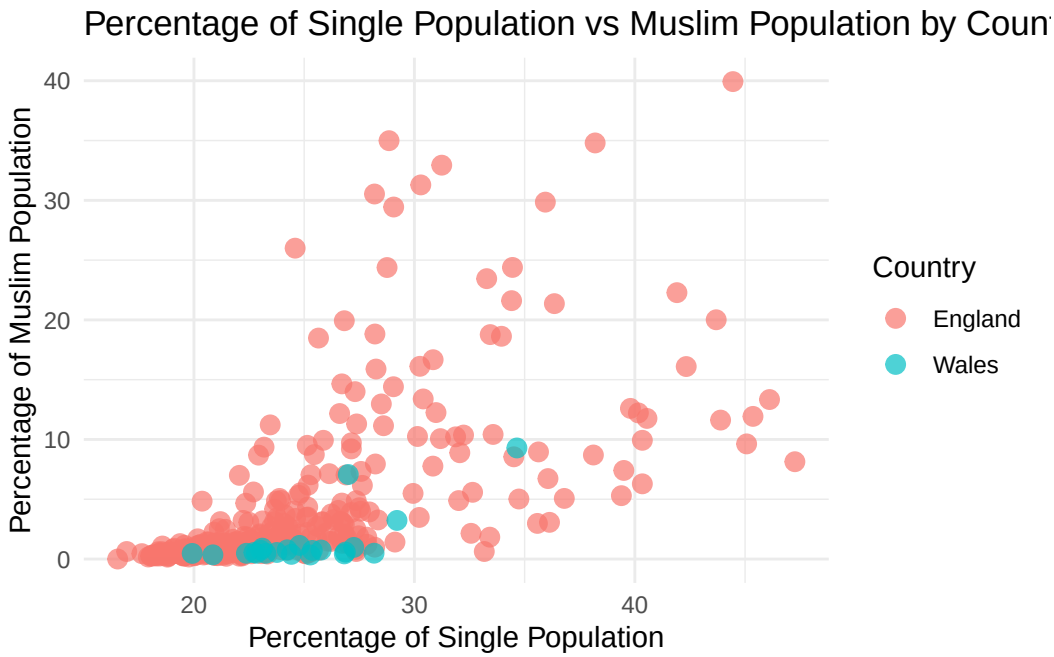
```
# Create the boxplot
ggplot(long_data, aes(x = Variable, y = Value, fill = Variable)) +
 geom_boxplot() +
 labs(
  title = "Comparison of Three Numerical Variables",
  x = "Variable",
  y = "Value"
 ) +
 theme_minimal() +
 theme(legend.position = "none")
```



6.3.4 Visualising Relationships:

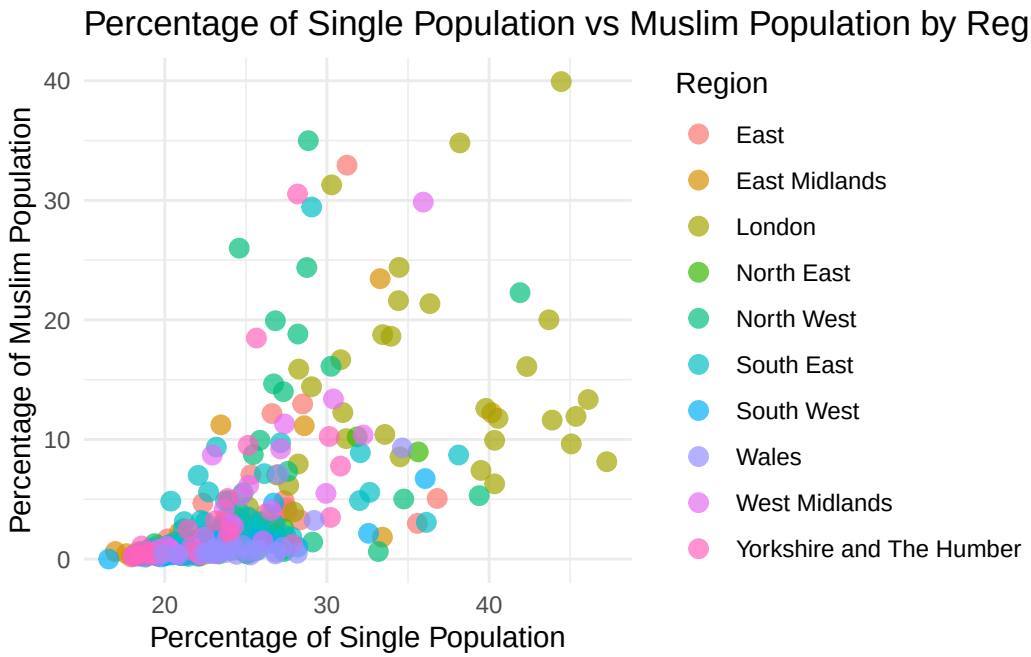
2 Numerical Variables: Scatterplot

```
ggplot(census_data, aes(x = pct_Single, y = pct_Muslim, color = factor(Country))) +
 geom_point(size = 3, alpha = 0.7) +
 labs(
  title = "Percentage of Single Population vs Muslim Population by Country",
  x = "Percentage of Single Population",
  y = "Percentage of Muslim Population",
  color = "Country"
 ) +
 theme_minimal()
```



2 Numerical Variables + 1 Catecorical: Scatterplot

```
ggplot(census_data, aes(x = pct_Single, y = pct_Muslim, color = factor(Region))) +
 geom_point(size = 3, alpha = 0.7) +
 labs(
  title = "Percentage of Single Population vs Muslim Population by Region",
  x = "Percentage of Single Population",
  y = "Percentage of Muslim Population",
  color = "Region"
 ) +
 theme_minimal()
```



3+ Numerical Variables: Correlogram

A correlogram is a visual representation of a correlation matrix, where the strength and direction of relationships between numerical variables are displayed. The matrix is typically a grid, with variables listed along both the rows and columns. Each cell in the matrix shows the correlation coefficient between two variables.

The library `corrplot` would do it for us. Check: https://cran.r-project.org/web/packages/corrplot/vignettes/co intro.html for deeper customisation.

In the corrplot package, a correlogram can use colors, shapes, or numbers to represent the correlation. For example, colors ranging from blue to red often signify negative to positive correlations, while white may indicate no correlation. Similarly, circles or other shapes can vary in size to depict the strength of the relationship. The corrplot function simplifies creating correlograms with customizable options. For example, using method = "number" places correlation coefficients in each cell, while method = "circle" represents correlations graphically.

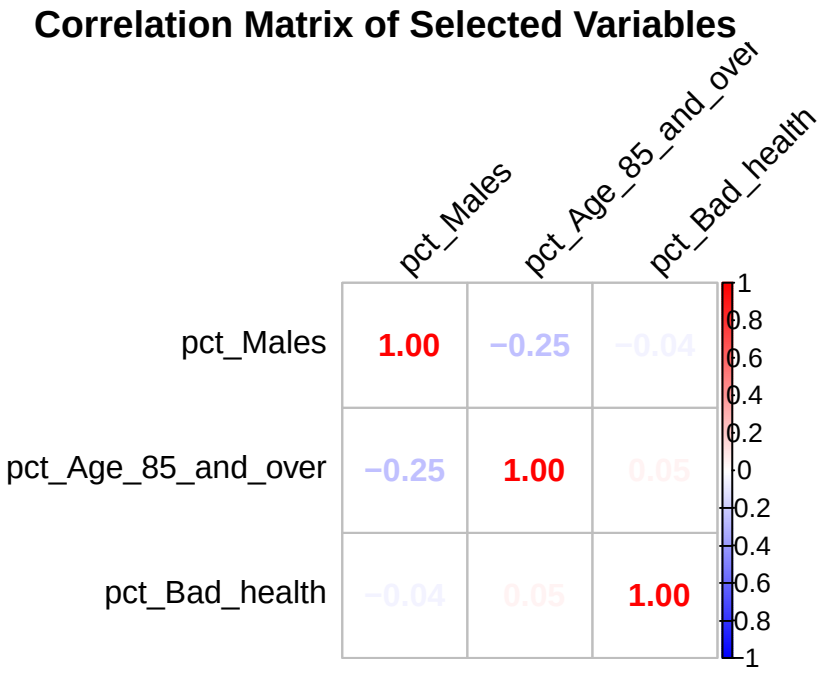
Correlograms are valuable for quickly identifying patterns or strong relationships between variables, such as spotting which features might influence one another in a dataset

```
library(corrplot)
```

```
Warning: package 'corrplot' was built under R version 4.4.2
```

```
corrplot 0.95 loaded
```

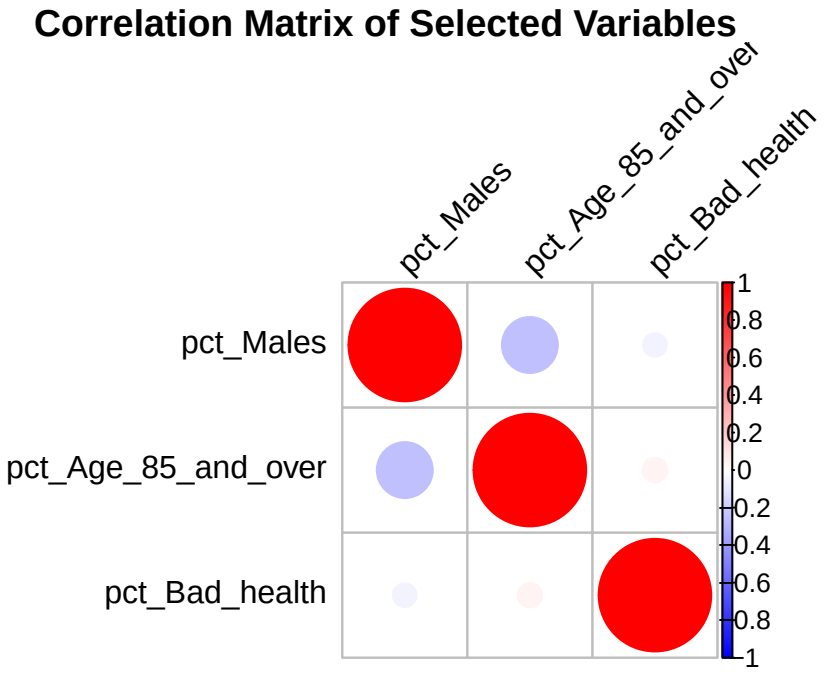
```r
# Select the necessary variables from the dataset
selected_vars <- census_data[, c("pct_Males", "pct_Age_85_and_over", "pct_Bad_health")]

# Calculate the correlation matrix
cor_matrix <- cor(selected_vars, use = "complete.obs")

# Generate the correlation plot
corrplot(cor_matrix,
     method = "number", # Display numbers
     col = colorRampPalette(c("blue", "white", "red"))(200), # Color gradient
     tl.col = "black", # Text color for labels
     tl.srt = 45,    # Rotate text labels
     title = "Correlation Matrix of Selected Variables",
     mar = c(0, 0, 1, 0)) # Adjust margins
```



We can also include circles to graphically represent correlations.

```r
# Generate the correlation plot with circles
corrplot(cor_matrix,
     method = "circle", # Display circles
     col = colorRampPalette(c("blue", "white", "red"))(200), # Color gradient
     tl.col = "black", # Text color for labels
     tl.srt = 45,    # Rotate text labels
```

```
     title = "Correlation Matrix of Selected Variables",
     mar = c(0, 0, 1, 0)) # Adjust margins
```



3+ Numerical Variables: Scatterplot matrix

Preparation:

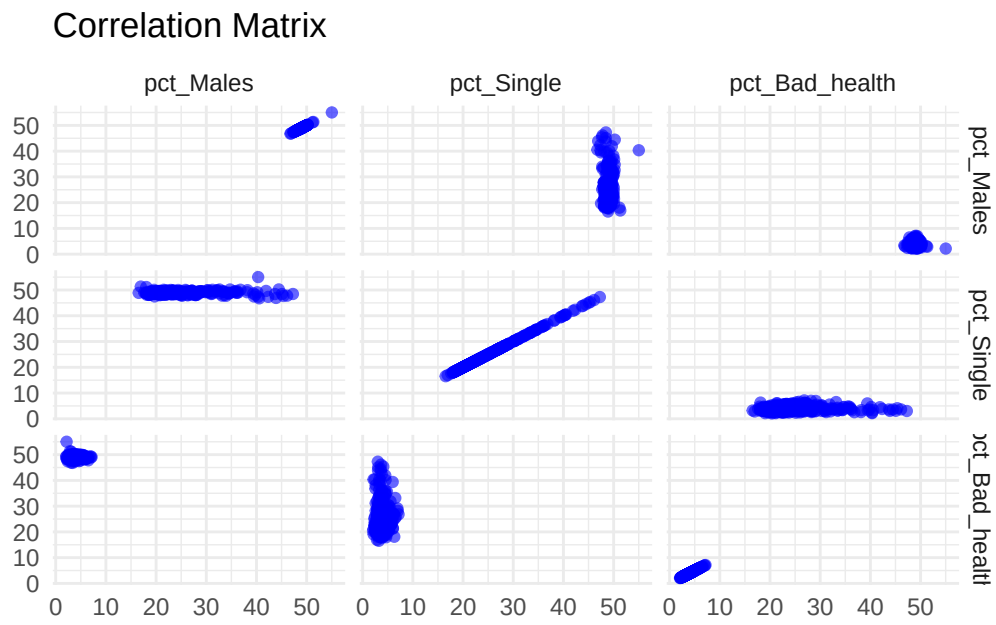
```
library(tidyverse)

# Select the variables
selected_vars <- census_data[, c("pct_Males", "pct_Single", "pct_Bad_health")]

# Create pairwise combinations of variable names
scatter_data <- expand.grid(
 Variable1 = names(selected_vars),
 Variable2 = names(selected_vars)
)

# Add the data values for the pairs
scatter_data <- scatter_data %>%
 rowwise() %>%
 mutate(
  Value1 = list(selected_vars[[Variable1]]), # Extract values for Variable1
```

```
  Value2 = list(selected_vars[[Variable2]]) # Extract values for Variable2
 ) %>%
 unnest(c(Value1, Value2)) # Unnest the lists into rows
```

This code is used to create a dataset that facilitates generating pairwise scatterplots between selected variables from a dataset. First, it selects the variables of interest (`pct_Males`, `pct_Single`, and `pct_Bad_health`) from the census_data dataset and stores them in a new data frame called `selected_vars`. Then, the `expand.grid()` function is used to generate all possible combinations of these variables, resulting in a data frame with two columns, `Variable1` and `Variable2`, where each row represents a pair of variables (e.g., `pct_Males` vs `pct_Single`). Finally, the `mutate()` function is applied to map the actual values of the paired variables (Variable1 and Variable2) to two new columns, Value1 and Value2. These columns contain the corresponding data points for the variable pair, enabling the creation of scatterplots where Value1 is plotted against Value2 for each combination.

Visualisation:

```
ggplot(scatter_data, aes(x = Value1, y = Value2)) +
 geom_point(alpha = 0.6, color = "blue") +
 facet_grid(Variable1 ~ Variable2, scales = "free") +
 labs(
  title = "Correlation Matrix",
  x = "",
  y = ""
 ) +
 theme_minimal()
```



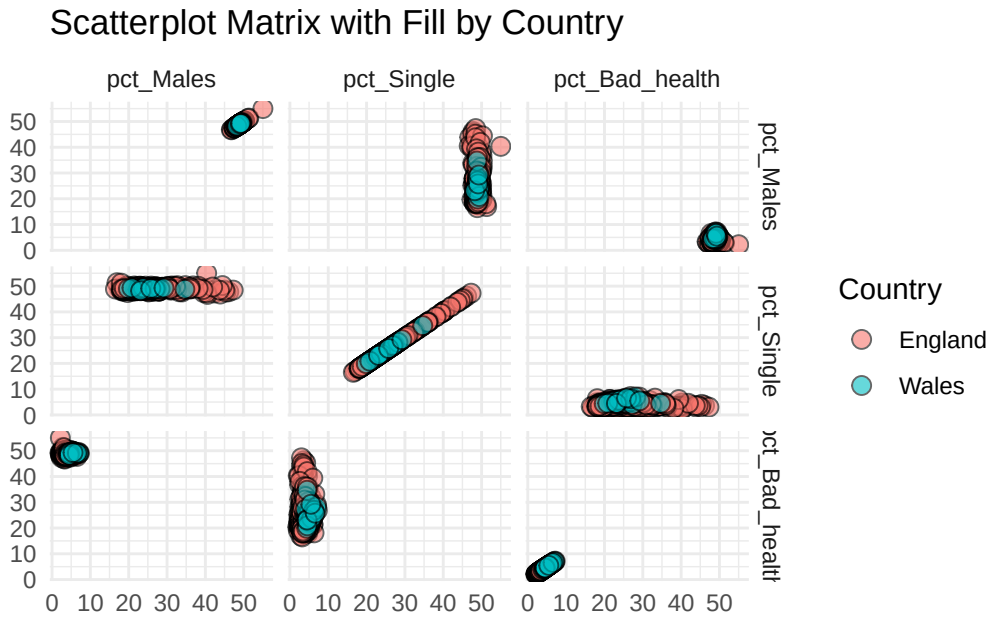
You can also include a categorical variable through colouring. Let's try with `Country`.

```
# Ensure the original data includes the 'Country' variable
selected_vars <- census_data[, c("pct_Males", "pct_Single", "pct_Bad_health", "Country")]

# Create pairwise combinations of variable names (excluding 'Country')
scatter_data <- expand.grid(
  Variable1 = names(selected_vars)[-4], # Exclude 'Country'
  Variable2 = names(selected_vars)[-4]
)
#  %>% filter(Variable1 != Variable2) # Remove self-comparisons

# Add the data values for the pairs
scatter_data <- scatter_data %>%
  rowwise() %>%
  mutate(
    Value1 = list(selected_vars[[Variable1]]), # Extract values for Variable1
    Value2 = list(selected_vars[[Variable2]]), # Extract values for Variable2
    Country = list(selected_vars$Country)     # Include 'Country' in the data
  ) %>%
  unnest(c(Value1, Value2, Country)) # Unnest the lists into rows
```

Now, the scatter_data includes the `Country` column, which can be used to map fill in

ggplot:

```
ggplot(scatter_data, aes(x = Value1, y = Value2, fill = Country)) +
  geom_point(alpha = 0.6, shape = 21, size = 3) + # Use shape with fill (e.g., 21)
  facet_grid(Variable1 ~ Variable2, scales = "free") +
  labs(
    title = "Scatterplot Matrix with Fill by Country",
    x = "",
    y = "",
    fill = "Country"
  ) +
  theme_minimal()
```

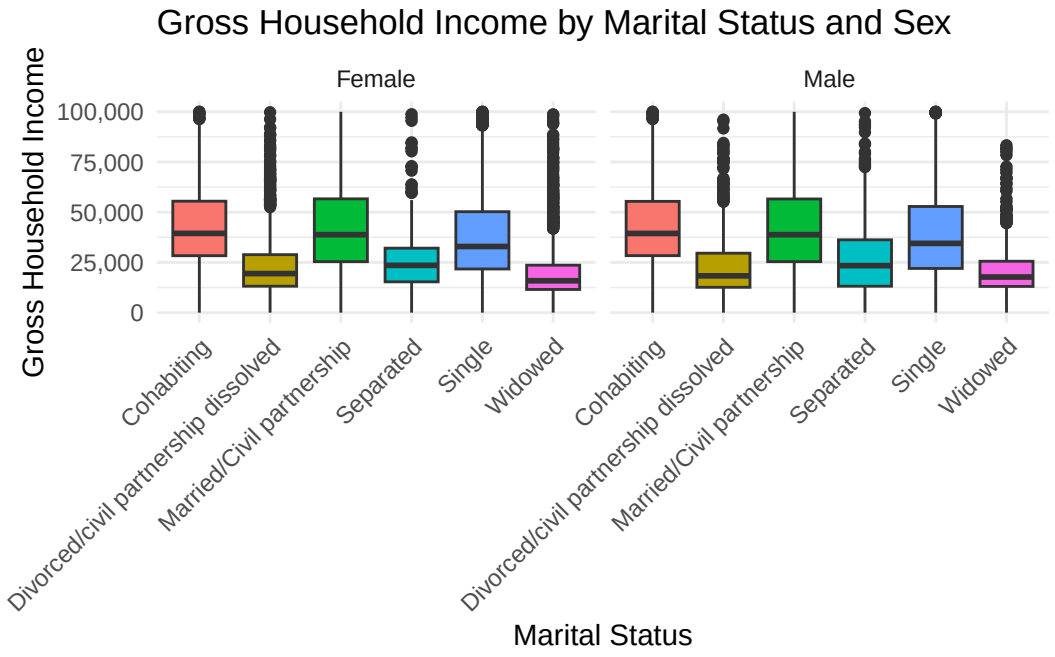


1 Numerical Variables, 2+ Cateorical Variables: Boxplot faceting

```
ggplot(frs_data, aes(x = marital_status, y = hh_income_gross, fill = marital_status)) +
 geom_boxplot() +
 facet_wrap(~ sex) +
 scale_y_continuous(
  limits = c(0, 100000), # limit the y-axis
  labels = label_comma() # Properly placed within scale_y_continuous()
 ) +
 labs(
```

```
 title = "Gross Household Income by Marital Status and Sex",
 x = "Marital Status",
 y = "Gross Household Income"
) + # Removed 'fill' label to avoid legend creation
theme_minimal() +
theme(
 axis.text.x = element_text(angle = 45, hjust = 1), # Rotate x-axis labels for readability
 legend.position = "none" # Corrected to lowercase 'none'
)
```

```
Warning: Removed 2769 rows containing non-finite outside the scale range
(`stat_boxplot()`).
```



1 Numerical Variables, 2+ Cateorical Variables: Violin Plot faceting

A violin plot combines aspects of a box plot and a density plot. It displays the distribution of a continuous variable, showing its probability density across different levels of a categorical variable. The plot consists of a vertical axis representing the variable of interest and horizontal axes that group data by categorical variables. The shape of the "violin" represents the distribution's density, often mirrored on both sides, making it easier to see the spread and skewness of the data.

```r
ggplot(frs_data, aes(x = marital_status, y = hh_income_gross, fill = marital_status)) +
 geom_violin() +
 facet_wrap(~ sex) +
 scale_y_continuous(
  limits = c(0, 100000),
  labels = label_comma() # Properly placed within scale_y_continuous()
 ) +
 labs(
  title = "Gross Household Income by Marital Status and Sex",
  x = "Marital Status",
  y = "Gross Household Income"
 ) + # Removed 'fill' label to avoid legend creation
 theme_minimal() +
 theme(
  axis.text.x = element_text(angle = 45, hjust = 1), # Rotate x-axis labels for readability
  legend.position = "none" # Corrected to lowercase 'none'
 )
```

```
Warning: Removed 2769 rows containing non-finite outside the scale range
(`stat_ydensity()`).
```

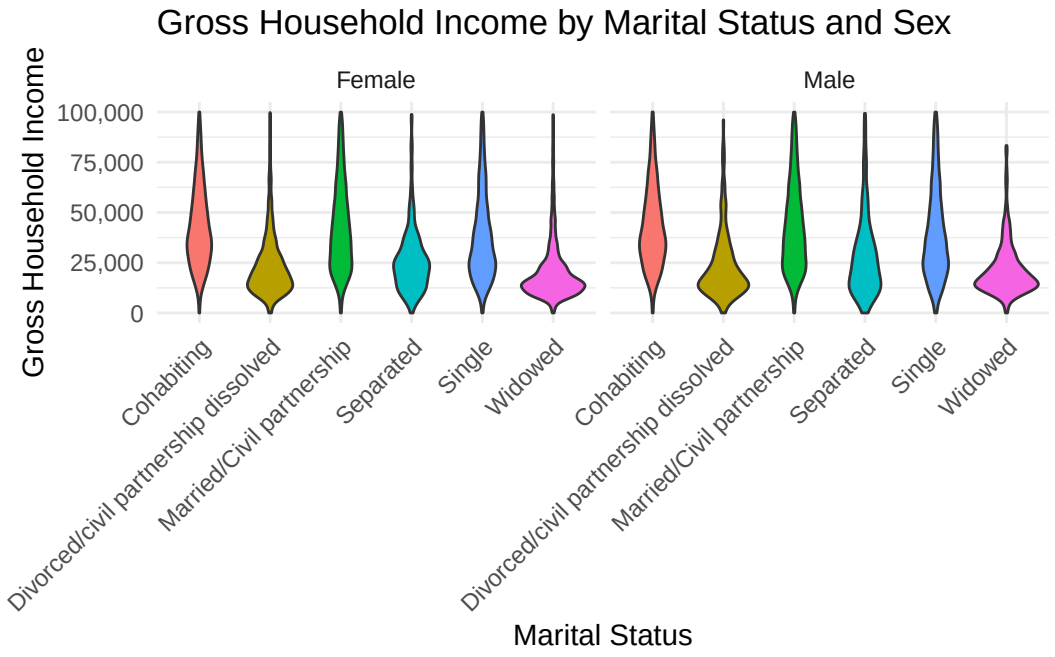


Table 6.6: Summary Statistics

Variable	N	Mean	Std. Dev.	Min	Pctl. 25	Pctl. 75	Max
pct_Very_bad_health	331	1.2	0.34	0.55	0.92	1.4	2.4
pct_No_qualifications	331	18	4	6.6	15	20	29
pct_Males	331	49	0.66	47	49	49	55
pct_Higher_manager_prof	331	13	4.7	5.5	9.8	16	40

6.4 Part 4: Publication-Ready Tables

We will be using `KableExtra` to create nicely formatted tables for a series of ouputs in Rstudio:

```
library(kableExtra)
```

```
Warning: package 'kableExtra' was built under R version 4.4.2
```

```
Attaching package: 'kableExtra'
```

```
The following object is masked from 'package:dplyr':

    group_rows
```

6.4.1 Summarising datasets

Only for summarising statistics we use `vtable` that makes use of `kableExtra` behind the scenes. You can use the function `st()` or `sumtable()`, they do the same thing. See https://cran.r-project.org/web/packages/vtable/vignettes/sumtable.html

```
library(vtable)
```

```
Warning: package 'vtable' was built under R version 4.4.2
```

```
# Generate the summary table
st(census_data, vars = c("pct_Very_bad_health", "pct_No_qualifications", "pct_Males", "pct_Hi
```

6.4.2 Creating a Well-Formatted Table from a Cross Tabulation

We can also convert a basic cross tabulation to a nice table

```
# cross tabulation
cross_tab <- table(frs_data$health, frs_data$highest_qual)

# Convert the cross-tabulation to a data frame
cross_tab_df <- as.data.frame(cross_tab)
```

This will output lots of rows, but it's just an example.

```
colnames(cross_tab_df) <- c("Health", "Qualification", "Percentage")

cross_tab_df %>%
 kbl(caption = "Health and Educational Qualification") %>%
 kable_styling(full_width = FALSE, position = "left") %>%
 add_header_above(c(" " = 1, "Highest Educational Qualification" = 2))
```

- **Rename Columns**: `colnames(cross_tab_df) <- c("Health", "Qualification", "Percentage")`. This assigns new column names to the data frame `cross_tab_df`.
- **Generate a Table with `kable`**: `cross_tab_df %>% kbl(caption = "Health and Educational Qualification")`. This uses the `kbl` function from the `kableExtra` package to create a basic, well-formatted table. The `caption` argument adds a title to the table: "Health and Educational Qualification.
- **Style the Table**: `%>% kable_styling(full_width = FALSE, position = "left")`. This enhances the appearance of the table with the `kable_styling` function. `full_width = FALSE` ensures the table is not stretched to the full width of the document or webpage; `position = "left"`: Aligns the table to the left side of the page.
- **Add a Header Row**: `%>% add_header_above(c(" " = 1, "Highest Educational Qualification" = 2))` Adds an additional header row above the table; `" " = 1` the first column ("Health") has a blank header in this new row; `"Highest Educational Qualification" = 2`: The next two columns ("Qualification" and "Percentage") are grouped under the label "Highest Educational Qualification.

6.4.3 Creating a Well-Formatted Table from a Cross Tabulation

Finally, we can derive a table to show the Multiple Linear Regression results.

Table 6.7: Health and Educational Qualification

	Highest Educational Qualification	
Health	Qualification	Percentage
Bad	A-level or equivalent	200
Fair	A-level or equivalent	868
Good	A-level or equivalent	1626
Not known	A-level or equivalent	1163
Very Bad	A-level or equivalent	55
Very Good	A-level or equivalent	1348
Bad	Degree or above	262
Fair	Degree or above	1097
Good	Degree or above	2953
Not known	Degree or above	1923
Very Bad	Degree or above	67
Very Good	Degree or above	2854
Bad	Dependent child	22
Fair	Dependent child	67
Good	Dependent child	342
Not known	Dependent child	9100
Very Bad	Dependent child	7
Very Good	Dependent child	760
Bad	GCSE or equivalent	524
Fair	GCSE or equivalent	1785
Good	GCSE or equivalent	3061
Not known	GCSE or equivalent	2165
Very Bad	GCSE or equivalent	133
Very Good	GCSE or equivalent	2061
Bad	Not known	841
Fair	Not known	1804
Good	Not known	1674
Not known	Not known	1447
Very Bad	Not known	279
Very Good	Not known	775
Bad	Other	231
Fair	Other	696
Good	Other	854
Not known	Other	587
Very Bad	Other	64
Very Good	Other	450

Table 6.8: Regression Results: Predicting Very Bad Health Percentage

	Coefficients			
Term	Estimate	Std. Error	t Value	P Value
(Intercept)	4.003	0.880	4.550	0.000
pct_No_qualifications	0.053	0.006	8.937	0.000
pct_Males	-0.074	0.018	-4.121	0.000
pct_Higher_manager_prof	-0.013	0.005	-2.670	0.008

```
# Load required libraries
library(broom)

# Fit the regression model
model <- lm(pct_Very_bad_health ~ pct_No_qualifications + pct_Males + pct_Higher_manager_pro
      data = census_data)
```

Let's tidy up the regression output.

```
regression_table <- tidy(model) %>%
 select(term, estimate, std.error, statistic, p.value) %>%
 rename(
  Term = term,
  Estimate = estimate,
  `Std. Error` = std.error,
  `t value` = statistic,
  `P value` = p.value
 )
```

Create and style the regression table

```
regression_table %>%
 kbl(
  caption = "Regression Results: Predicting Very Bad Health Percentage",
  digits = 3,
  col.names = c("Term", "Estimate", "Std. Error", "t Value", "P Value")
 ) %>%
 kable_styling(full_width = FALSE, bootstrap_options = c("striped", "hover", "condensed")) %
 column_spec(2:5, width = "3cm") %>% # Adjust column widths
 add_header_above(c(" " = 1, "Coefficients" = 4)) # Add grouped header
```

- `kable_styling()` This function customizes the appearance of a **`kable`** table. The arguments used are:
  - `full_width = FALSE`: Ensures the table is not stretched to fill the entire width of the page.
  - `bootstrap_options = c("striped", "hover", "condensed")`:
    - `striped`: Adds alternating row colors for better readability.
    - `hover`: Highlights rows when hovered over with the mouse (for HTML tables).
    - `condensed`: Reduces the table's vertical spacing, making it more compact.
- `column_spec(2:5, width = "3cm")`. This function customizes specific columns of the table.
  - `2:5`: Targets columns 2 through 5 for styling.
  - `width = "3cm"`: Sets the width of these columns to 3 cm, ensuring uniform and readable column sizes.
- `add_header_above(c(" " = 1, "Coefficients" = 4))`. This function adds an additional header row above the table. This is used to customize the header of a table.
  - `" " = 1` Leaves the first column (e.g., the Term column) without a header, spanning 1 column.
  - `"Coefficients" = 4`: This part means that columns 2, 3, 4, and 5 will be grouped together under one header, which is labeled as "Coefficients" (4 here indicates the other 4 tables). Essentially, these columns will have the same header, which makes the table more readable and organized.

Have a look here for some details on `kableExtra` https://bookdown.org/yihui/rmarkdown-cookbook/kableextra.html.

6.5 Part 5: Play with the code

- Experiment with different variables and aesthetics to deepen your understanding of ggplot2.
- Test modifying labels, themes, and colour schemes to create tailored visualisations.