LBOMETR Course Book

Jem Marie M. Nario

2025-03-23

Contents

Chapter 1

Introduction

Welcome to the **LBOMETR Course Book**! This book is designed to guide students through the course by providing all necessary resources, materials, and instructions.

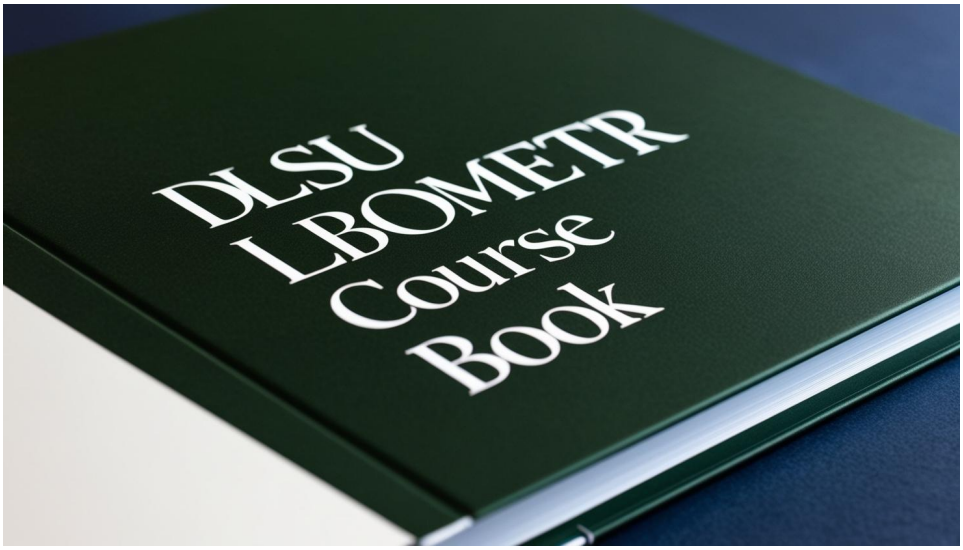


Figure 1.1: LBOMETR

This course book is intended to ensure that DLSU Carlos L. Tiu-School of Economics students will be able to learn more about Econometrics using R. You will find sections on the syllabus, course assessments, and group projects, as well as guidance for navigating the course effectively.

1.1 About Me

My name is **Jem Marie M. Nario**, and I am your lecturer for this course. I am excited to guide you through this journey of learning and discovery since I am also on a journey of learning and discovery while teaching part-time.

This book is a trial version which will be updated along the course as it also serves as a practice for me.

- **Email**: jem.nario@dlsu.edu.ph
- **LinkedIn**: linkedin.com/in/jmnario/

Feel free to reach out with any questions or concerns throughout the course.

Chapter 2

Syllabus

You can download the course syllabus using the link below:

Download Syllabus (Word Document)

2.1 Course Description

This course introduces Economics majors to more advanced commands and techniques used in the econometric software package **R**, which is commonly used in empirical research.

2.2 Learning Outcomes

2.2.1 Knowledge

- To be able to distinguish a theoretical economic model from a statistical econometric model.

- To be able to use the R software package in estimating advanced econometric models.
- To learn advanced econometric models so that students can learn new methods of research.

2.2.2 Skills

- Apply numerical and statistical techniques in economic analysis.
- Use statistical concepts as a language in economic discourse.
- Confidently write script files for economic analysis.

2.2.3 Behavior/Attitude

- To imbibe in the student the need for transparency and academic integrity when handling data analysis.
- To allow the student to learn to construct more complex programs from basic commands learned in class.

2.3 Grading

2.3.1 Grade Components

Component	Weight (%)
Attendance	5%
Group Participation	10%
Data Story Presentation	35%
Data Story Archive	50%

Component	Weight (%)
Total	100%

2.3.2 Grade Scale

Percentage Range	Grade
96 - 100	4.0
90 - 95.99	3.5
84 - 89.99	3.0
78 - 83.99	2.5
72 - 77.99	2.0
66 - 71.99	1.5
60 - 65.99	1.0

Chapter 3

Course Assessments

3.1 Data Story Archive

The **Data Story Archive** is the culmination of your group's work throughout the course. It includes your group's data story report, R script, practical assignments, and a group reflection, all compiled into a single professionally formatted PDF file.

3.1.1 Requirements

Your submission should follow this structure:

1. **Cover Page**:
    - Include the title of the Data Story, group members, and submission date.
2. **Table of Contents**:

- Provide a clear list of sections with page numbers.

3. **Data Story Report**:

   - The complete report should include:
     - **Introduction**: Problem statement and research question.
     - **Methods**: Data sources, methodology, and analysis techniques.
     - **Results**: Key findings supported by R-generated visuals.
     - **Discussion**: Implications of the findings and any limitations.
     - **Conclusion**: Summary and recommendations.
     - **Appendix**: Supporting tables, additional plots, or materials.

4. **R Script**:

   - Render your R script as an **HTML** using Quarto Markdown.
   - Ensure the script is well-structured, commented, and includes outputs like plots and tables.

5. **Computer Practicals**:

   - Include PDFs of all Quarto Markdown files from your computer practicals by printing the html as pdf.

6. **Group Reflection**:

   - Write a 1-2 page reflection on:
     - Your teamwork experience (challenges and successes).
     - What you learned from working on the data story.

- How the course contributed to your growth in data analysis and collaboration.

3.1.2 Submission

- Combine all the components into a **single PDF** file.
- Name your file as: `LBOMETR[Section_GroupNo.]_DataStoryArchive.pdf`
- **Deadline**: [11 April 2025, 21:00].
- In the event that the file is too big for Animospace, kindly submit as pdf to my email.

3.1.3 Grading Rubric for Data Story Archive

The grading rubric for the Data Story Archive is divided into three categories: **Content**, **Analysis and Technical Work**, and **Overall Presentation Quality**.

Category	Criteria	Points	Description
1. Content			

Category	Criteria	Points	Description
	Clarity of Objective	10	Clearly defined problem/question and its relevance to the course.
	Data Story Report	20	Completeness and quality of the report, including introduction, methods, results, and discussion.
	Appendix	10	Completeness of additional materials (e.g., tables, plots) in the appendix.
2. Analysis and Technical Work			

Category	Criteria	Points	Description
	R Script Quality	15	Well-structured, commented, and reproducible R script with outputs rendered as a PDF.
	Practical Assignments	15	Quality and completeness of PDFs rendered from Quarto Markdown files.
	Visualizations	15	Clear, meaningful, and well-designed plots and tables generated in R.
3. Overall Presentation Quality			

Category	Criteria	Points	Description
	Group Reflection	15	Thoughtful insights on teamwork, learning, and course experience.
	Formatting and Organization	10	Overall organization, formatting, and adherence to submission guidelines.
	Total	**100**	

3.2 Data Story Presentation

The **Data Story Presentation** is your group's opportunity to communicate your findings and insights through a live presentation. This format allows you to showcase animated visualizations and engage directly with the audience in real time. A room will be requested for you to be able to present in front of your classmates and I will be present online *hopefully this will be applicable;*

3.2.1 Requirements

1. **Objective**:
    - Your live presentation should effectively communicate your data story with clarity, engagement, and professionalism, making full use of visuals and animations to enhance understanding.
2. **Presentation Structure**: The presentation must include the following sections:
    - **Introduction**: Briefly introduce your topic, research question, and the significance of your data story (1 slide).
    - **Methods**: Provide a concise explanation of your data and analysis methodology (1-2 slides).
    - **Results**: Highlight the most important findings using R-generated visualizations, including animations if applicable (3-4 slides).
    - **Discussion and Conclusion**: Discuss the implications of your findings and conclude with actionable insights or recommendations (1 slide).
3. **Delivery**:
    - Each group member must actively participate in the presentation.
    - Presentation duration: **10 minutes**, followed by a **5-minute Q&A session**.
4. **Visualizations**:
    - Use animated or interactive visualizations (e.g., created with `gganimate` or other R packages) to effectively demonstrate key

trends and insights.

- Ensure visuals are clear, professional, and aligned with your narrative.

5. **Tools**:

   - Create your presentation using tools like Google Slides, Microsoft PowerPoint, or Canva.
   - Incorporate animated visualizations as needed.

6. **Submission**:

   - Submit your presentation slides as a **PDF file** named:

     `LBOMETR[Section_GroupNo.]_DataStoryPresentation.pdf`

   - Submit the file before your scheduled presentation time.

3.2.2 Grading Rubric

The grading rubric for the Data Story Presentation is divided into three categories: **Content**, **Visualizations**, and **Delivery and Engagement**.

Category	Criteria	Points	Description
1. Content			

Category	Criteria	Points	Description
	Introduction and Methods	10	Clear and concise introduction and explanation of methods.
	Results	20	Logical flow and depth of results, focusing on key findings.
	Discussion and Conclusion	10	Insightful discussion and actionable conclusion.
2. Visualizations			
	Quality of Visuals	20	Professional and well-designed visualizations, including appropriate use of animations.

Category	Criteria	Points	Description
	Relevance of Visuals	10	Visuals strongly support the analysis and enhance understanding.
3. Delivery and Engagement			
	Delivery	20	Confident, clear, and professional delivery by all group members.
	Audience Engagement	10	Creativity and ability to maintain audience attention.
	Q&A Session	10	Ability to effectively respond to audience questions.

Category	Criteria	Points	Description
	Time Management	10	Adherence to the 10-minute time limit and logical pacing.
	Total	**100**	

Chapter 4

Grouping Process

Students will be randomly assigned to groups of **4-5 members** based on their responses to a pre-course survey. The survey collects information that will be used to ensure fair and balanced groupings. The group assignments will be announced on the first day of the course.

4.1 Survey

Please complete the survey **before 14:30 PM on January 6, 2025** using the link:

- Google Form Survey Link

4.2 How Groups Are Formed

The groupings are created using RStudio. The coding ensures randomness while incorporating some aspects of the survey responses to balance groups.

If you wish to see the code used for grouping, you may contact me directly. However, please note: - The **CSV file with survey responses will not be shared** to protect your anonymity and privacy.

4.3 Announcement of Groups

The group assignments will be distributed on the **first day of the course**. Please check your assigned group and connect with your group members as soon as possible.

Chapter 5

Data Story Research Question

5.1 Guidelines in Conducting Data Story

5.1.1 Research Question:

In writing your Research Question, you should keep in mind the following acronym:

S-M-A-R-T

S-**S**pecific

M-Measurable

A-Achievable

R-Relevant

T-Time-bound

What do you think of this research question:

"*What is the effect of X(Twitter) on mental health?*"

Do you think this question is SMART?

How about this question:

"*What is the relationship between hours spent per day on X/Twitter and reported levels of anxiety among undergraduate students in DLSU during 2020-2021?*"

Which of the two achieves the SMART elements?

5.1.2 Relevance

What will be asked by the audience?

You need to ask yourselves the following questions; if you are able to, you are close to achieving a good presentation!

1. What is the problem to be solved?
2. Who cares about this problem and why?

5.1.3 For the data story, should I use Kaggle?

The answer is **NO** because of the following reasons:

1. It has already been cleaned and curated based on the owner's own purposes. It may not accurately reflect the real-world data.
2. It has limited context, potentially limiting your ability to interpret findings accurately.

3. While some Kaggle datasets have clear sources, others may have unclear origins or have undergone modifications by multiple users raising concerns about data integrity.

Thus, you should use credible sources like statistics from World Bank, government sites, etc. I am not saying that you can already drop Kaggle datasets. You can still use them for the following:

1. Benchmarking: Compare your model on a well-known Kaggle dataset
2. For learning: You can try to match how the datasets in Kaggle have been cleaned. You can also use the datasets to practice your codes before applying them to the data you found from other credible sources.

Important to note: Check data documentation from Kaggle; this will help you find the sources and will help supplement your data story with reputable sources.

Furthermore, this is LBOMETR class wherein you were taught how to clean a dataset. You have to gather and clean the data yourselves to align with your research question.

5.2 Comments on Research Questions:

Please check your emails for comments regarding your research questions. The approved and final research questions will be posted in Animospace as an announcement.

Chapter 6

Basic Introduction to R

This portion of the book offers an introduction to the basics of R. R offers a wide variety of functionality. Note that this book only offers basic Econometric analysis. It will be useful to have some basic familiarity with R and its syntax but this is not strictly necessary.

Each chapter includes both R code and results to make it easier for students to follow along, even without detailed knowledge of R.

6.1 Session Information

This version of the book was built using R version 4.4.2. See below for the session information:

```
## R version 4.4.2 (2024-10-31 ucrt)
## Platform: x86_64-w64-mingw32/x64
## Running under: Windows 11 x64 (build 26100)
```

```
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=English_Netherlands.utf8  LC_CTYPE=English_Netherlands.utf8
## [4] LC_NUMERIC=C                         LC_TIME=English_Netherlands.utf8
##
## time zone: Europe/Berlin
## tzcode source: internal
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] samplingbook_1.2.4  survey_4.4-2        survival_3.7-0      Matrix_1.7-1
##  [6] pps_1.0             sandwich_3.1-1      car_3.1-3           carData_3.0-5
## [11] lmtest_0.9-40       gtsummary_2.0.4     psych_2.5.3         av_0.9.4
## [16] leaflet_2.2.2       WDI_2.7.8           gifski_1.32.0-1     rnaturalearth
## [21] gganimate_1.0.9     readxl_1.4.3        bookdown_0.41       wooldridge_1
## [26] zoo_1.8-12          forecast_8.23.0     lubridate_1.9.4     forcats_1.0.0
## [31] dplyr_1.1.4         purrr_1.0.2         readr_2.1.5         tidyr_1.3.1
## [36] ggplot2_3.5.1       tidyverse_2.0.0     plm_2.6-5
##
## loaded via a namespace (and not attached):
##  [1] rstudioapi_0.17.1       jsonlite_1.8.9          wk_0.9.4                r
##  [5] farver_2.1.2            rmarkdown_2.29          vctrs_0.6.5             t
```

```
##  [9] tinytex_0.54          htmltools_0.5.8.1     progress_1.2.3        curl_6.0.
## [13] cellranger_1.1.0      s2_1.1.7              Formula_1.2-5         TTR_0.24.
## [17] sass_0.4.9            KernSmooth_2.23-24    bslib_0.8.0           htmlwidge
## [21] cachem_1.1.0          gt_0.11.1             commonmark_1.9.2      lifecycle
## [25] pkgconfig_2.0.3       R6_2.5.1              fastmap_1.2.0         rbibutils
## [29] collapse_2.1.0        digest_0.6.37         colorspace_2.1-1      miscTools
## [33] crosstalk_1.2.1       labeling_0.4.3        timechange_0.3.0      httr_1.4.
## [37] abind_1.4-8           mgcv_1.9-1            compiler_4.4.2        proxy_0.4
## [41] withr_3.0.2           DBI_1.2.3             MASS_7.3-61           classInt_
## [45] tools_4.4.2           units_0.8-5           quantmod_0.4.26       nnet_7.3-
## [49] glue_1.8.0            rnaturalearthdata_1.0.0 quadprog_1.5-8      nlme_3.1-
## [53] generics_0.1.3        lpSolve_5.6.23        gtable_0.3.6          tzdb_0.4.
## [57] class_7.3-22          hms_1.1.3             xml2_1.3.6            utf8_1.2.
## [61] pillar_1.10.0         markdown_1.13         mitools_2.4           splines_4
## [65] tweenr_2.0.3          lattice_0.22-6        tidyselect_1.2.1      knitr_1.4
## [69] urca_1.3-4            xfun_0.49             timeDate_4041.110     stringi_1
## [73] yaml_2.3.10           evaluate_1.0.1        codetools_0.2-20      cli_3.6.3
## [77] Rdpack_2.6.2          munsell_0.5.1         jquerylib_0.1.4       Rcpp_1.0.
## [81] bdsmatrix_1.3-7       parallel_4.4.2        fracdiff_1.5-3        prettyuni
## [85] viridisLite_0.4.2     xts_0.14.1            e1071_1.7-16          crayon_1.
## [89] maxLik_1.5-2.1        rlang_1.1.4           mnormt_2.1.1          cards_0.4
```

6.2 Preliminaries

The first step is to gain access to R, which is free and available on the R website: http://cran.r-project.org/. Simply go to the R website, select the

appropriate location and operating system, and follow the instructions to download the base distribution of R. RStudio offers a user friendly environment to run R and is recommended.

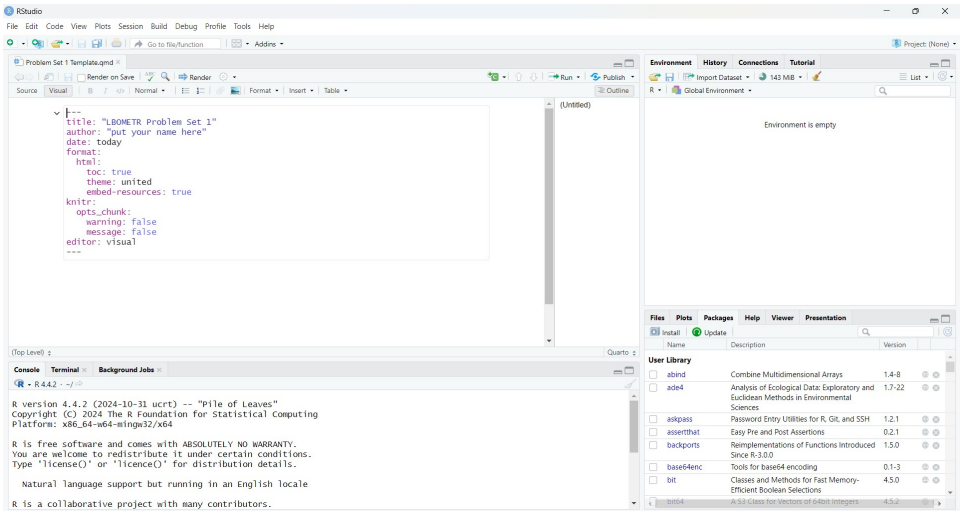


Figure 6.1: RStudio Screen

Once R is opened, we can begin to run commands. R commands can be run directly from the console, from the R script editor or from a text editor separate from R.

R offers detailed help files for each function. To access help, run:

```
?sum
```

All lines proceeded by a # are comments and will not run. For example:

```
# This is a comment. R will not recognize this as a command.
```

6.3 Quarto Markdown

In LBOMETR, Quarto Markdown will be used by the students when submitting the Scripts for the Data Story Archive. Quarto Markdown is a tool for creating documents, reports and presentations using Markdown and executable code. Below is a concise guide to help you get started, along with key shortcuts for both Mac and Windows.

6.3.1 1. Starting a Quarto File

To begin creating a Quarto document, follow these steps:

1. Open RStudio.
2. Go to **File > New File > Quarto Document.**
3. Choose the document type (e.g., HTML, PDF, Word, etc.) and specify whether the document will include code. For ease, we will use the html document type. I have also added a sample Quarto Markdown file you can copy.

   Quarto Markdown Template

6.3.2 2. Quarto Key Features

Code Chunks

Code chunks allow you to include and run code inside your document.

Inline Code

Embed R code in text using backticks and `r` .

6.3.3 Quarto Markdown Shortcuts

Action	Windows Shortcut	Mac Shortcut
Insert a new code chunk	Ctrl+Alt+I	Cmd+Alt+I
Run current code chunk	Ctrl+Shift+Enter	Cmd+Shift+Enter
Run all code chunks	Ctrl+Alt+R	Cmd+Alt+R
Run current line/selection	Ctrl+Enter	Cmd+Enter
Knit/Render document	Ctrl+Shift+K	Cmd+Shift+K
Comment/uncomment lines	Ctrl+Shift+C	Cmd+Shift+C
Insert pipe (%>%)	Ctrl+Shift+M	Cmd+Shift+M
Headings	/Number of Heading (if in Visual mode) Prefix line with #, ##, etc. manually (in Source mode)	/Number of Heading (if in Visual mode) Prefix line with #, ##, etc. manually (in Source mode)
Bold	Ctrl+B	Cmd+B
Italic	Ctrl+I	Cmd+I
Inline code	Surround with backticks (') manually	Surround with backticks (') manually

*Note: you can choose between Source or Visual (upper left); personally, it is easier for me to use the Visual Mode compared to the Source Mode.

6.4 Packages

Each package of interest must be installed and loaded before it can be used. The packages will not be immediately available when R is opened. A package only has to be installed once on a computer, but the package will have to be loaded every time R is restarted.

We can install a package individually as we need them. For example, to install **tidyverse** and **psych**, we would do:

```
install.packages("tidyverse")
install.packages("psych")
```

In the tidyverse package, the **ggplot2** is usually included; if you do not see the package in the Packages list at the lower right, you can do this:

```
if(!("ggplot2" %in% installed.packages()[,"Package"])) install.packages("ggplot2")
```

Now that we have our packages successfully installed, we can go ahead and load them into R. Here we will load the tidyverse package as an example. We can use of all the functions available in that package once it is loaded into R. We load packages by using a `library()` function. The input is the name of the package, not in quotes.

```
library(tidyverse)
```

We can look up all of the functions within a package by using a `help()` function. For example, let's look at the functions available in the **tidyverse** package.

```
help(package = tidyverse)
```

Note that the package argument is necessary to look up all of the functions. We can also detach a package if we no longer want it loaded. This is sometimes useful if two packages do not play well together. Here we will use a `detach()` function.

```
detach(package:tidyverse)
```

For simplicity, we will assume that the reader has restarted R at the beginning of each tutorial.

6.5 Instructions for Managing Working Directories

This guide outlines how team members should set up their local working directories for collaboration, handle `.qmd` files, and organize them in a shared Google Drive.

6.5.1 1. Local Working Directory Setup

Each team member should create a local folder on their own laptops to work on `qmd` files. This folder is where you will store and edit your files before uploading them to the shared Google Drive.

6.5.1.1 Steps:

1. Create a folder on your laptop named: **DLSU_LBOMETR_Section**
2. Use this folder to save and organize your `.qmd` files while working locally.

6.5.2 2. File Naming Convention

To avoid confusion, ensure all `.qmd` files are named as follows:

- Include your name or initials and a brief description of the content
- Example:
  - `jem_nario_descriptivestatistics.qmd`
  - `jmn_piechart.qmd`

6.5.3 3. Shared Google Drive Setup

A **shared Google Drive** will serve as the central repository for all project files, including:

- `.qmd` files from all team members

- Data files
- Rendered HTML and PDF files for final submission
- Supporting documents or references.

6.5.4 4. Workflow for `.qmd` files

For each team member:

1. Work locally
    - Create your `.qmd` file in your local `DLSU_LBOMETR_Section` folder
    - Ensure it is well-documented and organized.
2. Upload to Google Drive

For the Team Leader:

1. Collect and Combine Files
    - Gather all `.qmd` files from the team folder on the shared drive.
    - Combine them
2. Render the final report

6.5.5 5. Rendering the Final Report

The final report should be rendered in HTML and printed by the team leader.

6.5.6 Summary Workflow

- **Each Team Member**:
  - Work on your `.qmd` file locally.
  - Upload your file to the shared Google Drive under `team-members-qmd`.
- **Team Leader**:
  - Collect `.qmd` files from the shared drive.
  - Combine them into a single `final_report.qmd`.
  - Render the final report into HTML and PDF.
  - Upload the rendered files to the `final-report` folder on the shared Google Drive.

This ensures an organized and efficient workflow while centralizing all files in the shared Google Drive for easy access and submission.

Chapter 7

Data Management - Cross-Sectional Data

7.1 Where to Get Data?

Before we proceed to Data Management, let us first find where we can get data for the Data Story Archive. Note that the data you collect should still ensure that you are following the Code of Ethics and analyze Ethical Considerations.

Please view the necessary documents from the Office of the Vice Chancellor for Research and Innovation (https://www.dlsu.edu.ph/research/research-manual/)

A list of links you can search and get data from:

Note: I will not include the best links as they are pretty straightforward and these are governmental databases like the ones from World Bank, IMF,

UN, Philippine Statistics Authority, and Bangko Sentral ng Pilipinas. The list here is a general list but use with proper discretion.

Name	Link	Notes
Kaggle	https://www.kaggle.com/datasets	Kaggle is where users can provide datasets; it is important to cite the sources. Mostly, datasets in Kaggle can be used for your practice.
Awesome Public Datasets	https://github.com/awesomedata/awesome-public-datasets	This repository is filled with public datasets, mostly from International contexts.
Google Dataset Search	https://datasetsearch.research.google.com/	You can download publicly available datasets from searching through Google. Though, sometimes the datasets come from 'Statista.com'. You can check the sources from the search.

7.2 Preliminaries

7.2.1 Dataset

The dataset to be used can be downloaded here: Chapter2 Practice and will be included in the Files in Animospace. The dataset was modified from the Wooldridge package in R as practice material. The particular dataset is the 'htv' dataset.

```
#install the wooldridge package. Check previous chapter on how to install packages.
load(wooldridge)
?htv #to find out about the particular dataset.
```

NOTE: The htv dataset help will tell you what the variables mean, however, for our practice, we will use the modified version of this dataset.

7.2.2 Packages

We will mostly use the `tidyverse` package, in particular, the `dplyr` package and the `tidyr` package; double-check in your Packages list whether you have these two packages; if not, you can simply install them.

7.2.3 Setting up the Directory

This is the most important step! Make sure to place the downloaded file in this folder: **DLSU_LBOMETR_Section** in your laptops. Remember, this is your local working directory. This is the working directory you choose. You can set up your working directory in the following ways:

1. Using the R Studio Menu (works for both Mac and Windows)

   a. Go to **Session > Set Working Directory > Choose Working Directory**

2. Windows:

```
# Use double backslashes `\\` or forward slashes `/`
setwd("C:\\Users\\YourUsername\\Documents")  # Example with backslashes
setwd("C:/Users/YourUsername/Documents")     # Example with forward slashes
```

3. Mac

```
# Use forward slashes `/`
setwd("/Users/YourUsername/Documents")
```

To check:

```
getwd()
```

7.2.4 Clean Everything

Do this step every time you use other data or when we do the other chapters.

```
# Remove all objects in the global environment
rm(list = ls())


# Perform garbage collection to free up memory
gc()
```

```
##            used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells  5655682 302.1    9801260 523.5  9801260 523.5
## Vcells 11542241  88.1   36965975 282.1 94237569 719.0
```

7.2.5 Importing the Dataset

We can use the `read.csv()` to load the `csv` file into R. Always call the file as something short and easily understandable. Ensure the downloaded file is in the working directory before you load the file. If the downloaded file is not located in the working directory, you will encounter issues.

I will name the file as `ch2_p1`

```
ch2_p1<-read.csv("Ch2Practice.csv")
```

We can use the `head()` function to inspect the first six rows of the dataset:

```
head(ch2_p1)
```

```
##        WAGE  ABILITY EDUCATION NORTHEAST NORTHCENTRAL WEST SOUTH EXPERIENCE MOTHEREDUC
## 1 12.019231 5.027738        15        no           no  yes    no          9         12
## 2  8.912656 2.037170        13       yes           no   no    no          8         12
## 3 15.514334 2.475895        15       yes           no   no    no         11         12
## 4 13.333333 3.609240        15       yes           no   no    no          6         12
## 5 11.070110 2.636546        13       yes           no   no    no         15         12
## 6 17.482517 3.474334        18       yes           no   no    no          8         12
##   URBAN X18INNORTHEAST X18INNORTHCENTRAL X18INSOUTH X18INWEST X18INURBAN X17TUITION X18
## 1   yes              1                 0          0         0          1   7.582914   7
## 2   yes              1                 0          0         0          1   8.595144   9
```

```
## 3   yes              1                0          0          0          1   7.3
## 4   yes              1                0          0          0          1   9.4
## 5   yes              1                0          0          0          1   7.3
## 6   yes              1                0          0          0          1   7.3
```

7.3 Data Cleaning

As you can see, there are 22 columns. Let's simplify by only choosing the following: WAGE, URBAN, X17TUITION, X18TUITION and EXPER.2. We can do this using the `select()` function in `dplyr`. We will save them into a new data frame, `ch2_p1.1`.

```
library(dplyr)
```

```
ch2_p1.1<-select(ch2_p1, #the original dataset
       WAGE, URBAN, X17TUITION, X18TUITION, EXPER.2)
```

7.3.1 Renaming the Variables

We will edit the names to much easier conventions. First, let us say that we just want to change them to lowercase names.

```
names(ch2_p1.1)<-tolower(names(ch2_p1.1))
```

Inspect:

```
head(ch2_p1.1)
```

```
##        wage urban x17tuition x18tuition exper.2
## 1 12.019231   yes   7.582914   7.260242      81
## 2  8.912656   yes   8.595144   9.499537      64
## 3 15.514334   yes   7.311346   7.311346     121
## 4 13.333333   yes   9.499537  10.162070      36
## 5 11.070110   yes   7.311346   7.311346     225
## 6 17.482517   yes   7.311346   7.311346      64
```

Let us change the names of `x17tuition, x18tuition,`and`exper.2` to the names similar to what is found in the 'htv' dataset: `x17tuition` to `tuit17`, `x18tuition` to `tuit18` and `exper.2` to `expersq` . To do this, we will use the `rename()` in `dplyr`. We will also use the (`%>%`) for this.

```
ch2_p1.1<-ch2_p1.1 %>%
  rename(
    tuit17 = x17tuition,
    tuit18 = x18tuition,
    expersq = exper.2
  )
```

Inspect again:

```
head(ch2_p1.1)
```

```
##        wage urban   tuit17    tuit18 expersq
```

```
## 1 12.019231   yes 7.582914  7.260242      81
## 2  8.912656   yes 8.595144  9.499537      64
## 3 15.514334   yes 7.311346  7.311346     121
## 4 13.333333   yes 9.499537 10.162070      36
## 5 11.070110   yes 7.311346  7.311346     225
## 6 17.482517   yes 7.311346  7.311346      64
```

7.3.2 Sorting certain values

Let's say, we want to arrange `wage`. We will create a different data for this. We use `arrange` in `dplyr` package.

```
ch2_p1sort<-arrange(ch2_p1.1,
                    wage)
```

Inspect:

```
head(ch2_p1sort)
```

```
##       wage urban    tuit17    tuit18 expersq
## 1 1.023529   yes  2.088957  2.251239     169
## 2 1.073345    no  7.520245  7.520245     196
## 3 1.102362   yes  7.355460  6.922371     196
## 4 1.250000   yes  9.417682  8.826549     100
## 5 1.373626   yes  9.692757  9.692757     225
## 6 1.442308    no 11.280367 11.280367      NA
```

What if you have this kind of data?

```
head(ch2_p2)
```

```
##    ch2_p2
## 1 $10,000
## 2 $20,500
## 3 $15,250
## 4 $30,000
## 5 $50,750
```

Let's sort this:

```
ch2_p2sort<-arrange(ch2_p2)
head(ch2_p2)
```

```
##    ch2_p2
## 1 $10,000
## 2 $20,500
## 3 $15,250
## 4 $30,000
## 5 $50,750
```

It did not work. The problem is, `ch2_p2` is not numeric. We can check:

```
class(ch2_p2$ch2_p2)
```

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

We need to make it into a numeric value but we have a , and $. We need to remove them. We use the `str_replace` function in the `stringr` package.

```
library(stringr)
```

```
ch2_p2$ch2_p2<-str_replace(
  ch2_p2$ch2_p2, #column we want to edit
  pattern = ',', #what to find
  replacement = '' #what to replace it with
)
```

```
head(ch2_p2)
```

```
##   ch2_p2
## 1 $10000
## 2 $20500
## 3 $15250
## 4 $30000
## 5 $50750
```

Now, let us remove the dollar sign; usually, simply doing the same thing we did with the comma works, but, there are some symbols that are used as "special character". To "force" R to replace the presence of '$', we add two backslashes before the dollar sign.

```
ch2_p2$ch2_p2<-str_replace(
  ch2_p2$ch2_p2,
  pattern = '\\$',
```

```
  replacement = ''
)
```

Can you inspect it on your own?

Simply type the code in the empty code chunk then run it by pressing **Ctrl+Enter** or **Cmd+Enter**

Now, sort `ch2_p2`

```
ch2_p2sort<-arrange(
  ch2_p2,
  ch2_p2
)
```

```
head(ch2_p2sort)
```

```
##   ch2_p2
## 1  10000
## 2  15250
## 3  20500
## 4  30000
## 5  50750
```

We can see that it was arranged, however, take a look at the way ch2_p2 was encoded; it is not numeric. So, we need to change this.

```
class(ch2_p2$ch2_p2)
```

```
## [1] "character"
```

Change to numeric through `as.numeric()`

```
ch2_p2$ch2_p2<-as.numeric(ch2_p2$ch2_p2)
```

Inspect on your own:

7.3.3 Pipe Operator

`%>%` allows functions to be chained; it can be read as "then" - it tells R to do whatever comes after it to the stuff that comes before it.

7.3.4 Adding columns

We will be using the pipe operator and the `mutate` to add a new column to `ch2_p1.1` based from details found in `ch2_p1`, particularly, NORTHEAST, NORTHCENTRAL, WEST, and SOUTH. We will call this new column as `location`

```
ch2_p1.1<-ch2_p1.1 %>%
  mutate(
    location = case_when( #creates conditional statements
      ch2_p1$NORTHEAST == "yes" ~ "northeast", #If NE is "yes", location is "nor
    ch2_p1$WEST == "yes"~"west", #If WEST is "yes", location is "west"
    ch2_p1$NORTHCENTRAL == "yes"~ "northcen",
```

```
      ch2_p1$SOUTH == "yes"~ "south",
    TRUE~"other")
  )
```

Inspect the data:

Now I want you to create a new column called, `tuit_diff` wherein it is the difference between `tuit18` and `tuit17`. In this case, there is no need to use `case_when` since there is no conditional statements to be used. It is straightforward that you simply need to subtract `tuit17` from `tuit18`. You will need to use `mutate` still. How will you create that?

7.3.5 Transforming values

Now, you can see that urban is a `character` that is "yes/no". We need to change that to numeric value. This is particularly useful when we use dummy variables later on. We will not use `case_when` as it is not necessary; rather, we will use `ifelse`:

```
ch2_p1.1<-ch2_p1.1 %>%
  mutate(
    urban = ifelse(urban=="yes", 1,0) #replace "yes" with 1 and "no" with 0
  )
head(ch2_p1.1) #default is first 6 rows
```

```
##        wage urban   tuit17    tuit18 expersq  location
## 1 12.019231     1 7.582914  7.260242      81      west
## 2  8.912656     1 8.595144  9.499537      64 northeast
```

```
## 3 15.514334     1 7.311346  7.311346     121 northeast
## 4 13.333333     1 9.499537 10.162070      36 northeast
## 5 11.070110     1 7.311346  7.311346     225 northeast
## 6 17.482517     1 7.311346  7.311346      64 northeast
```

Now, I want you to create groups for `expersq`. `NA` should now be 0, assign 1 if less than 50, assign 2 if between 50 and 100, assign 3 if between 100 and 200, and for the rest, assign 4.

Clue: conditional statements like between 50 and 100 should be like this:

```
values>=50 & values < 100
```

Your answer should look like this:

```
##        wage urban   tuit17    tuit18 expersq  location
## 1 12.019231     1 7.582914  7.260242       2      west
## 2  8.912656     1 8.595144  9.499537       2 northeast
## 3 15.514334     1 7.311346  7.311346       3 northeast
## 4 13.333333     1 9.499537 10.162070       1 northeast
## 5 11.070110     1 7.311346  7.311346       4 northeast
## 6 17.482517     1 7.311346  7.311346       2 northeast
```

7.3.6 Summarizing

Let us get the average of wages by location, which we'll call `ave.wage`, by using the `group_by()` and `summarise()` functions in dplyr

```
ch2_p1.1ave<-ch2_p1.1 %>%
  group_by(location) %>% #group by location, THEN
  summarise(ave.wage=mean(wage)) #calculate the mean of wages for each location
head(ch2_p1.1ave)
```

```
## # A tibble: 4 x 2
##   location  ave.wage
##   <chr>        <dbl>
## 1 northcen      12.5
## 2 northeast     15.0
## 3 south         12.4
## 4 west          14.1
```

Say that you want to see the average wage in the south area. We can do this by using `filter()`

```
ch2_p1.1ave %>% filter(location=="south")
```

```
## # A tibble: 1 x 2
##   location ave.wage
##   <chr>       <dbl>
## 1 south        12.4
```

How would you sort the dataset by average wage, from highest to lowest? Now you see that it is arranged alphabetically, so how will you arrange it?

7.3.7 Merging datasets

We have two main datasets, `ch2_p1.1` and `ch2_p1.1ave`. By doing this, we could compare side-by-side each observation compared to the average per location.

We will join the datasets by `location` variable, since that is consistent across both datasets. We name the new file as `ch2_p1merged`:

```
ch2_p1merged<-merge(x=ch2_p1.1, y=ch2_p1.1ave, by="location")
head(ch2_p1merged)
```

```
##   location      wage urban   tuit17   tuit18 expersq ave.wage
## 1 northcen 15.294118     1 8.334936 8.334936       3 12.54078
## 2 northcen 17.006804     1 8.334936 8.334936       0 12.54078
## 3 northcen  3.755868     1 8.334936 8.334936       3 12.54078
## 4 northcen  5.288462     1 6.742574 7.198132       2 12.54078
## 5 northcen  9.072165     1 7.305873 7.356897       0 12.54078
## 6 northcen  9.384164     1 8.334936 8.334936       3 12.54078
```

7.3.8 Splitting datasets

Say I want to save different datasets based on the `location` column.

```
northcen_data<-ch2_p1.1 %>% filter(location=="northcen")
```

You can save it as a `.csv` file:

```
write.csv(northcen_data, "northcentral.csv", row.names = FALSE)
```

Can you do the others?

7.3.9 Dates

We are going to work on dates when we move to the next chapter but, here is something initial and necessary.

```
##   ID date_of_birth
## 1  1    15-05-1990
## 2  2    20-08-1985
## 3  3    01-12-2000
## 4  4    10-03-1995
## 5  5    25-07-2010
```

```
## 'data.frame':    5 obs. of  2 variables:
##  $ ID           : int  1 2 3 4 5
##  $ date_of_birth: chr  "15-05-1990" "20-08-1985" "01-12-2000" "10-03-1995" ...
```

To convert the `character` format to date format, we do this:

```
date_data$date_of_birth <- as.Date(date_data$date_of_birth, format = "%d-%m-%Y")


head(date_data)
```

```
##   ID date_of_birth
## 1  1    1990-05-15
```

```
## 2  2    1985-08-20
## 3  3    2000-12-01
## 4  4    1995-03-10
## 5  5    2010-07-25
```

Say you want to calculate the age:

```
date_data$age <-as.numeric(floor((Sys.Date()-date_data$date_of_birth)/365.25))
head(date_data)
```

```
##   ID date_of_birth age
## 1  1    1990-05-15  34
## 2  2    1985-08-20  39
## 3  3    2000-12-01  24
## 4  4    1995-03-10  30
## 5  5    2010-07-25  14
```

7.3.9.1 Custom Reference Date:

```
ref_date<-as.Date("2020-01-01")
date_data$age2<-as.numeric(floor((ref_date-date_data$date_of_birth)/365.25))
head(date_data)
```

```
##   ID date_of_birth age age2
## 1  1    1990-05-15  34   29
## 2  2    1985-08-20  39   34
## 3  3    2000-12-01  24   19
```

```
## 4  4    1995-03-10  30   24
## 5  5    2010-07-25  14    9
```

7.4 Data Management Practical

In your Quarto Markdown files, you need to answer the following questions. Answers will be given the week after the Practical as a form of Feedback. You can answer by group. It would be great to include which group member did what.

In the first part of the practical, answer the following reflection:

1. Do you think you were able to input correctly all the codes in the empty code chunks? Why do you think so? What did you find difficult?

Now comes the practical proper:

Using the `kpop idols` dataset which you download: Practical 1 - also made available in Animospace, answer the questions.

Ensure that you can render individually before uploading in your shared Google Drive. Failure to render the file means you were unable to do the Practical. The leader must take note of this since accomplishing the practicals are part of your grade in Group Participation and the Data Story Archive.

1. Separate the dataset into two datasets: one for `males` and one for females. Save these datasets as `males.csv` and `females.csv`
2. Edit the column names for both `males` and `females` datasets to make them shorter, easier to understand, and consistent

3. Remove the following columns from both datasets: Instagram, Korean Name, K.Stage Name, Stage Name
4. How many males and females are in the dataset? *Hint:* `nrow()`
5. Create a binary variable `not_seoul` for both datasets.
    1. Assign `1` if `birthplace` is **not Seoul,** and `0` otherwise
    2. Count how many individuals are from Seoul and how many are not for both datasets.
6. How many males are eligible for military service (ages 18-28 as of February 27, 2024)?
    1. Filter the `males` dataset for this age range, how many from the Country of South Korea (only filter according to Country for straightforwardness) and count how many qualify.
7. Assign generations based on age (as of June 29, 2023 when the Korean Age system was abolished) for both datasets.
    1. Generation criteria:
        1. 1st Gen: Age>=40
        2. 2nd Gen: 31<=Age<=39
        3. 3rd Gen: 25<=Age<=30
        4. 4th Gen: Age<=24

        Count the number of individuals in each generation for males and females
8. Create a new column `income` for both datasets using hypothetical values based from your hypothesis on age influencing income levels of idols. Explain first what your hypothesis is - are idols who are older

earning more or less? Why or why not? Also think if you believe females earn more than males or vice versa?

1. Use `set.seed()` for reproducibility and generate random income values.
    1. Please have different income values depending on their age. You can do similar groupings as Step 7 for this.
2. Compare the mean income

9. Combine the `males` and `females` datasets
10. Calculate the income difference between males and females
    1. Create a column `income_diff` to calculate how much more or less each individual's income is compared to the average income of the other gender.
11. Save the final dataset named **Ch2_Practical_Section_GrpNo.csv**

7.5 Data Management (Cross-Sectional) Feedback

This feedback will contain only what the answers should look like and some clues and hints. It does not contain the entire codes.

1. Loading the dataset and loading the needed libraries: `dplyr` and `lubridate`

```
##   Stage.Name.Stage.Name Full.Name.Full.Name Korean.Name.Korean.Name K..Stage.Name.K..S
## 1               Taeyeon          Kim Taeyeon
## 2                 Sunny           Lee Sunkyu
```

```
## 3               Tiffany       Hwang Miyoung
## 4               Hyoyeon         Kim Hyoyeon
## 5                  Yuri           Kwon Yuri
## 6              Sooyoung       Choi Sooyoung
##   Date.of.Birth.Date.of.Birth Group.Group Country.Country Height.Height Weight
## 1                   3/9/1989        SNSD     South Korea           160
## 2                  5/15/1989        SNSD     South Korea           158
## 3                   8/1/1989        SNSD     South Korea           163
## 4                  9/22/1989        SNSD     South Korea           158
## 5                  12/5/1989        SNSD     South Korea           167
## 6                  2/10/1990        SNSD     South Korea           170
##   Gender.Gender Instagram.Instagram
## 1             F          taeyeon_ss
## 2             F          svnnynight
## 3             F        xolovestephi
## 4             F         watasiwahyo
## 5             F             yulyulk
## 6             F          hotsootuff
```

2. Separate the dataset into Males and Females and save as CSVs. You might notice that the column name is `Gender.Gender`. You have to type it out. Later, we will change the names.

```
##   Stage.Name.Stage.Name Full.Name.Full.Name Korean.Name.Korean.Name K..St
## 1                 T.O.P      Choi Seunghyun
## 2               Taeyang       Dong Youngbae
## 3             G-Dragon         Kwon Jiyong
```

```
## 4               Daesung             Daesung
## 5               Seungri       Lee Seunghyun
## 6               Leeteuk        Park Jeongsu
##   Date.of.Birth.Date.of.Birth  Group.Group Country.Country Height.Height Weight.We
## 1                  11/4/1987      BIGBANG     South Korea           180
## 2                  5/18/1988      BIGBANG     South Korea           174
## 3                  8/18/1988      BIGBANG     South Korea           177
## 4                  4/26/1989      BIGBANG     South Korea           178
## 5                 12/12/1990                  South Korea           176
## 6                   7/1/1983 Super Junior     South Korea           179
##   Gender.Gender Instagram.Instagram
## 1             M
## 2             M
## 3             M
## 4             M
## 5             M
## 6             M

##   Stage.Name.Stage.Name Full.Name.Full.Name Korean.Name.Korean.Name K..Stage.Name.
## 1               Taeyeon         Kim Taeyeon
## 2                 Sunny          Lee Sunkyu
## 3               Tiffany       Hwang Miyoung
## 4               Hyoyeon         Kim Hyoyeon
## 5                 Yuri           Kwon Yuri
## 6              Sooyoung       Choi Sooyoung
##   Date.of.Birth.Date.of.Birth Group.Group Country.Country Height.Height Weight.Wei
## 1                   3/9/1989        SNSD     South Korea           160
```

```
## 2                  5/15/1989        SNSD     South Korea           158
## 3                   8/1/1989        SNSD     South Korea           163
## 4                  9/22/1989        SNSD     South Korea           158
## 5                  12/5/1989        SNSD     South Korea           167
## 6                  2/10/1990        SNSD     South Korea           170
##   Gender.Gender Instagram.Instagram
## 1             F          taeyeon_ss
## 2             F          svnnynight
## 3             F        xolovestephi
## 4             F         watasiwahyo
## 5             F             yulyulk
## 6             F          hotsootuff
```

3. Edit Column Names and Remove Unnecessary Columns

We are going to use the pipe operator for this and if you are lazy to type all th

```r
#only those that remain
#col<-c("Full.Name.Full.Name", "Date.of.Birth.Date.of.Birth"...)
```

THEN, after you select the columns to keep, you can rename. Note that you need to

```
```

```
##       Full_Name       DOB  Grp     Country  Ht Wt            BP Gender
## 1   Kim Taeyeon  3/9/1989 SNSD South Korea 160 44        Jeonju      F
## 2    Lee Sunkyu 5/15/1989 SNSD South Korea 158 43    California      F
## 3 Hwang Miyoung  8/1/1989 SNSD South Korea 163 50 San Francisco      F
## 4   Kim Hyoyeon 9/22/1989 SNSD South Korea 158 48       Incheon      F
## 5     Kwon Yuri 12/5/1989 SNSD South Korea 167 45        Goyang      F
## 6 Choi Sooyoung 2/10/1990 SNSD South Korea 170 48       Gwangju      F
```

```
##        Full_Name        DOB           Grp     Country  Ht Wt        BP Gender
## 1 Choi Seunghyun  11/4/1987      BIGBANG South Korea 180 65     Seoul      M
## 2  Dong Youngbae  5/18/1988      BIGBANG South Korea 174 56 Uljeongbu      M
## 3    Kwon Jiyong  8/18/1988      BIGBANG South Korea 177 58     Seoul      M
## 4        Daesung  4/26/1989      BIGBANG South Korea 178 63   Incheon      M
## 5  Lee Seunghyun 12/12/1990              South Korea 176 60   Gwangju      M
## 6   Park Jeongsu   7/1/1983 Super Junior South Korea 179 59     Seoul      M
```

4. Count the Number of Males and Females

   As mentioned, use `nrow`. Now, I will introduce you to the `cat` function, The `cat` function will print in the result some text and what will be seen when you include the object you created that reveals the number of males (or females). We should all have the same number. If not, something is wrong.

```
#cat("Number of males:", nummale,"\n")
```

```
## Number of males: 843
```

```
## Number of females: 823
```

5. Create a binary variable `not_seoul`

   Here, create a new column for not_seoul in both males and females. Use `ifelse` and the statement should include `!=`

6. Count Individuals from and not from Seoul

   I introduce the `count` function. Simply add `count` after choosing the dataset. So, the code is choose males THEN count those that are not_seoul. We should have the same number.

```
## Males from Seoul and not Seoul:
```

```
## From Seoul: 98
```

```
## Not from Seoul: 745
```

```
## Females from Seoul and not Seoul:
```

```
## From Seoul: 108
```

```
## Not from Seoul: 715
```

7. Filter Males Eligible for Military Service

   Use reference date and when you filter, you can actually combine filtering the Age to be: Age`>= 18, Age<=28, Country == "South Korea"`. We should have the same number.

```
## Number of males eligible for military service: 496
```

8. Assign Generations Based on Age

   We should have the same numbers. If you have created an `Age` column for males before, you cannot use that for this number since the reference dates are different. You need to create a new column for `Age` for both males and females.

```
## Generation distribution for males:

## 1st Gen: 8
## 2nd Gen: 137
## 3rd Gen: 358
## 4th Gen: 340

## Generation distribution for females:

## 1st Gen: 4
## 2nd Gen: 131
## 3rd Gen: 302
## 4th Gen: 386
```

9. Create Income Column Based on Hypothesis

   My hypothesis is that older idols earn more since they have been in the industry much longer. I also hypothesize that females earn less than males.

   I introduce a new function here, it is the `runif(n(), value, value)`. This function is to just create income values which will be in-line with

the number of rows in the dataset. However, I am amenable in any strategy you employ to generate income here. I just need to know your hypothesis and how you plan to do the income column. In fact, if you want, you can even create the income column in Excel already. Just let me know.

Another strategy is to still do `case_when` and have smaller increments in `Age` then have income values. Another, have the same income for each generation, that is also fine. Again, this is for practice purposes.

```
# Generate income values based on age group directly
set.seed(123)  # For reproducibility

males <- males %>%
  mutate(Income = case_when(
    Age2 >= 40 ~ runif(n(), 50000, 70000),  # 1st Gen
    Age2 >= 31 & Age2 <= 39 ~ runif(n(), 40000, 60000),  # 2nd Gen
    Age2 >= 25 & Age2 <= 30 ~ runif(n(), 30000, 50000),  # 3rd Gen
    Age2 <= 24 ~ runif(n(), 20000, 40000)  # 4th Gen
  ))

females <- females %>%
  mutate(Income = case_when(
    Age2 >= 40 ~ runif(n(), 50000, 70000),  # 1st Gen
    Age2 >= 31 & Age2 <= 39 ~ runif(n(), 40000, 60000),  # 2nd Gen
    Age2 >= 25 & Age2 <= 30 ~ runif(n(), 30000, 50000),  # 3rd Gen
    Age2 <= 24 ~ runif(n(), 20000, 40000)  # 4th Gen
  ))
```

10. Combine datasets

    So, I taught you how to merge the datasets. You can do that as well, however, it is important to determine which came from the males, which came from the females so, you need to create a new column in the males and the females that would include the Gender.

    I also show a different way of merging datasets. I use `bind_rows` since males and females have the same columns. However, if you view the `combined` dataset, you will notice that the females will appear after the males. This is fine.

```
# Ensure column names match and add a Gender column
males <- males %>% mutate(Gender = "Male")
females <- females %>% mutate(Gender = "Female")

# Combine the datasets
combined <- bind_rows(males, females)

# View the combined dataset
View(combined)
```

Calculate the Income Difference

I use the `group_by` function and the `summarise` function. I got the Mean and the Median. Later on, I will discuss the difference between getting the mean and the median or when best to use the mean or the median.

```
## # A tibble: 2 x 3
##   Gender Mean_Income Median_Income
```

```
##   <chr>        <dbl>         <dbl>
## 1 Female      36858.        36414.
## 2 Male        37907.        37246.

## The gender with the higher mean income is: Male

## The gender with the higher median income is: Male
```

11. Now that we have the combined dataset, simply save it as a CSV file and you are done with the practical.

Chapter 8

Data Management - Time Series and Panel Data

For this portion of the discussion, we will use one dataset then generate randomly here in R some practice data frames.

8.1 Topic Guide:

1. Changing from daily to monthly to quarterly to yearly
2. Change from long to wide and vice-versa
3. Missing values

8.2 Time Series Data

For this, we will use Chapter3 Practice found in the Modules.

8.2.1 Preliminaries

Always remember the first steps: Set Working Directory and Clean the Global Environment.

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

```
##            used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells  5655371 302.1    9801260 523.5  9801260 523.5
## Vcells 11563812  88.3   36965975 282.1 94237569 719.0
```

To make sure that you are using the correct directory and that you have all the files you need, use `list.files()` function.

```
list.files()
```

Now, we load the following packages: `dplyr`, `lubridate`, and `zoo`. Make sure you have all 3 installed; if not, install them.

```
library(dplyr)
library(lubridate)
library(zoo)
```

Now we load the csv file:

```
ch3_p1<-read.csv("Ch3Practice.csv")
head(ch3_p1)
```

```
##           ds       y
## 1 2015-06-13 232.402
## 2 2015-06-14 233.543
## 3 2015-06-15 236.823
## 4 2015-06-16 250.895
## 5 2015-06-17 249.284
## 6 2015-06-18 249.007
```

We need to understand our data;

```
str(ch3_p1)
```

```
## 'data.frame':    1825 obs. of  2 variables:
##  $ ds: chr  "2015-06-13" "2015-06-14" "2015-06-15" "2015-06-16" ...
##  $ y : num  232 234 237 251 249 ...
```

8.2.2 Convert to Date Format

As you can see, our ds is what our date column is, however, it is in the `character` format. We need to convert it to the `Date` class. We also need to do this to a copy of the raw data for further modifications.

```
ch3_p1.1<-ch3_p1
head(ch3_p1.1$ds)
```

```
## [1] "2015-06-13" "2015-06-14" "2015-06-15" "2015-06-16" "2015-06-17" "2015-06-18"
```

```
class(ch3_p1.1$ds)
```

```
## [1] "character"
```

```
ch3_p1.1$ds <- as.Date(ch3_p1.1$ds, format = "%Y-%m-%d")

class(ch3_p1.1$ds)
```

```
## [1] "Date"
```

8.2.3 Aggregate Data

We now have `daily` data. Say we want to create weekly data, we use the `cut` function to group dates by week, month, quarter and year.

Since we know for sure that the `date` column is in `date` format, no need to check, however, it is always useful to check the class of `date`.

8.2.3.1 Aggregate by Week

Add new columns for each aggregation.

```
ch3_p1.1$week<-cut(ch3_p1.1$ds, breaks = "week")
```

The `cut` is used to divide the date into intervals while the `breaks` specifies that the `date` be divided into weekly intervals.

Check creation of the week column

```
head(ch3_p1.1)
```

```
##           ds       y       week
## 1 2015-06-13 232.402 2015-06-08
## 2 2015-06-14 233.543 2015-06-08
## 3 2015-06-15 236.823 2015-06-15
## 4 2015-06-16 250.895 2015-06-15
## 5 2015-06-17 249.284 2015-06-15
## 6 2015-06-18 249.007 2015-06-15
```

Since we have the `y` column which is actually Bitcoin Price, we need to aggregate that weekly using the `aggregate` function

```
week_y<-aggregate(. ~ week, #refers to all other columns in the dataset and groups them b
                  data=ch3_p1.1,
                  FUN = mean #specifies that the mean function should be applied to the n
                  )
```

You will notice that this creates a separate data frame. We will merge `week_y` with `ch3_p1.1`

```
ch3_p1.1<-merge(ch3_p1.1, week_y, by = "week", #ensures the merge aligns based on the wee
                suffixes = c("","_weekly")) #adds _weekly to the column name to distingui
```

We will slightly do the same thing when aggregating by month, quarter and year. I will do the initial steps, but please do the succeeding steps on your own.

8.2.3.2 Aggregate by Month

```
# Add a month column
ch3_p1.1$month <- format(ch3_p1.1$ds, "%Y-%m")
```

```
# Calculate monthly means
month_y <- aggregate(. ~ month, data = ch3_p1.1, FUN = mean)
```

8.2.3.3 Aggregate by Quarter

This is different since we will use the `paste0` and the `format` functions. The format function extracts the year from the date and extracts the quarter from the date. The paste0 combines the year and quarter without a space between them so that it results in which quarter of which year.

```
ch3_p1.1$quarter <- paste0(format(ch3_p1.1$ds, "%Y"), " ", quarters(ch3_p1.1$ds))
```

8.2.3.4 Aggregate by Year

```
ch3_p1.1$year<-format(ch3_p1.1$ds, "%Y")
```

Can you aggregate the Bitcoin values on your own?

8.3 Modifying Long and Wide Datasets

8.3.1 How to Determine

Aspect	Long Format	Wide Format
Rows	Each row represents a single observation (or measurement)	Each row represents an entity (like a group)
Columns	Variables are split into multiple rows	Variables are spread across multiple columns
Compactness	More rows, fewer columns	Fewer rows, more columns

8.3.1.1 Examples of Long and Wide Formats

8.3.1.1.1 1. Student Scores

8.3.1.1.1.1 Long Format

Student	Subject	Score
Alice	Math	90
Alice	Science	85
Bob	Math	88
Bob	Science	85

8.3.1.1.1.2 Wide Format

Student	Math	Science
Alice	90	85
Bob	88	85

8.3.1.1.2 2. Temperature Data

8.3.1.1.2.1 Long Format

Date	Location	Temperature
2023-01-01	City A	15
2023-01-01	City B	20
2023-01-02	City A	16
2023-01-02	City B	21

8.3.1.1.2.2 Wide Format

Date	City A	City B
2023-01-01	15	20
2023-01-02	16	21

8.3.2 Setting up the Dataset

8.3.2.1 Generate a Long Dataset

For practice, we will generate a long dataset. We will need `set.seed` for reproducibility when you want to run the entire code again and to generate

random values, we will use the `rnorm` function.

```
set.seed(123)


#Generate a long dataset
long<-data.frame(
  id = rep(1:5, each = 3), #creates 5 groups and repeated 3 times each
  variable = rep(c("A", "B", "C"), times=5), #Variables A,B,C are created and repeated 5
  value = rnorm(15, mean = 50, sd = 10) #creates random values for 15 observations with m
)
print(long)
```

```
##    id variable    value
## 1   1        A 44.39524
## 2   1        B 47.69823
## 3   1        C 65.58708
## 4   2        A 50.70508
## 5   2        B 51.29288
## 6   2        C 67.15065
## 7   3        A 54.60916
## 8   3        B 37.34939
## 9   3        C 43.13147
## 10  4        A 45.54338
## 11  4        B 62.24082
## 12  4        C 53.59814
## 13  5        A 54.00771
## 14  5        B 51.10683
```

```
## 15  5        C 44.44159
```

8.3.2.2 Converting Long to Wide Format

We will use the `pivot_wider()` function from the `tidyr` package.

```
library(tidyr)

wide<-pivot_wider(
  data = long,
  names_from = variable, #Columns to become new column names
  values_from = value #Values to put under new columns
)

print(wide)
```

```
## # A tibble: 5 x 4
##      id     A     B     C
##   <int> <dbl> <dbl> <dbl>
## 1     1  44.4  47.7  65.6
## 2     2  50.7  51.3  67.2
## 3     3  54.6  37.3  43.1
## 4     4  45.5  62.2  53.6
## 5     5  54.0  51.1  44.4
```

8.3.2.3 Convert Wide to Long

We will use the converted wide dataset for this. We will use the `pivot_longer` function.

```
long2<-pivot_longer(
  data = wide,
  cols = A:C, #Which columns to collapse
  names_to = "variable", #New column for variable names
  values_to = "value" #new column for values
)


print(long2)
```

```
## # A tibble: 15 x 3
##       id variable value
##    <int> <chr>    <dbl>
##  1     1 A         44.4
##  2     1 B         47.7
##  3     1 C         65.6
##  4     2 A         50.7
##  5     2 B         51.3
##  6     2 C         67.2
##  7     3 A         54.6
##  8     3 B         37.3
##  9     3 C         43.1
## 10     4 A         45.5
## 11     4 B         62.2
```

```
## 12     4 C         53.6
## 13     5 A         54.0
## 14     5 B         51.1
## 15     5 C         44.4
```

8.4 Missing Values

In handling missing values in R, we focus on 3 common methods:

1. Replacing missing values with `0`
2. Removing rows or columns with missing values
3. Replacing missing values with the mean

8.4.1 Setting up a Sample Dataset

```r
set.seed(123)
ch3_p2<-data.frame(
  id = 1:5,
  var1 = c(10, NA, 30, 40, NA),
  var2 = c(NA, 15, 25, 35, 45),
  var3 = rnorm(5, mean = 50, sd=10)
)

print(ch3_p2)
```

```
##   id var1 var2     var3
```

```
## 1  1   10   NA 44.39524
## 2  2   NA   15 47.69823
## 3  3   30   25 65.58708
## 4  4   40   35 50.70508
## 5  5   NA   45 51.29288
```

8.4.2 Replace with 0

8.4.2.1 For a Specific Column:

```
ch3_p2.1<-ch3_p2
ch3_p2.1$var1[is.na(ch3_p2.1$var1)]<-0
print(ch3_p2.1)
```

```
##   id var1 var2     var3
## 1  1   10   NA 44.39524
## 2  2    0   15 47.69823
## 3  3   30   25 65.58708
## 4  4   40   35 50.70508
## 5  5    0   45 51.29288
```

8.4.2.2 For the Entire Dataset

```
ch3_p2.2<-ch3_p2
ch3_p2.2[is.na(ch3_p2.2)]<-0


print(ch3_p2.2)
```

```
##   id var1 var2     var3
## 1  1   10    0 44.39524
## 2  2    0   15 47.69823
## 3  3   30   25 65.58708
## 4  4   40   35 50.70508
## 5  5    0   45 51.29288
```

8.4.3 Remove Rows or Columns with Missing Values

8.4.3.1 Remove Rows with Missing Values

```
ch3_p2.3<-ch3_p2
ch3_p2.3<-na.omit(ch3_p2.3)
print(ch3_p2.3)
```

```
##   id var1 var2     var3
## 3  3   30   25 65.58708
## 4  4   40   35 50.70508
```

8.4.3.2 Remove Columns with Missing Data

```
ch3_p2.4<-ch3_p2
ch3_p2.4<-ch3_p2.4[, colSums(is.na(ch3_p2.4)) ==0]
print(ch3_p2.4)
```

```
##   id     var3
```

```
## 1  1 44.39524
## 2  2 47.69823
## 3  3 65.58708
## 4  4 50.70508
## 5  5 51.29288
```

8.4.4 Imputing Missing Values with the Mean

Here, we will impute missing values with the mean and unlike previous methods, this method uses a `for` function to loop through columns and check for columns with missing values. However, this method needs to have numeric format.

```
ch3_p2.5<-ch3_p2
for(col in names(ch3_p2.5)){
  if(is.numeric(ch3_p2.5[[col]])) {#Check if column is numeric
    ch3_p2.5[[col]][is.na(ch3_p2.5[[col]])]<-mean(ch3_p2.5[[col]], na.rm=TRUE)
    }
}
print(ch3_p2.5)
```

```
##   id     var1 var2     var3
## 1  1 10.00000   30 44.39524
## 2  2 26.66667   15 47.69823
## 3  3 30.00000   25 65.58708
## 4  4 40.00000   35 50.70508
## 5  5 26.66667   45 51.29288
```

8.4.5 Note:

When doing the three methods, there are pros and cons.

1. Replacing with Zero can introduce bias if `0` is not realistic in the context of the data
2. Remove Missing Values can result in significant data loss
3. Imputing the Mean assumes the data is evenly distributed and can distort the variability in the data

Always make sure you should understand your data.

8.4.6 Best Practice:

1. Always **explore and visualize** patterns first.

Leading to…

8.4.7 Next Meeting:

1. Visualizing the data with base R package

8.4.8 Final Note:

No Practical for this session; please work on the previous practical and the research problem to be submitted on January 24.

Next week, we will have a computer practical.

Chapter 9

Visualizations using ggplot2

For this lecture, we will be using `ggplot2` package. Please make sure that you have it installed. The package works best with data in the 'long' format so it helps to modify the dataset to this format rather than a wide format.

9.1 Preliminaries

9.1.1 Load the dataset

The dataset can be downloaded from the Modules. We will use `Ch4PracticeA` for this portion of the discussion.

Unlike previous datasets that we loaded, we are now loading an Excel file. We will need the `readxl` package for this. Also, it is important to check if there are additional sheets and which sheet you will need. There are actually two sheets in the Excel file, but we will only use the first sheet named `base`.

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

```
##            used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells  5655268 302.1    9801260 523.5  9801260 523.5
## Vcells 11544345  88.1   36965975 282.1 94237569 719.0
```

```
library(readxl)
ch4_p1 <- read_excel("Ch4PracticeA.xlsx", sheet = "base")
```

I will not delve deeply in the description. Please read up on it. The description can be found in the last sheet of the Excel file. Next, we load the following package: `tidyverse`

```
library(tidyverse)
```

9.1.2 Template

There is a basic template that can be used for different types of plots:

```
<DATA> %>%
  ggplot(aes(<MAPPINGS>))+
  <GEOM_FUNCTION>()
```

`ggplot` is a function that expects a data frame to be the first argument. This allows for us to change from specifying the `data =` argument within the ggplot function and instead pipe the data into the function.

Use the `ggplot()` function...

```
ch4_p1 %>%
  ggplot()
```

Now, we define the mapping (using the aesthetic (aes) function), by selecting the variables to be plotted and specifying how to present them in the graph, e.g. as x/y positions or characteristics such as size, shape, color, etc.

```
ch4_p1 %>%
  ggplot(aes(x=poverty_index,
             y=health_expenditures))
```

The next step is to add geom which will make the graphical representations of the data. These include:

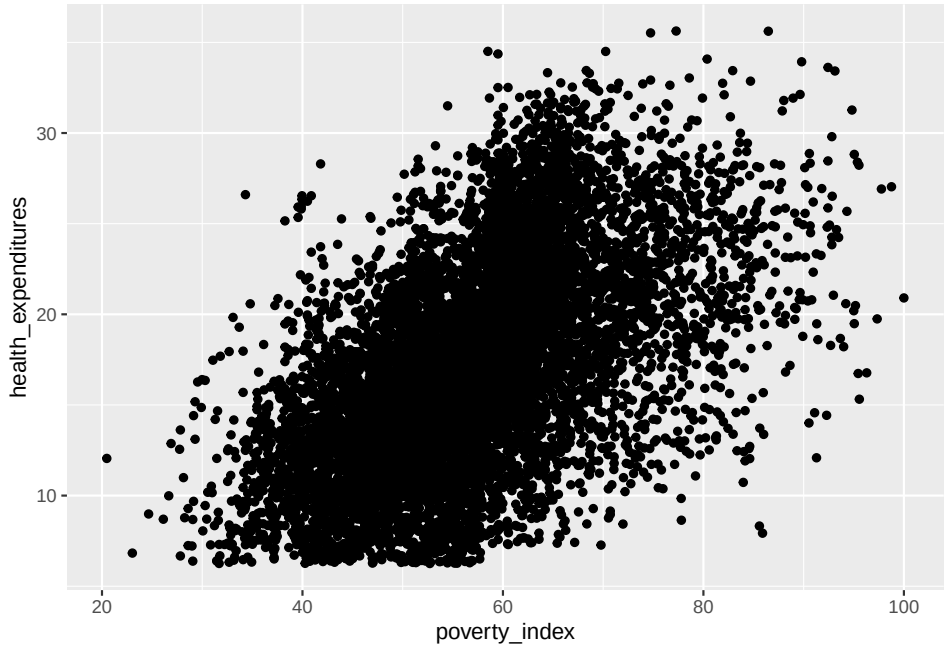
- geom_point() for scatter plots, dot plots, etc.
- geom_boxplot() for boxplots
- geom_line() for trend lines, time series, etc.
- geom_bar() for bar plots and pie charts

9.2 Visualizing Data using geom

9.2.1 Scatterplots

Let us use the geom_point() first then we will do the others after. Also, scatterplots are useful when you want to display the relationship between two continuous variables. Can you give me an example of when to use scatterplots?

```
ch4_p1 %>%
  ggplot(aes(x=poverty_index,
             y=health_expenditures)) +
  geom_point()
```

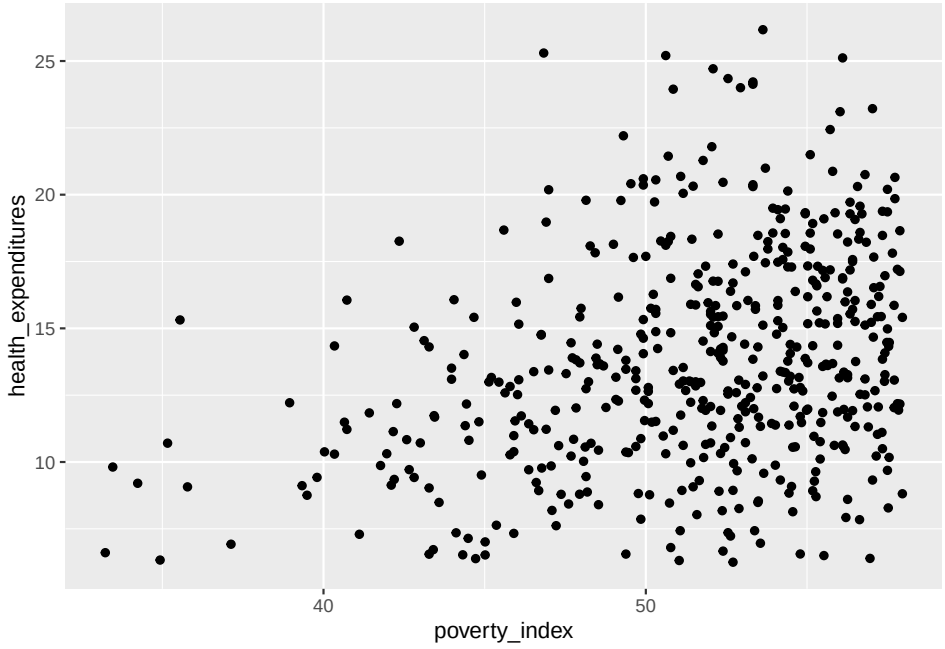


This visualization is so unclear; this is due to the number of observations being more than 9,000. Let's just use the first 500 rows as this is for our practice and for visualization purposes.

```
fch4_p1 <- head(ch4_p1, 500)
View(fch4_p1)
```

This is more manageable; let's try the scatterplot again, this time using the `fch4_p1` dataset.

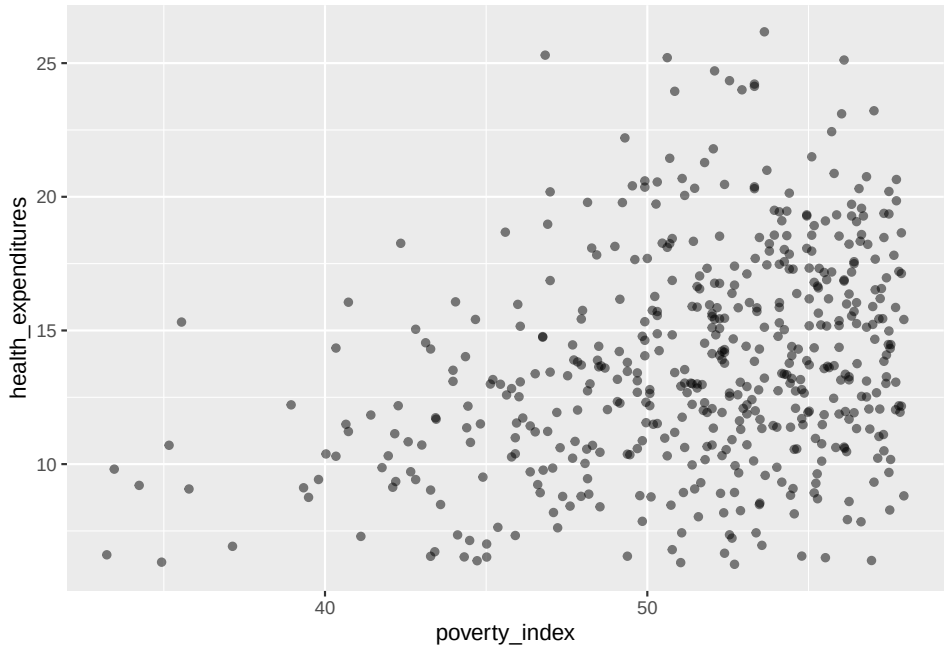
```
fch4_p1%>%
  ggplot(aes(x=poverty_index,
             y=health_expenditures)) +
  geom_point()
```



This is more visible; There are some points that overlap with each other. Let us incorporate some strategies to try and ensure that there will be no overplotting issues.

The first strategy is changing the transparency of the points. To control the transparency of points, we add the `alpha` argument. The range of transparency is from 0 to 1, with lower values corresponding to more transparency. The default value is 1. Let's try to change the `alpha` to 0.5.

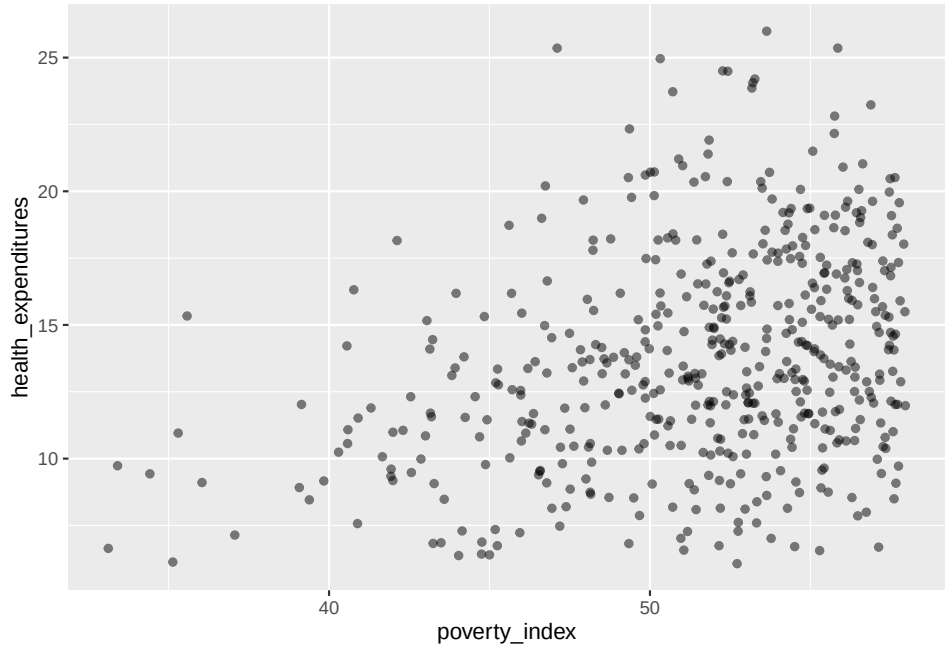
```
fch4_p1 %>%
  ggplot(aes(x=poverty_index,
             y=health_expenditures)) +
  geom_point(alpha=0.5)
```



Some of the points are gray while the others are much darker, then we can see (*slightly*) the difference.

Another method that we can do is jittering the points on the plot to see the locations where there are overplotting points. Jittering adds randomness into the position of the points. To do this, we add `geom_jitter()` rather than `geom_point()`. Also, we need to edit the `width` and `height`. You can experiment but if you want less spread, pick values between 0.1 and 0.4.

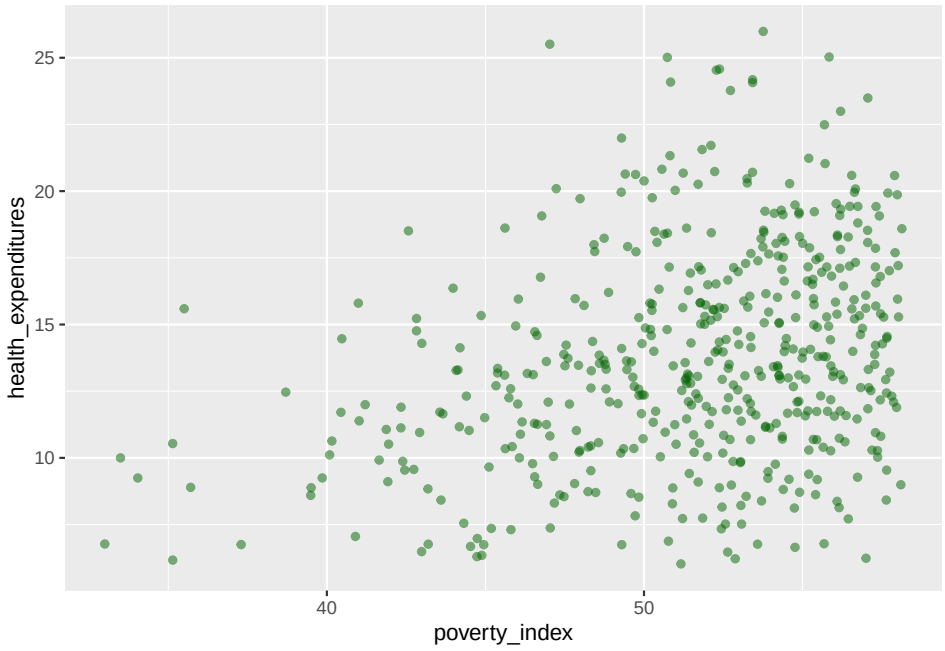
```
fch4_p1 %>%
  ggplot(aes(x=poverty_index,
             y=health_expenditures)) +
  geom_jitter(alpha = 0.5,
              width= 0.3,
              height= 0.3)
```



Let us add color to `geom_jitter()`

```
fch4_p1 %>%
  ggplot(aes(x=poverty_index,
             y=health_expenditures)) +
  geom_jitter(alpha = 0.5,
              color = "darkgreen",
```

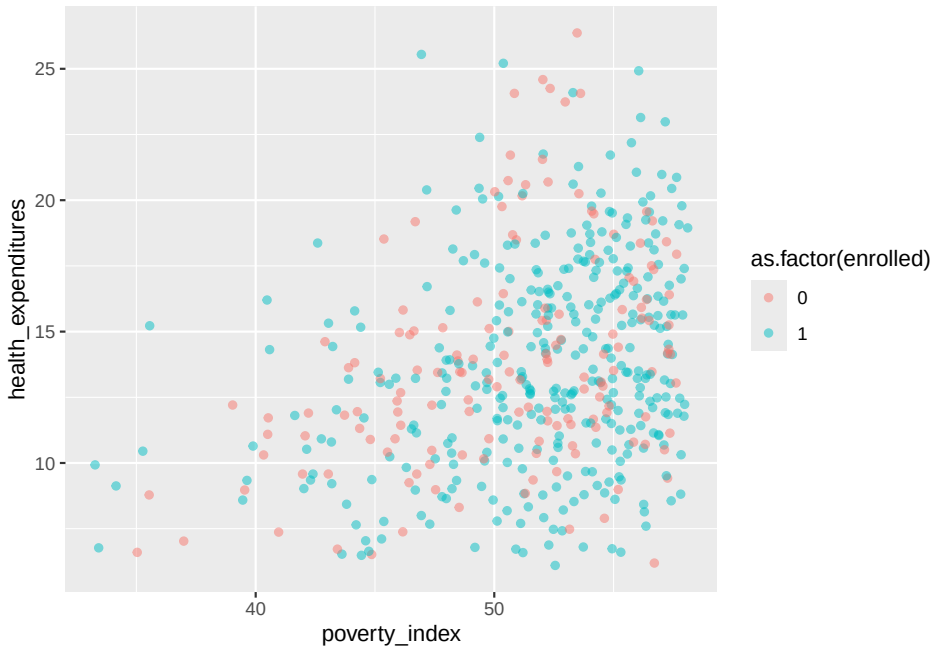
```
                width= 0.3,
                height= 0.3)
```



If you want to have different colors depending on a certain variable, we need to use a vector as an input in the argument `color`. Here though, we map features of the data to a certain color. When we map a variable in our data to the color of the points, ggplot2 will provide a different color corresponding to the different values of the variable. We will continue to specify the value of alpha, width, and height outside of the aes function because we are using the same value for every point.

```
fch4_p1 %>%
  ggplot(aes(x=poverty_index,
             y=health_expenditures))+
```

```
  geom_jitter(aes(color=as.factor(enrolled)), #this is because the enrolled is of num typ
              alpha=0.5,
              width=0.3,
              height=0.3)
```



9.2.2 Boxplots

Here is how to make a box plot that is useful when summarizing the distribution of a continuous variable, highlighting the median, quartiles, and potential outliers.

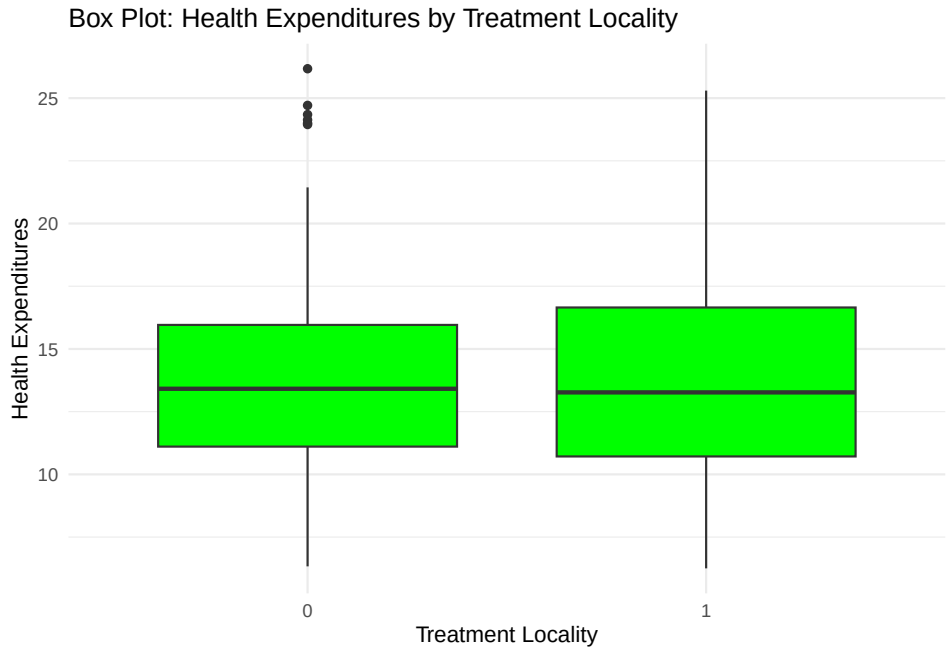
To interpret a boxplot, take note of the following things:

1. Horizontal Line in the Box: this is the median wherein it represents the 50% of the data.

2. The Box Itself: The box represents the middle 50% of the data, the bottom of the box represent first quartile so 25% of the data falls below this value, and the top of the box represent the third quartile so 75% of the data falls below this value.
3. Whiskers extend from the box to represent the range of the data excluding outliers so it usually end at the smallest and largest data points.
4. Outliers are points outside the whiskers.

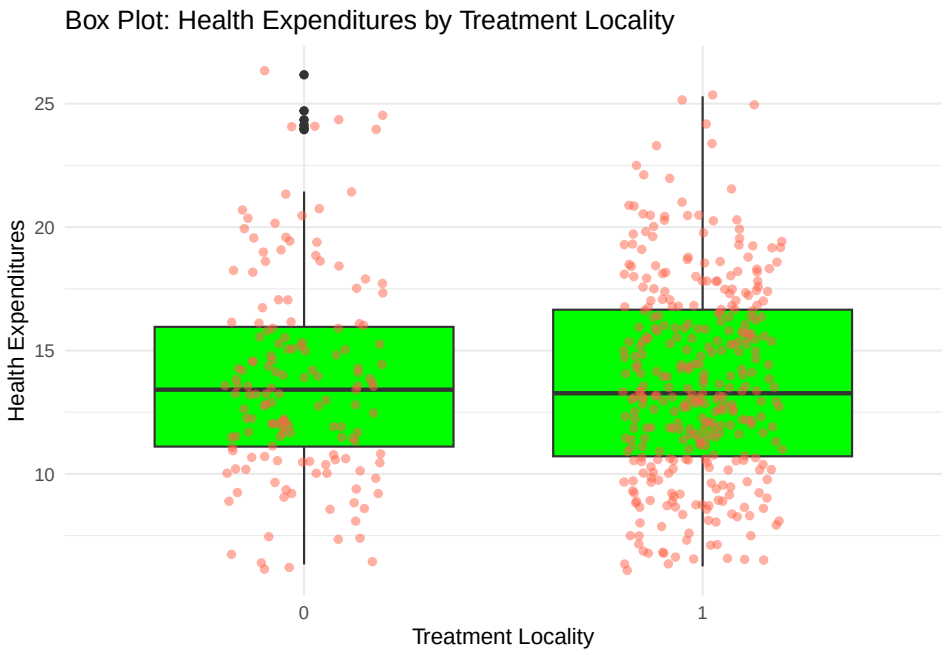
*The `labs` argument is to add labels in the plot.

```
fch4_p1 %>%
ggplot(aes(x = as.factor(treatment_locality), y = health_expenditures)) +
  geom_boxplot(fill = "green") +
  labs(title = "Box Plot: Health Expenditures by Treatment Locality",
       x = "Treatment Locality",
       y = "Health Expenditures") +
  theme_minimal()
```



You can also add some points in the box plot by adding `geom_jitter`

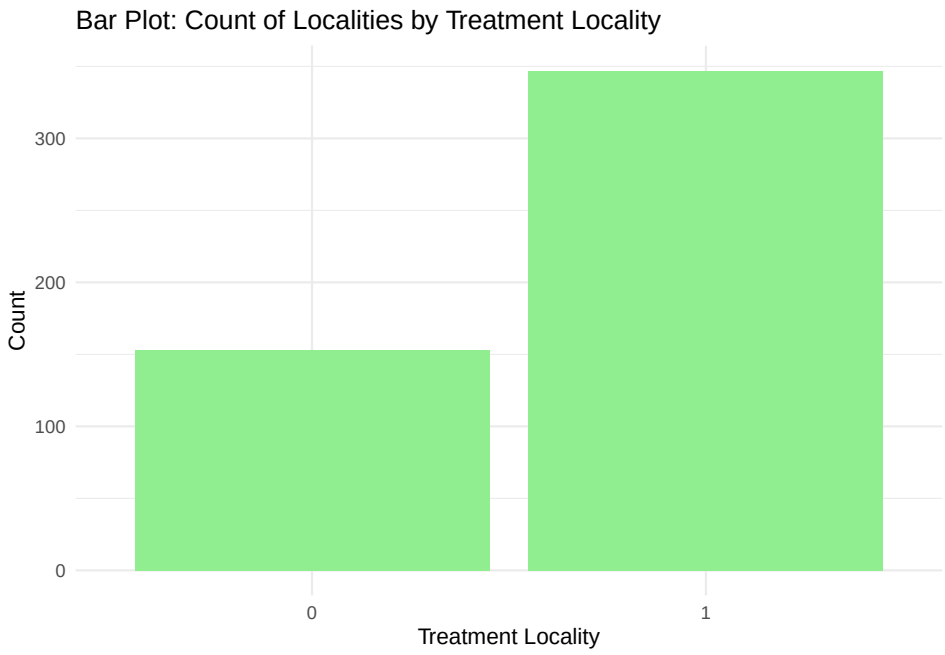
```
fch4_p1 %>%
ggplot(aes(x = as.factor(treatment_locality), y = health_expenditures)) +
  geom_boxplot(fill="green") +
geom_jitter(alpha = 0.5,
color = "tomato",
width = 0.2,
height = 0.2) +
  labs(title = "Box Plot: Health Expenditures by Treatment Locality",
       x = "Treatment Locality",
       y = "Health Expenditures") +
  theme_minimal()
```



9.2.3 Bar plots

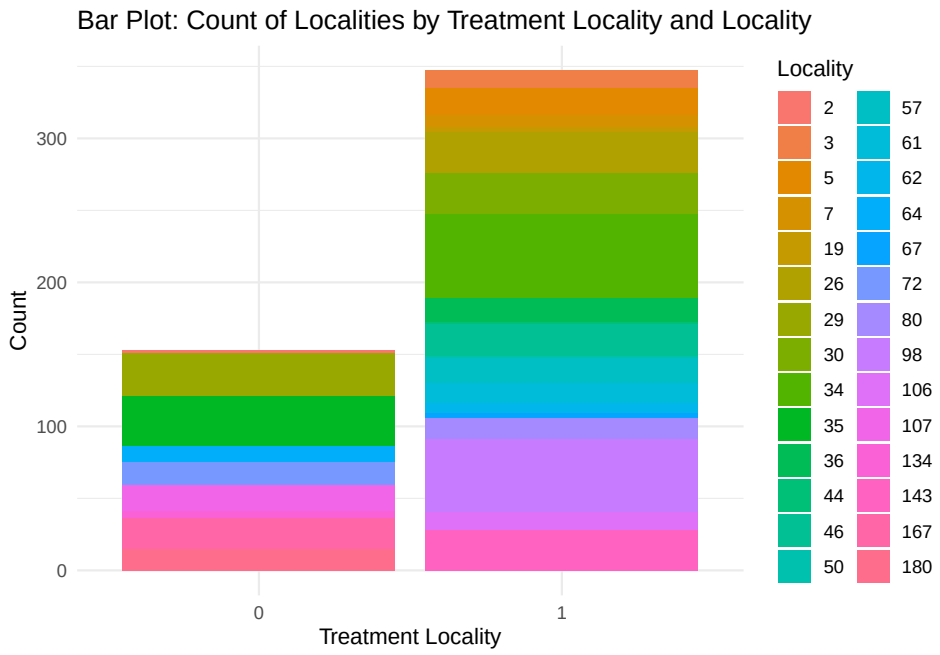
Barplots are also useful for visualizing categorical data. By default, `geom_bar` accepts a variable for x, and plots the number of instances each value of x (in this case, treatment_locality) appears in the dataset.

```
fch4_p1 %>%
ggplot(aes(x = as.factor(treatment_locality))) +
  geom_bar(fill = "lightgreen") +
  labs(title = "Bar Plot: Count of Localities by Treatment Locality",
       x = "Treatment Locality",
       y = "Count") +
  theme_minimal()
```



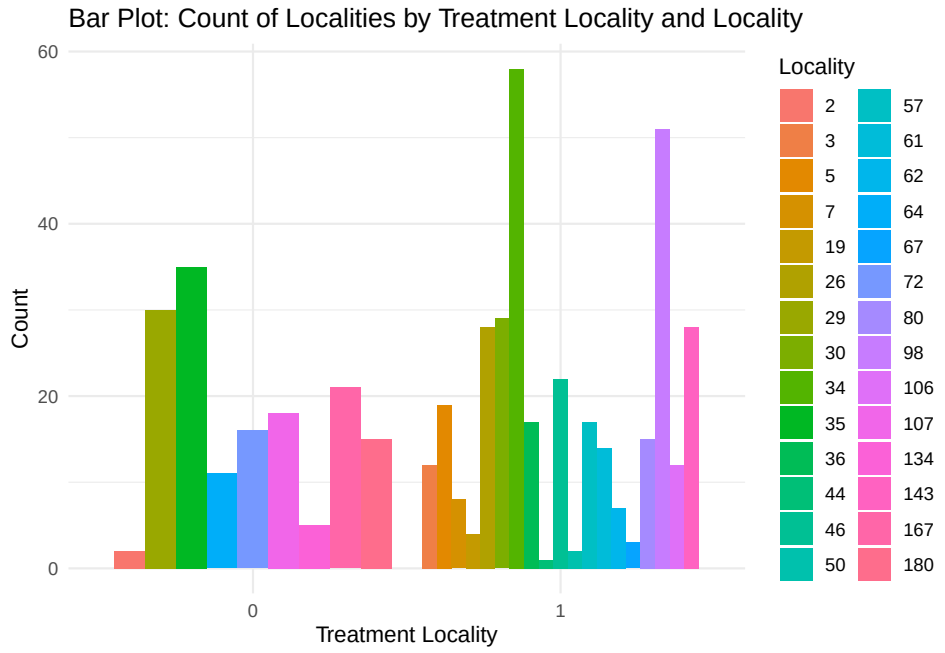
Let us change the fill to be `locality_identifier`. Note, we will have a lot of colors here but this is done for visualization purposes and practice.

```
fch4_p1 %>%
  ggplot(aes(x = as.factor(treatment_locality), fill = as.factor(locality_identifier))) +
  geom_bar() +
  labs(title = "Bar Plot: Count of Localities by Treatment Locality and Locality",
       x = "Treatment Locality",
       y = "Count",
       fill = "Locality") +
  theme_minimal()
```



This creates a stacked bar chart. These are generally more difficult to read than side-by-side bars. We can separate the portions of the stacked bar that correspond to each village and put them side-by-side by using the position argument for `geom_bar()` and setting it to "dodge".

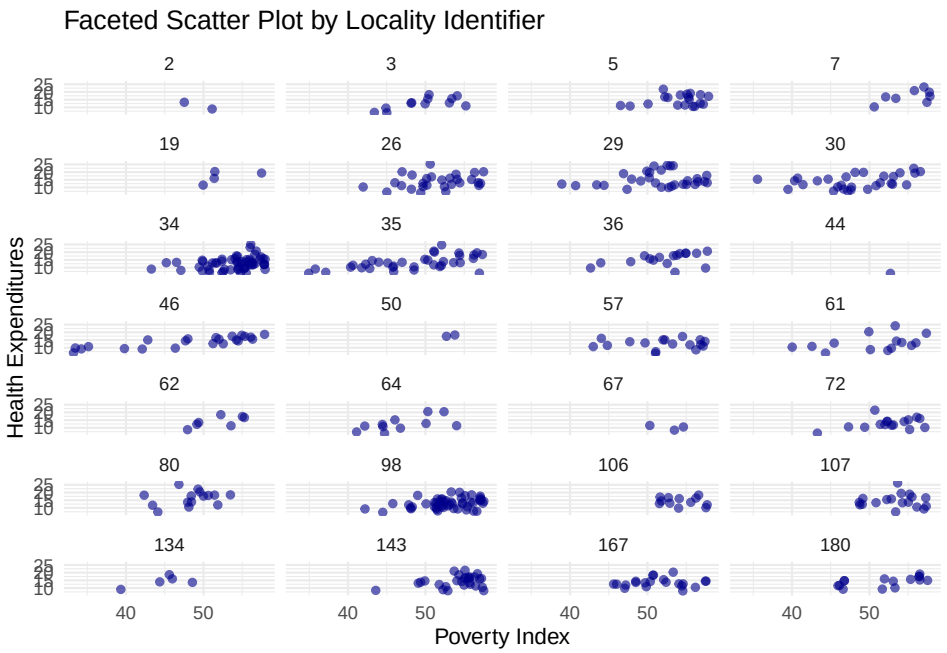
```
fch4_p1 %>%
  ggplot(aes(x = as.factor(treatment_locality), fill = as.factor(locality_identi
  geom_bar(position = "dodge") +
  labs(title = "Bar Plot: Count of Localities by Treatment Locality and Locality"
       x = "Treatment Locality",
       y = "Count",
       fill = "Locality") +
  theme_minimal()
```



9.2.4 Faceting

ggplot2 has a special technique called faceting that allows the user to split one plot into multiple plots based on a factor included in the dataset.

```
fch4_p1 %>%
ggplot(aes(x = poverty_index, y = health_expenditures)) +
  geom_point(alpha = 0.6, color = "darkblue") +
  facet_wrap(~ locality_identifier, ncol = 4) +
  labs(title = "Faceted Scatter Plot by Locality Identifier",
       x = "Poverty Index",
       y = "Health Expenditures") +
  theme_minimal()
```

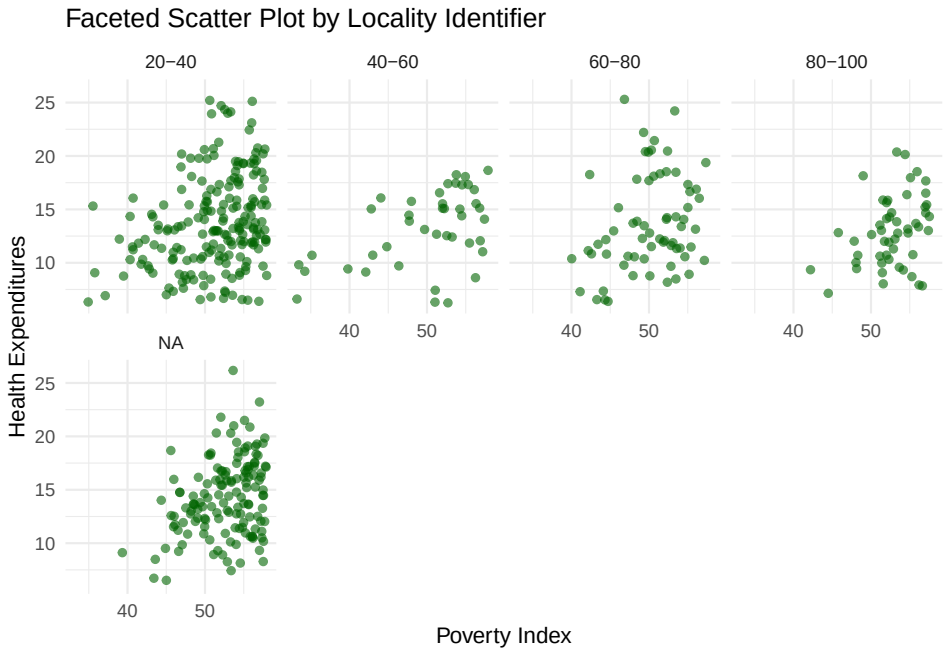


It doesn't look that nice; let's group the locality_identifier to make it more visually appealing:

```
fch4_p1 <- fch4_p1 %>%
  mutate(locality_group = cut(locality_identifier,
                              breaks = c(20, 40, 60, 80, 100),
                              labels = c("20-40", "40-60", "60-80", "80-100"),
                              include.lowest = TRUE))

table(fch4_p1$locality_group)
```

```
##
##  20-40  40-60  60-80 80-100
##    197     42     66     51
```

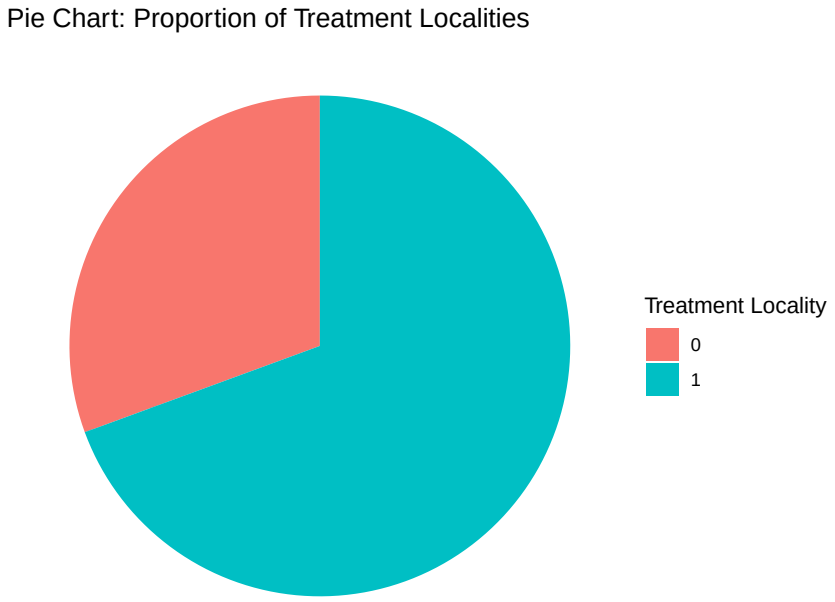
```
fch4_p1 %>%
ggplot(aes(x = poverty_index, y = health_expenditures)) +
  geom_point(alpha = 0.6, color = "darkgreen") +
  facet_wrap(~ locality_group, ncol = 4) +
  labs(title = "Faceted Scatter Plot by Locality Identifier",
       x = "Poverty Index",
       y = "Health Expenditures") +
  theme_minimal()
```



9.2.5 Pie Chart

Pie charts are used to illustrate proportions or parts of a whole (limited to a small number of categories). Before we do the pie chart, we need to count how many observations there are in the variable we want to analyze.

```
treatment_counts <- fch4_p1 %>%
  count(treatment_locality)


ggplot(treatment_counts, aes(x = "", y = n, fill = as.factor(treatment_locality))
  geom_bar(stat = "identity", width = 1) +
  coord_polar(theta = "y") + #this makes it a pie chart
  labs(title = "Pie Chart: Proportion of Treatment Localities",
       fill = "Treatment Locality") +
  theme_void() #another way to have a nice background
```

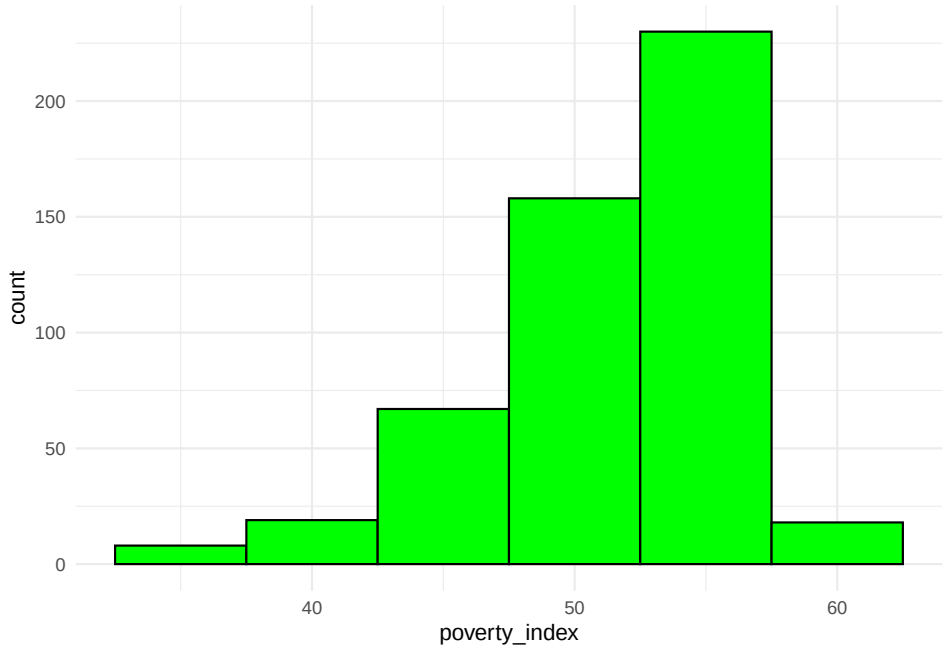


9.2.6 Histogram

Though it looks similar to a bar plot, a histogram is different since it displays the distribution of a continuous variable. It groups the data into intervals

called bins and shows the frequency of data points within each bin.

```
fch4_p1 %>%
  ggplot(aes(x=poverty_index))+
  geom_histogram(binwidth = 5, fill="green", color="black")+
  theme_minimal()
```



Here is a cheat sheet for ggplot2 from the ones who developed the package: ggplot2 Cheat Sheet

9.3 Visualizing Time Series Data

When visualizing time series data, it is important to ensure that the time variable is formatted as Date. For this portion of the lecture, we use

`Ch4PracticeB.xlsx` which is found in the Modules. Make sure to clean the environment, load the file and rename the columns since they are quite long. I will not show the codes for this portion anymore as I am sure you already know how.

```
##            used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells  5655230 302.1    9801260 523.5  9801260 523.5
## Vcells 11545080  88.1   36965975 282.1 94237569 719.0
```

```
## # A tibble: 6 x 12
##   year  nominal_gdp_current nominal_gdp_constant gdp_growth_current gdp_growth
##   <chr>               <dbl>                <dbl>              <dbl>
## 1 1986               692852              4298952              0.065
## 2 1987               777283              4486464              0.122
## 3 1988               910280              4786920              0.171
## 4 1989              1054529              5082939              0.158
## 5 1990              1227882              5239629              0.164
## 6 1991              1422958              5216764              0.159
## # i 5 more variables: interest_payments <dbl>, amortization_payments <dbl>, ph
## #   inflation <dbl>, fdi_net <dbl>
```

1. Ensure the time variable is formatted as `Date`

   I will not show the code for this since this has been done in previous lectures

9.3.1 Template

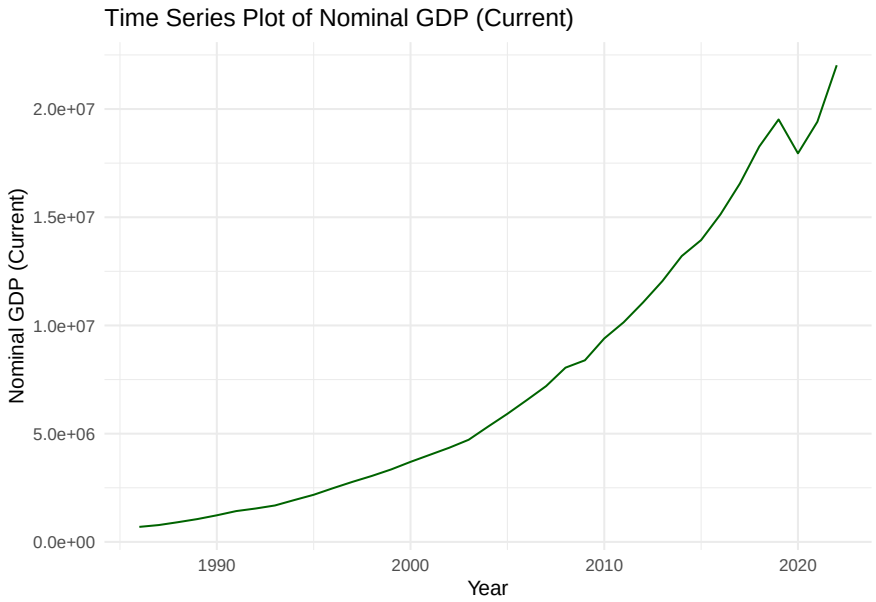
To make a time series visualization, this is the template:

```
ggplot(data, aes(x = time_variable, y = value_variable)) +
  geom_line(color = "blue") +   #adds a trend line
  labs(title = "Time Series Plot", x = "Time", y = "Value") +
  theme_minimal()
```

9.3.2 Time Series Plot

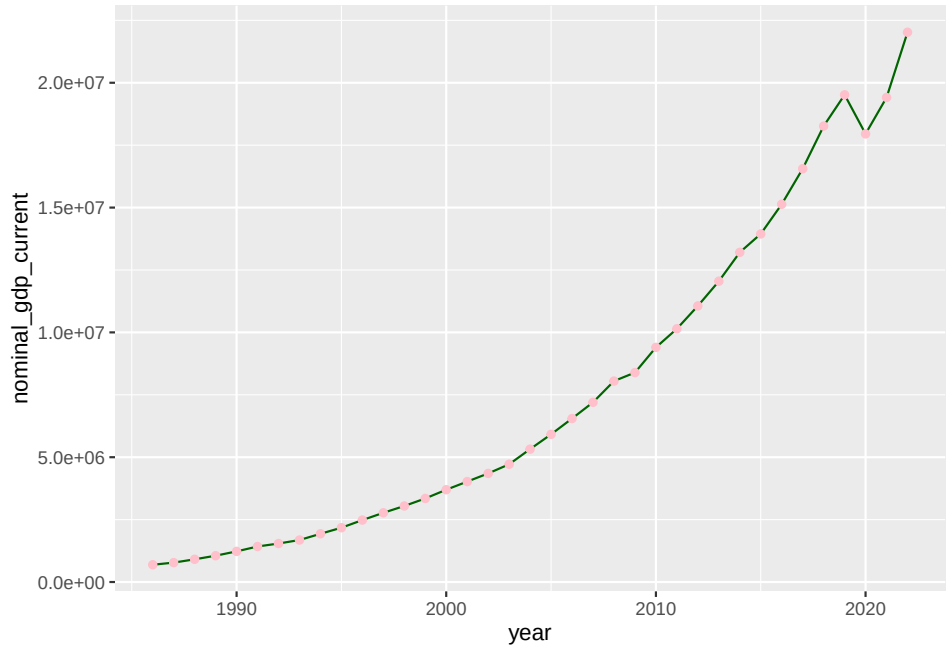
You might wonder why mine does not have a warning. Usually, there will be warnings that will come out. To remove it, simply add inside the `{r}`, `warning=FALSE` like: `{r,warning=FALSE}`

```
ch4_p2 %>%
  ggplot(aes(x=year, y=nominal_gdp_current))+
  geom_line(color="darkgreen")+
  labs(title = "Time Series Plot of Nominal GDP (Current)", x = "Year", y = "Nominal
  theme_minimal()
```



We can add points for clarity.

```
ch4_p2 %>%
  ggplot(aes(x=year, y=nominal_gdp_current))+
  geom_line(color="darkgreen")+
  geom_point(color= "pink")
```




```
  labs(title = "Time Series Plot of Nominal GDP (Current)", x = "Year", y = "Nominal GDP
```

```
## $x
## [1] "Year"
##
## $y
## [1] "Nominal GDP (Current)"
##
## $title
## [1] "Time Series Plot of Nominal GDP (Current)"
##
## attr(,"class")
## [1] "labels"
```

9.4 Practical: Visualizations

For this practical, we will use the `quotas` dataset. Please read the quotas_codebook to understand what each column means. Both files are found in the modules.

1. Inspect the dataset by using `str()` and `summary()` and describe the types of variables available.
2. Plot a histogram of the total population (`tot_pop71_true`). Adjust the number of bins to 15. What does this tell you about the distribution of population sizes in 1971?
3. Plot a bar plot of the count of Assembly Constituencies (`AC_type_noST`) based on reservation status. Which category has the highest count?
4. Plot the literacy rates (`Plit71`) against employment rates (`P_W71`). Add a regression line using `geom_smooth()`. What relationship do you observe?
5. Add color to the scatter plot by using color to distinguish between different reservation statuses (`AC_type_noST`). How do the patterns differ across groups?
6. Create a box plot of literacy rates (`Plit71`) for different reservation statuses (`AC_type_noST`). What do you notice about the spread of literacy rates across categories?
7. Create a faceted scatter plot of literacy rates (`Plit71`) vs. employment rates (`P_W71`), grouped by reservation status (`AC_type_noST`). What differences can you identify between the facets?
8. Create a pie chart showing the proportion of constituencies by reservation status (`AC_type_noST`). What does this chart reveal about the

dataset?

9. What did you find most challenging or rewarding about working with this dataset? Which visualization techniques did you find most useful for communicating your insights?
10. For the time series plot, I want you to search the **Literacy rate, adult total (% of people ages 15 and above)**from the World Bank. I want you to download as CSV the data. Then, I want you to choose India only. Create a time series plot for India.

9.5 Feedback: Practical Visualizations

For the Quarto Markdown files, you can include texts, like you are just regularly typing in Word. Only have the codes in `code chunks` then the analysis as text, rather than as comments.

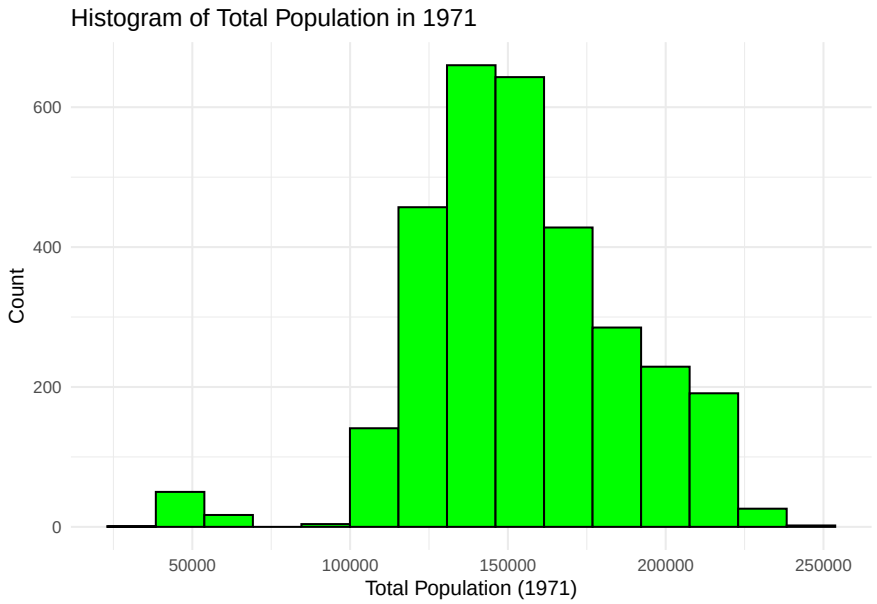
1. Use `str()` and `summary()` by loading necessary libraries: `ggplot2` and `dplyr`

```
## 'data.frame':    3134 obs. of  21 variables:
##  $ State_number_2001: int  2 2 2 2 2 2 2 2 2 2 ...
##  $ AC_no_2001       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ AC_number_1976   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ State_name_1976  : chr  "Himachal Pradesh " "Himachal Pradesh " "Himachal Prade
##  $ AC_name_1976     : chr  "Kinnaur" "Rampur" "Rohru" "Jubbal-Kotkhai" ...
##  $ AC_type_noST     : chr  NA "SC" "GEN" "GEN" ...
##  $ tot_pop71_true   : int  49835 56788 58969 50083 46745 53083 47237 55368 51571 4
##  $ SC_percent71_true: num  19.4 33.1 29.5 24.7 28.9 ...
```

```
##  $ Plit71           : num  27.7 25.7 18.9 30.8 18.5 ...
##  $ Plit71_SC        : num  14.07 13.67 5.64 18.19 11.15 ...
##  $ P_W71            : num  60.5 52.3 34.1 50.7 60.1 ...
##  $ P_al71_SC        : num  12.4 1.65 6.98 8.06 2.89 ...
##  $ P_al71_nonSC     : num  4.29 1.05 8.29 8.17 1.61 ...
##  $ P_elecVD01       : num  100 100 98.3 100 100 ...
##  $ P_elecVD01_sc    : num  100 100 97.8 100 100 ...
##  $ P_educVD01       : num  93.1 92.9 86.9 72.2 79.1 ...
##  $ P_educVD01_sc    : num  94.8 95.8 87.4 74 79 ...
##  $ P_medicVD01      : num  60.4 44.4 34 27.1 25.7 ...
##  $ P_medicVD01_sc   : num  58.7 48.7 32.9 29.9 25.4 ...
##  $ P_commVD01       : num  94.7 70.9 74.7 91.9 52 ...
##  $ P_commVD01_sc    : num  90.2 71.8 77.3 91.8 54 ...

##  State_number_2001   AC_no_2001    AC_number_1976   State_name_1976    AC
##  Min.   : 2.00     Min.   :  1.0   Min.   :  1.00   Length:3134        Le
##  1st Qu.: 9.00     1st Qu.: 49.0   1st Qu.: 56.25   Class :character   Cl
##  Median :22.00     Median :101.0   Median :117.00   Mode  :character   Mo
##  Mean   :19.47     Mean   :119.5   Mean   :133.50
##  3rd Qu.:28.00     3rd Qu.:178.0   3rd Qu.:197.00
##  Max.   :33.00     Max.   :403.0   Max.   :419.00
##
##  tot_pop71_true   SC_percent71_true     Plit71         Plit71_SC
##  Min.   : 27568   Min.   : 0.2017   Min.   : 2.742   Min.   : 1.966   Min
##  1st Qu.:133962   1st Qu.: 8.9001   1st Qu.:18.689   1st Qu.: 8.048   1st
##  Median :151681   Median :14.7556   Median :25.720   Median :13.305   Med
##  Mean   :154329   Mean   :15.2454   Mean   :28.525   Mean   :16.304   Mea
```

```
##  3rd Qu.:174641   3rd Qu.:20.0968   3rd Qu.:35.787   3rd Qu.:21.488   3rd Qu.:36.7
##  Max.   :242840   Max.   :66.5320   Max.   :78.294   Max.   :66.887   Max.   :60.5
##
##    P_elecVD01     P_elecVD01_sc      P_educVD01     P_educVD01_sc     P_medicVD01
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   :  0.0
##  1st Qu.: 89.02   1st Qu.: 90.53   1st Qu.: 93.54   1st Qu.: 93.65   1st Qu.: 39.9
##  Median : 99.54   Median : 99.78   Median : 98.26   Median : 98.60   Median : 60.5
##  Mean   : 89.15   Mean   : 89.70   Mean   : 95.03   Mean   : 95.24   Mean   : 61.0
##  3rd Qu.:100.00   3rd Qu.:100.00   3rd Qu.: 99.86   3rd Qu.: 99.99   3rd Qu.: 83.2
##  Max.   :100.00   Max.   :100.00   Max.   :100.00   Max.   :100.00   Max.   :100.0
##  NA's   :58       NA's   :58       NA's   :58       NA's   :58       NA's   :58
##  P_commVD01_sc
##  Min.   :  0.00
##  1st Qu.: 54.51
##  Median : 76.40
##  Mean   : 72.70
##  3rd Qu.: 94.44
##  Max.   :100.00
##  NA's   :58
```

2. Plot a histogram of the total population (`tot_pop71_true`). Adjust the number of bins to 15. What does this tell you about the distribution of population sizes in 1971?

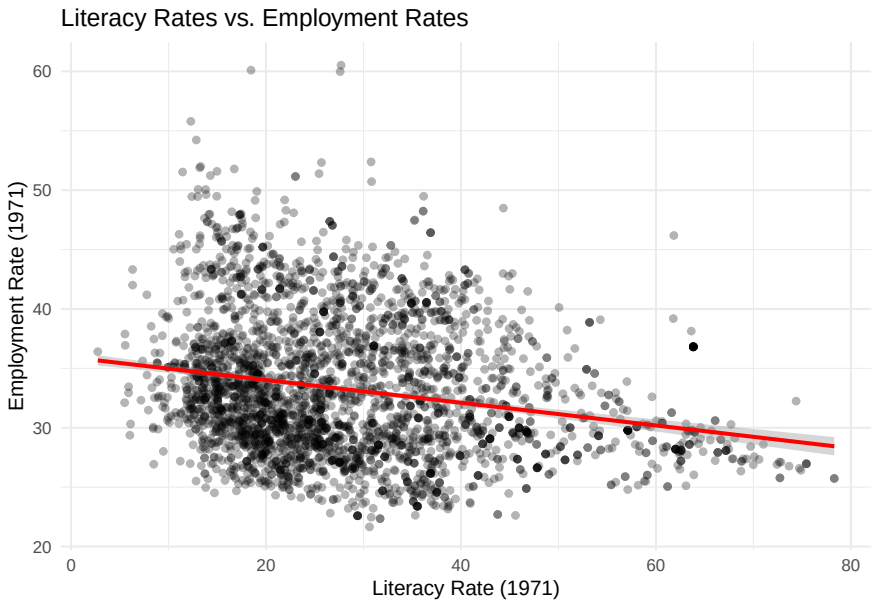


3. Plot a bar plot of the count of Assembly Constituencies (`AC_type_noST`) based on reservation status. Which category has the highest count?

```
{{r, echo=FALSE,eval=TRUE}} # Bar plot of count of Assembly
Constituencies by reservation status ggplot(quotas, aes(x =
factor(AC_type_noST))) +   geom_bar(fill = "darkgreen", color
= "black") +   labs(title = "Count of Assembly Constituencies
by Reservation Status",        x = "Reservation Status",
y = "Count") +   theme_minimal()
```

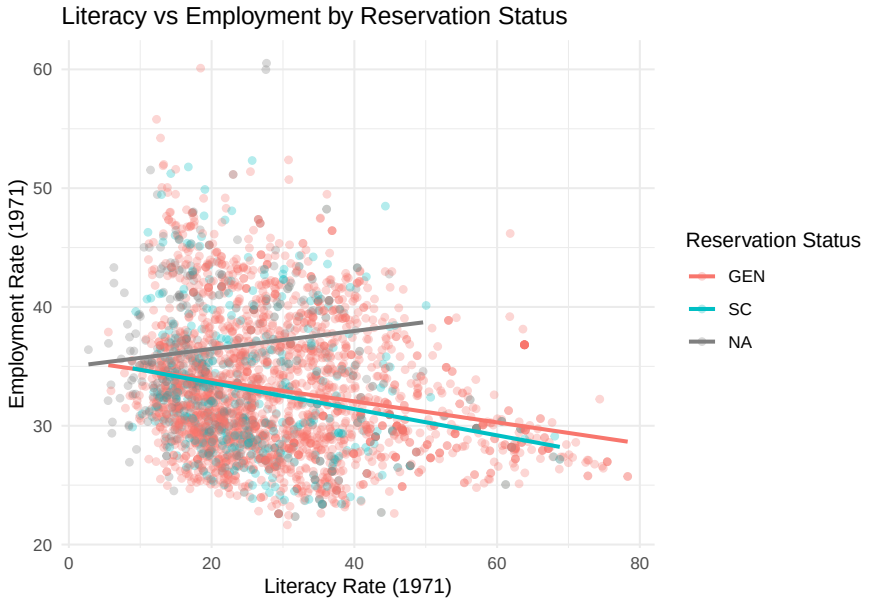
4. Plot the literacy rates (`Plit71`) against employment rates (`P_W71`). Add a regression line using `geom_smooth()`. What relationship do you observe?

```
## `geom_smooth()` using formula = 'y ~ x'
```

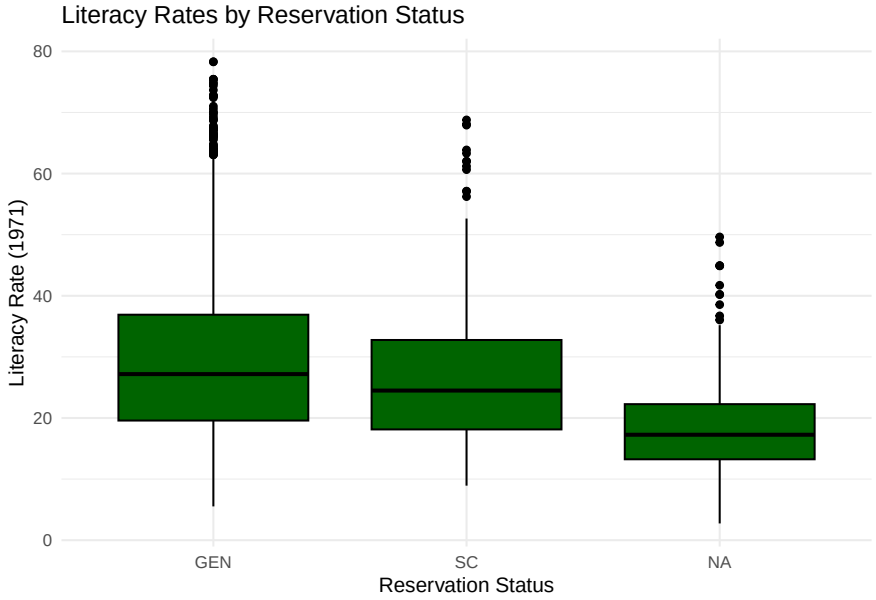


5. Add color to the scatter plot by using color to distinguish between different reservation statuses (`AC_type_noST`). How do the patterns differ across groups?

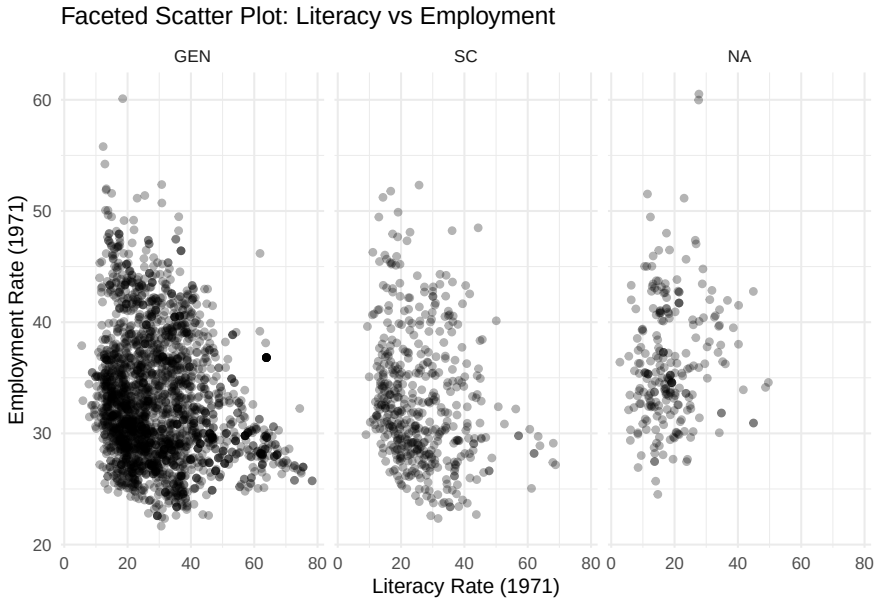
```
## `geom_smooth()` using formula = 'y ~ x'
```



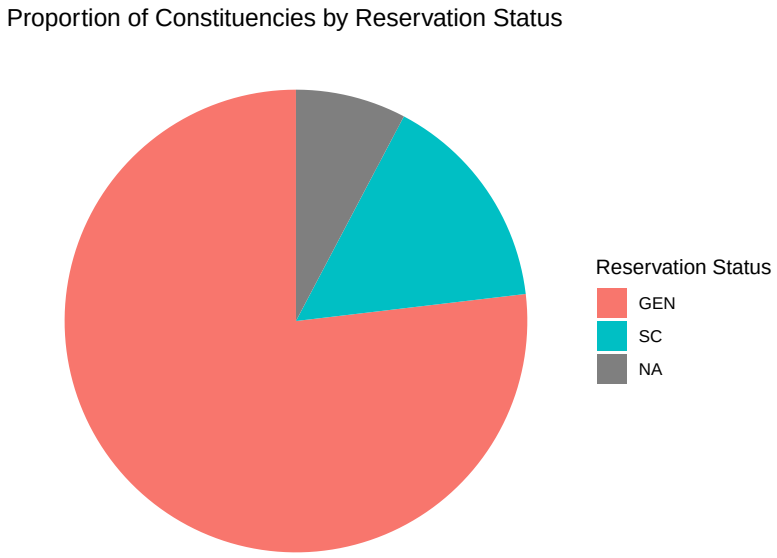
6. Create a box plot of literacy rates (`Plit71`) for different reservation statuses (`AC_type_noST`). What do you notice about the spread of literacy rates across categories?



7. Create a faceted scatter plot of literacy rates (`Plit71`) vs. employment rates (`P_W71`), grouped by reservation status (`AC_type_noST`). What differences can you identify between the facets?



8. Create a pie chart showing the proportion of constituencies by reservation status (`AC_type_noST`). What does this chart reveal about the dataset?



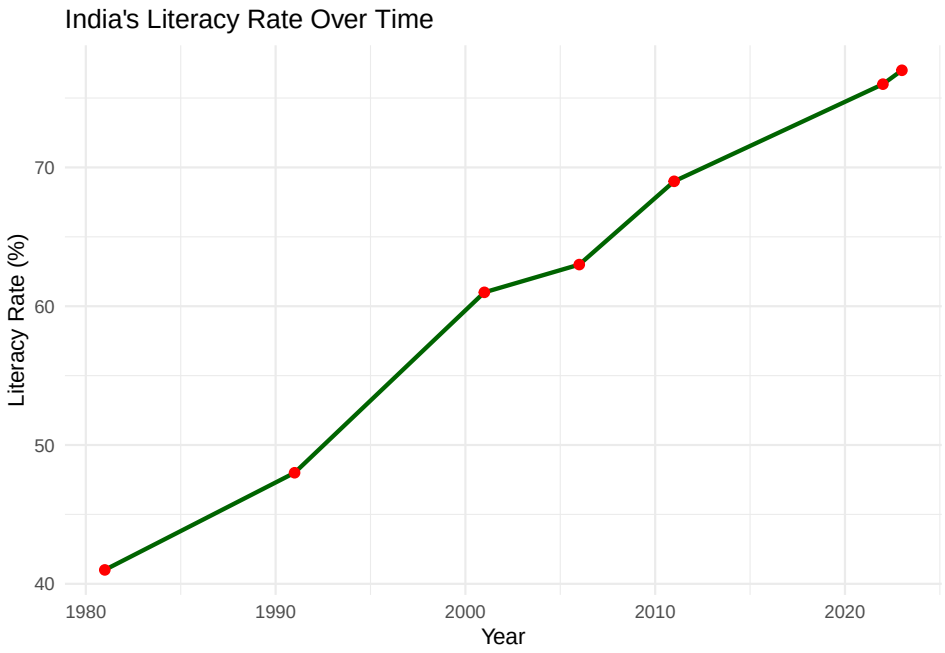
9. For the time series plot, I want you to search the **Literacy rate, adult total (% of people ages 15 and above)**from the World Bank. I want you to download as CSV the data. Then, I want you to choose India only. Create a time series plot for India.

   Steps include:

   1. Loading the dataset
   2. Filtering the dataset to just include India
   3. Analyze the data; you will notice that it is in the wide dataset; it is therefore important to modify the dataset to long dataset for visualization
   4. Before doing the visualization, also notice that the time has a character "X" and some of the literacy rate does not have any data. Remove the "X" and the "NA"

```r
library(tidyr)
library(dplyr)

# Load the dataset
world_bank_data <- read.csv("API_SE.ADT.LITR.ZS_DS2_en_csv_v2_160.csv", skip = 4)

# Filter data for India
india_data <- world_bank_data %>% filter(`Country.Name` == "India")

# Convert wide format to long format
india_long <- india_data %>%
  select(-c(`Country.Code`, `Indicator.Name`, `Indicator.Code`)) %>%
  pivot_longer(cols = -`Country.Name`, names_to = "Year", values_to = "Literacy Rate")

#  Remove "X" prefix from Year and convert to numeric
india_long$Year <- as.numeric(gsub("X", "", india_long$Year))

# Remove missing values
india_long <- india_long %>% filter(!is.na(`Literacy Rate`))

ggplot(india_long, aes(x = Year, y = `Literacy Rate`)) +
  geom_line(color = "darkgreen", size = 1) +
  geom_point(color = "red", size = 2) +
  labs(title = "India's Literacy Rate Over Time",
       x = "Year",
       y = "Literacy Rate (%)") +
```

```
  theme_minimal()
```



Chapter 10

Advanced Visualizations

10.1 Preliminaries

10.1.1 Install Packages

For this lecture, you need to install a lot of packages. Please do this before our lecture as it will take a long time to install them. Furthermore, there might be issues in installing, such as needing to install other packages. Please let the professor know if you encounter any issues.

```
# Install necessary packages
if (!require(gganimate)) install.packages("gganimate")
if (!require(ggplot2)) install.packages("ggplot2")
if (!require(dplyr)) install.packages("dplyr")
if (!require(sf)) install.packages("sf")
if (!require(rnaturalearth)) install.packages("rnaturalearth")
if (!require(readr)) install.packages("readr")
```

```
if (!require(tidyr)) install.packages("tidyr")
if (!require(gifski)) install.packages("gifski")
if (!require(WDI)) install.packages("WDI")
if (!require(leaflet)) install.packages("leaflet")

#Load packages
library(gganimate)
library(ggplot2)
library(dplyr)
library(sf)
library(rnaturalearth)
library(readr)
library(tidyr)
library(gifski)
library(WDI)
library(leaflet)
```

10.2 Animated Visualizations

`gganimate` includes animation to `ggplot2`; It adds some classes to the plot object in order to customise how it should change with time.

- `transition_*()` defines how the data should be spread out and how it relates to itself across time.
- `view_*()` defines how the positional scales should change along the animation.

- `shadow_*()` defines how data from other points in time should be presented in the given point in time.
- `enter_*()`/`exit_*()` defines how new data should appear and how old data should disappear during the course of the animation.
- `ease_aes()` defines how different aesthetics should be eased during transitions

  For this lecture, we will use the `quotas` dataset and fetch some World Bank Development Indicators from the `WDI` package.

10.2.1 Animated Bar Chart

At the start, you need to do the same steps as that of when doing the visualizations *without* animations.

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

```
##            used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells  5655410 302.1    9801260 523.5  9801260 523.5
## Vcells 11545967  88.1   36965975 282.1 94237569 719.0
```

```
# Load the dataset
quotas <- read.csv("quotas.csv")


ch5.1<-quotas %>%
  ggplot(aes(x=factor(AC_type_noST)))+
```

```
  geom_bar(fill="darkgreen", color="black")+
  labs(title = "Animated Bar Chart: Assembly Constituencies",
       x="Reservation Status", y="Count")+
  theme_minimal()+
  transition_states(AC_type_noST, transition_length = 2, state_length = 1)
```

The `transition_states` animates transitions between different categorical states. The `AC_type_noST` is the categorical variable that defines different animation states. The `transition_length` controls how long the transition between states lasts, measured in animation frames. While the `state_length` defines how long each state remains static before transitioning to the next one.

10.2.1.1 Saving and Embedding Animation

This is very important; it is different for Quarto where the gif is automatically rendered; however, later, we will find out how to save in Quarto.

```
anim_file <- "bar_chart.gif"
animate(ch5.1, duration = 4, fps = 10, renderer = gifski_renderer(anim_file),
        preview = TRUE)  # Preview the animation before rendering
# Save animation as a GIF
knitr::include_graphics(anim_file)
```

The animate function generates the animation of the ggplot object `ch5.1`. `duration = 4`: Specifies that the animation should run for 4 seconds in total. `fps = 10`: Defines the frame rate, meaning the animation will

show 10 frames per second. `renderer = gifski_renderer(anim_file)`: Uses the `gifski_renderer` to save the animation as a GIF file named `bar_chart.gif`. `preview = TRUE`: Allows you to view the animation immediately in the RStudio Viewer before saving it as a file.

10.2.2 Animated Scatter Plot

```
ch5.2<-quotas %>%
  ggplot(aes(x=Plit71, y=P_W71))+
  geom_point(aes(color=factor(AC_type_noST)))+
  labs(title="Animated Scatter Plot: Literacy vs. Employment",
       x="Literacy Rate (1971)", y="Employment Rate (1971)", color="Reservation Status")+
  theme_minimal()+
  transition_reveal(Plit71)
```

Here, the `transition_reveal` animates the points to be revealed over literacy rates

10.2.2.1 Saving and Embedding Animation

```
anim_file2<-"scatter_plot.gif"
animate(ch5.2, duration=20, fps=10, renderer=gifski_renderer(anim_file2),
        preview=TRUE)
knitr::include_graphics(anim_file2)
```

To slow down the animation, increase the duration and decrease fps.

10.2.3 Animated Faceted Scatter Plot

10.2.3.1 Fetching WDI Data

For this portion, we will make use of the WDI database. To search which indicator you wish to work with, type `WDIsearch("keyword")`

```
ch5<-WDI(
  country = c("USA","CHN", "IND", "BRA","NLD", "JPN"),
  indicator = c("NY.GDP.PCAP.CD", "SP.DYN.LE00.IN"),
  start = 2000,
  end = 2020,
  extra = TRUE
)
```

This code chunk pulls data from the World Bank WDI database of 6 countries and two indicators (GDP per capita (current US$) and Life Expectancy at Birth (years). It fetches data from 2000 to 2020 and includes extra metadata such as region names and income levels.

```
#Cleaning the dataset
ch5 <- ch5 %>%
  rename(GDPpc = NY.GDP.PCAP.CD, Life_Exp = SP.DYN.LE00.IN)
ch5<-ch5 %>%
  filter(!is.na(GDPpc), !is.na(Life_Exp))
head(ch5)
```

```
##   country iso2c iso3c year status lastupdated    GDPpc Life_Exp
```

```
## 1  Brazil    BR   BRA 2000         2025-01-28 3766.548   69.737 Latin America & Caribbe
## 2  Brazil    BR   BRA 2001         2025-01-28 3176.289   70.195 Latin America & Caribbe
## 3  Brazil    BR   BRA 2002         2025-01-28 2855.940   70.410 Latin America & Caribbe
## 4  Brazil    BR   BRA 2003         2025-01-28 3090.607   70.720 Latin America & Caribbe
## 5  Brazil    BR   BRA 2004         2025-01-28 3663.823   71.131 Latin America & Caribbe
## 6  Brazil    BR   BRA 2005         2025-01-28 4827.782   71.753 Latin America & Caribbe
##                income lending
## 1 Upper middle income    IBRD
## 2 Upper middle income    IBRD
## 3 Upper middle income    IBRD
## 4 Upper middle income    IBRD
## 5 Upper middle income    IBRD
## 6 Upper middle income    IBRD
```

```
ch5.3<-ch5 %>%
  ggplot(aes(x=GDPpc, y=Life_Exp))+
  geom_point(aes(color=region), alpha=0.7, size=3)+
  labs(title="Faceted Scatter Plot: GDP vs. Life Expectancy",
       subtitle = "Year: 2000-2020",
       x="GDP per Capita (USD)",
       y="Life Expectancy (Years)",
       color="Region")+
  theme_minimal()+
  facet_wrap(~country, ncol=3)+
  scale_x_log10()+ #log scale for better visualization
  transition_states(year, transition_length = 2, state_length = 1)
```

There are additional things here like the `size=3` which changes the size of points. the `scale_x_log10()` was added because it applies a logarithmic scale to the x-axis since GDP per capital values vary widely, and a log scale makes comparisons clearer. The `transition_states` has `year` since each frame would represent a different year.

10.2.3.2 Saving and Embedding Animation

```
anim_file3<-"faceted_scatter_plot.gif"
animate(ch5.3, duration=10, fps=15, renderer=gifski_renderer(anim_file3),
        preview=TRUE)
knitr::include_graphics(anim_file3)
```

This does not really provide any information. Let us try visualizing in a different way.

10.3 Animated Time Series

Let us just choose USA for this.

```
ch5_1<-ch5 %>%
  filter(iso3c=="USA")
```

```
ch5.4<-ch5_1 %>%
  ggplot(aes(x=year, y=GDPpc))+
  geom_line(color="green", size = 1.2)+
  geom_point(color="purple", size=2)+
```

```
  labs(title = "US GDP per Capita Growth over Time",
       subtitle = "Year:2000-2020",
       x="Year",
       y="GDP per Capita (Current US$)")+
  theme_minimal()+
  transition_reveal(year)
```

10.3.0.1 Saving and Embedding Animation

```
anim_file4<-"us_timeseries.gif"
animate(ch5.4, duration = 10, fps = 15, renderer = gifski_renderer(anim_file4),
        preview = TRUE)
knitr::include_graphics(anim_file4)
```

10.4 Animated Faceted Time Series

```
ch5.5<-ch5 %>%
  ggplot(aes(x=year, y=GDPpc, group=country))+
  geom_line(aes(color=country), size=1.2)+
  labs(title = "Faceted Time Series: GDP Growth Over Time",
       subtitle = "Year: 2000-2020",
       x="Year",
       y="GDP (Current US$)",
       color="Country")+
  theme_minimal()+
```

```
  facet_wrap(~country, scales = "free_y")+ #allows free-scaling in the y-axis
  transition_reveal(year)
```

10.4.0.1 Saving and Embedding Animation

```
anim_file5<-"timeseries.gif"
animate(ch5.5, duration = 10, fps = 15, renderer = gifski_renderer(anim_file5),
        preview = TRUE)
knitr::include_graphics(anim_file5)
```

10.4.1 Saving in Quarto

For Data Presentation, you need to save interactive visualizations so that you can place them in your presentations.

```
# Create an animated plot
p <- ggplot(DATA, aes(MAPPINGS)) +
  geom_function +
  transition_states(gear, transition_length = 2, state_length = 1)

# Save as GIF
anim_save("animation.gif", p)

# Save as MP4
anim_save("animation.mp4", p)
```

10.5 Maps

10.5.1 Where to get shapefiles?

The `rnaturalearth` provides some shapefiles you can use. A suggested site to find shapefiles of different countries: https://gadm.org/index.html . You can also locate shapefiles from the government sites.

We will make use of the packages like `WDI`, `rnaturalearth`,`sf`, `gganimate`, and `leaflet`. Do note however, that gganimate and leaflet does not work for PDF, thus, it can only be used for your Data Story Presentation.

10.6 Static Maps

10.6.1 Fetching GDP Data from WDI

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

```
##            used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells  5655607 302.1    9801260 523.5  9801260 523.5
## Vcells 11548614  88.2   36965975 282.1 94237569 719.0
```

```
ch5_2<-WDI(country = "all",
           indicator = "NY.GDP.MKTP.CD",
           start = 2000,
           end = 2022,
```

```
            extra=TRUE)

#clean the dataset
ch5_2<-ch5_2 %>%
  rename(gdp=NY.GDP.MKTP.CD,
         iso_a3=iso2c) %>%
  drop_na(gdp) #dropping missing values
```

We are retrieving GDP data for all countries from 2000 to 2022 and we are also renaming columns to match with map data found in the `rnaturalearth`. We also remove missing GDP values

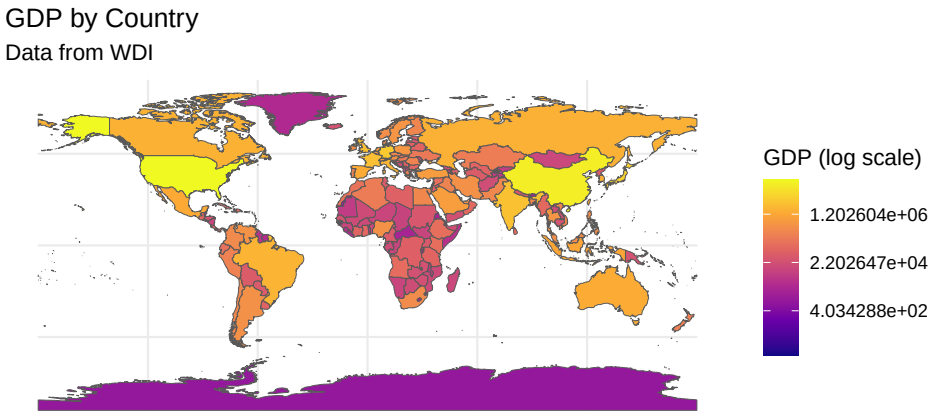
```
ch5map<-ne_countries(scale = "medium", returnclass = "sf")
```

This fetches the world map with country boundaries in `sf` (simple features) format to visualize the GDP data. We fetch medium-scale country boundaries in spatial data format and the `returnclass="sf"` ensures it can be used with `ggplot2`

```
#Merge
wgdp<-ch5map %>%
  left_join(ch5_2, by="iso_a3")
```

We need to match the GDP data with the corresponding country for visualization. You can opt to retrieve the world map to find out how to merge both.

```
ggplot(data = wgdp) +
  geom_sf(aes(fill = gdp_md)) +
  scale_fill_viridis_c(option = "plasma", trans = "log", na.value = "grey") +
  theme_minimal() +
  labs(title = "GDP by Country",
       subtitle = "Data from WDI",
       fill = "GDP (log scale)")
```



We use the merged map and GDP data and `geom_sf(aes(fill=gdp_md))` filles countries based on GDP. The `scale_fill_viridis_c` uses "plasma" which is best used for maps. The log scale transformation is used to better differentiate large economies and small economies and grey fill for missing data.

10.6.1.1 Animated World Map for GDP

Now we create an animated version of the map. You need to check that your time variable does not have any NA. It would be better to choose the years that are available consecutively so we will filter the data to only include from 2009-2019.

```
unique(wgdp$gdp_year)
```

```
##  [1] 2019 2014 2018 2017 2016 2013 2010 2009 2012 2006 2015 2003 2007 2011
```

```
# Filter for years 2009-2019
wgdp_filtered <- wgdp %>%
  filter(gdp_year %in% 2009:2019)
```

```
ch5.6 <- ggplot(data = wgdp_filtered) +
  geom_sf(aes(fill = gdp_md)) +
  scale_fill_viridis_c(option = "plasma", trans = "log", na.value = "grey50") +
  theme_minimal() +
  labs(title = "World GDP by Country: {closest_state}",
       subtitle = "Data from World Development Indicators",
       fill = "GDP (log scale)") +
  transition_states(gdp_year) +
  ease_aes('linear')
```

10.6.1.2 Saving and Embedding Animation

```
anim_file6<-"worldgdp.gif"
animate(ch5.6, duration = 10, fps = 15, renderer = gifski_renderer(anim_file6),
        preview = TRUE)
knitr::include_graphics(anim_file6)
```

10.7 Philippine Regional Map

We will use `Ch6.xlsx` containing Regional GDP and use the Philippine Regions shapefile.

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

```
##            used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells  5656740 302.2    9801260 523.5  9801260 523.5
## Vcells 11589233  88.5   36965975 282.1 94237569 719.0
```

```
library(sf)
layers<-st_layers("PH_Adm1_Regions.shp")
print(layers)
```

```
## Driver: ESRI Shapefile
## Available layers:
##            layer_name geometry_type features fields                crs_name
```

```
## 1 PH_Adm1_Regions.shp       Polygon       17      7 WGS 84 / UTM zone 51N
## 2         Regions.shp       Polygon       17     19                WGS 84
```

```
sp_df = sf::st_read(
  dsn= 'PH_Adm1_Regions.shp',
  layer="Regions.shp")
```

```
## Reading layer `Regions.shp' from data source `C:\Users\jemma\OneDrive\Document
## Simple feature collection with 17 features and 19 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 114.2779 ymin: 4.587294 xmax: 126.605 ymax: 21.12189
## Geodetic CRS:  WGS 84
```

The code loads the `sf` package then `st_layers` checks all available layers in the shapefile because some shapefiles contains multiple layers like administrative boundaries, water bodies, etc. so we display the available layers in the console. The `st_read(dsn "PH_Adm1_Regions.shp")` specifies the data source and the `layer = "Regions.shp"` specifies the layer name to read (the one we will use and need).

Let us check if the shapefile was loaded correctly.

```
print(sp_df) #check structure
```

```
## Simple feature collection with 17 features and 19 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
```

```
## Bounding box:  xmin: 114.2779 ymin: 4.587294 xmax: 126.605 ymax: 21.12189
## Geodetic CRS:  WGS 84
## First 10 features:
##    psgc_code                            name corr_code geo_level city_class inc_class u
## 1      1e+08        Region I (Ilocos Region)     1e+07       Reg       <NA>      <NA>
## 2      2e+08      Region II (Cagayan Valley)     2e+07       Reg       <NA>      <NA>
## 3      3e+08      Region III (Central Luzon)     3e+07       Reg       <NA>      <NA>
## 4      4e+08        Region IV-A (CALABARZON)     4e+07       Reg       <NA>      <NA>
## 5      5e+08         Region V (Bicol Region)     5e+07       Reg       <NA>      <NA>
## 6      6e+08     Region VI (Western Visayas)     6e+07       Reg       <NA>      <NA>
## 7      7e+08    Region VII (Central Visayas)     7e+07       Reg       <NA>      <NA>
## 8      8e+08   Region VIII (Eastern Visayas)     8e+07       Reg       <NA>      <NA>
## 9      9e+08 Region IX (Zamboanga Peninsula)     9e+07       Reg       <NA>      <NA>
## 10     1e+09    Region X (Northern Mindanao)     1e+08       Reg       <NA>      <NA>
##    adm1_pcode                           adm1_en            adm1_alt adm0_pcode
## 1         PH01        Region I (Ilocos Region)       Ilocos Region         PH Philippine
## 2         PH02      Region II (Cagayan Valley)      Cagayan Valley         PH Philippine
## 3         PH03      Region III (Central Luzon)       Central Luzon         PH Philippine
## 4         PH04        Region IV-A (Calabarzon)          Calabarzon         PH Philippine
## 5         PH05         Region V (Bicol Region)        Bicol Region         PH Philippine
## 6         PH06     Region VI (Western Visayas)     Western Visayas         PH Philippine
## 7         PH07    Region VII (Central Visayas)     Central Visayas         PH Philippine
## 8         PH08   Region VIII (Eastern Visayas)     Eastern Visayas         PH Philippine
## 9         PH09 Region IX (Zamboanga Peninsula) Zamboanga Peninsula         PH Philippine
## 10        PH10    Region X (Northern Mindanao)   Northern Mindanao         PH Philippine
##    shape_area shape_sqkm                       geometry
## 1    1.043983   12307.35 MULTIPOLYGON (((120.9714 18...
```

```
## 2    2.241812   26387.73 MULTIPOLYGON (((121.9488 21...
## 3    1.793513   21304.16 MULTIPOLYGON (((122.2342 16...
## 4    1.326720   15846.63 MULTIPOLYGON (((122.3079 14...
## 5    1.446324   17338.38 MULTIPOLYGON (((122.9882 11...
## 6    1.657591   20047.63 MULTIPOLYGON (((121.4341 12...
## 7    1.178431   14293.66 MULTIPOLYGON (((124.093 11....
## 8    1.726804   20835.68 MULTIPOLYGON (((124.3678 12...
## 9    1.196677   14596.05 MULTIPOLYGON (((123.4129 8....
## 10   1.435115   17489.29 MULTIPOLYGON (((124.6987 9....
```

We then check available attributes and also check the region names so that we can merge the shapefile with our GDP data.

```
colnames(sp_df)
```

```
##  [1] "psgc_code"  "name"       "corr_code"  "geo_level"  "city_class" "inc_cla
##  [9] "pop_2020"   "status"     "adm1_pcode" "adm1_en"    "adm1_alt"   "adm0_pc
## [17] "shape_len"  "shape_area" "shape_sqkm" "geometry"
```

```
head(sp_df)
```

```
## Simple feature collection with 6 features and 19 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 119.7497 ymin: 9.444193 xmax: 124.4249 ymax: 21.12189
## Geodetic CRS:  WGS 84
##   psgc_code                        name corr_code geo_level city_class inc_cla
```

```
## 1     1e+08    Region I (Ilocos Region)     1e+07       Reg       <NA>      <NA>    <NA
## 2     2e+08  Region II (Cagayan Valley)     2e+07       Reg       <NA>      <NA>    <NA
## 3     3e+08  Region III (Central Luzon)     3e+07       Reg       <NA>      <NA>    <NA
## 4     4e+08    Region IV-A (CALABARZON)     4e+07       Reg       <NA>      <NA>    <NA
## 5     5e+08     Region V (Bicol Region)     5e+07       Reg       <NA>      <NA>    <NA
## 6     6e+08 Region VI (Western Visayas)     6e+07       Reg       <NA>      <NA>    <NA
##   adm1_pcode                     adm1_en        adm1_alt adm0_pcode           adm0_en
## 1       PH01    Region I (Ilocos Region)   Ilocos Region         PH Philippines (the) 1
## 2       PH02  Region II (Cagayan Valley)  Cagayan Valley         PH Philippines (the) 1
## 3       PH03  Region III (Central Luzon)   Central Luzon         PH Philippines (the) 1
## 4       PH04    Region IV-A (Calabarzon)      Calabarzon         PH Philippines (the) 1
## 5       PH05     Region V (Bicol Region)    Bicol Region         PH Philippines (the) 1
## 6       PH06 Region VI (Western Visayas) Western Visayas         PH Philippines (the) 1
##   shape_sqkm                       geometry
## 1   12307.35 MULTIPOLYGON (((120.9714 18...
## 2   26387.73 MULTIPOLYGON (((121.9488 21...
## 3   21304.16 MULTIPOLYGON (((122.2342 16...
## 4   15846.63 MULTIPOLYGON (((122.3079 14...
## 5   17338.38 MULTIPOLYGON (((122.9882 11...
## 6   20047.63 MULTIPOLYGON (((121.4341 12...
```

```
unique(sp_df$name)
```

```
##  [1] "Region I (Ilocos Region)"                    "Region II (Cagayan Val
##  [3] "Region III (Central Luzon)"                  "Region IV-A (CALABARZO
##  [5] "Region V (Bicol Region)"                     "Region VI (Western Vis
##  [7] "Region VII (Central Visayas)"                "Region VIII (Eastern V
```

```
##  [9] "Region IX (Zamboanga Peninsula)"                          "Region X (Nort
## [11] "Region XI (Davao Region)"                                 "Region XII (SO
## [13] "National Capital Region (NCR)"                            "Cordillera Adm
## [15] "Region XIII (Caraga)"                                     "MIMAROPA Regio
## [17] "Bangsamoro Autonomous Region In Muslim Mindanao (BARMM)"
```

```
sp_df <- sp_df %>%
  rename(region = name)
```

Now, we load our Regional GDP package;

```
library(readxl)
rgdp<-read_excel("Ch5PracticeB.xlsx",
                 sheet = "Sheet1")
```

```
## New names:
## * `` -> `...1`
```

```
head(rgdp)
```

```
## # A tibble: 6 x 26
##   ...1  region  `2000` `2001` `2002` `2003` `2004` `2005` `2006` `2007` `2008
##   <chr> <chr>    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl
## 1 NCR   Nation~ 2.42e9 2.48e9 2.51e9 2.62e9 2.84e9 2.99e9 3.16e9 3.37e9 3.52e9
## 2 CAR   Cordil~ 1.44e8 1.49e8 1.56e8 1.64e8 1.73e8 1.76e8 1.83e8 1.96e8 2.03e8
## 3 I     Ilocos~ 2.40e8 2.45e8 2.53e8 2.64e8 2.78e8 2.91e8 3.08e8 3.26e8 3.36e8
## 4 II    Cagaya~ 1.58e8 1.64e8 1.66e8 1.71e8 1.86e8 1.83e8 2.01e8 2.14e8 2.19e8
## 5 III   Centra~ 7.07e8 7.49e8 7.92e8 8.26e8 8.52e8 8.86e8 9.27e8 9.82e8 1.03e9
```

```
## 6 IVA   CALABA~ 1.05e9 1.07e9 1.12e9 1.18e9 1.24e9 1.31e9 1.37e9 1.45e9 1.50e9 1.49e9
## # i 9 more variables: `2015` <dbl>, `2016` <dbl>, `2017` <dbl>, `2018` <dbl>, `2019` <
## #   `2022` <dbl>, `2023` <dbl>
```

We need to convert the columns to have the same format first so that we can edit all of them later on when we transform to long format since different formats cannot be combined when modifying to long format.

```
colnames(rgdp)<-as.character(colnames(rgdp))
rgdp<-rgdp %>% select(-"...1")
```

Now, we can modify

```
rgdpl<-rgdp %>%
  pivot_longer(cols = -region, names_to = "Year", values_to = "GDP") %>%
  mutate(Year=as.numeric(Year),
         GDP=as.numeric(GDP))
head(rgdpl)
```

```
## # A tibble: 6 x 3
##   region                    Year         GDP
##   <chr>                    <dbl>       <dbl>
## 1 National Capital Region  2000 2416391870.
## 2 National Capital Region  2001 2483504980.
## 3 National Capital Region  2002 2507171644.
## 4 National Capital Region  2003 2624052475.
## 5 National Capital Region  2004 2841837836.
## 6 National Capital Region  2005 2988546218.
```

```
unique(rgdpl$region)
```

```
##  [1] "National Capital Region"                                  "Cordillera Administ
##  [3] "Ilocos Region"                                            "Cagayan Valley"
##  [5] "Central Luzon"                                            "CALABARZON"
##  [7] "MIMAROPA Region"                                          "Bicol Region"
##  [9] "Western Visayas"                                          "Central Visayas"
## [11] "Eastern Visayas"                                          "Zamboanga Peninsula
## [13] "Northern Mindanao"                                        "Davao Region"
## [15] "SOCCSKSARGEN"                                             "Caraga"
## [17] "Bangsamoro Autonomous Region\r\nin Muslim Mindanao"
```

Since we know that the names are different in the shapefile, we rename the regions in the `rgdpl`. I create a mapping table for this. You can use this again when you do your own mapping.

```
library(dplyr)

# Create a mapping table for inconsistent region names
region_mapping <- c(
  "National Capital Region" = "National Capital Region (NCR)",
  "Cordillera Administrative Region" = "Cordillera Administrative Region (CAR)",
  "Ilocos Region" = "Region I (Ilocos Region)",
  "Cagayan Valley" = "Region II (Cagayan Valley)",
  "Central Luzon" = "Region III (Central Luzon)",
  "CALABARZON" = "Region IV-A (CALABARZON)",
  "MIMAROPA Region" = "MIMAROPA Region",
```

```
  "Bicol Region" = "Region V (Bicol Region)",
  "Western Visayas" = "Region VI (Western Visayas)",
  "Central Visayas" = "Region VII (Central Visayas)",
  "Eastern Visayas" = "Region VIII (Eastern Visayas)",
  "Zamboanga Peninsula" = "Region IX (Zamboanga Peninsula)",
  "Northern Mindanao" = "Region X (Northern Mindanao)",
  "Davao Region" = "Region XI (Davao Region)",
  "SOCCSKSARGEN" = "Region XII (SOCCSKSARGEN)",
  "Caraga" = "Region XIII (Caraga)",
  "Bangsamoro Autonomous Region\r\nin Muslim Mindanao" = "Bangsamoro Autonomous Region In
)

# Apply corrections to GDP dataset
rgdpl$region <- recode(rgdpl$region, !!!region_mapping)

# Check if all regions now match
setdiff(rgdpl$region, sp_df$region)  # Should return an empty set if all match
```

Now, we can merge.

```
rgdp_map<-sp_df %>%
  left_join(rgdpl, by = "region")
setdiff(rgdpl$region, sp_df$region)
```

```
##  [1] "National Capital Region"                    "Cordillera Administrative R
##  [3] "Ilocos Region"                              "Cagayan Valley"
##  [5] "Central Luzon"                              "CALABARZON"
```

```
##  [7] "Bicol Region"                                  "Western Visayas"
##  [9] "Central Visayas"                               "Eastern Visayas"
## [11] "Zamboanga Peninsula"                           "Northern Mindanao"
## [13] "Davao Region"                                  "SOCCSKSARGEN"
## [15] "Caraga"                                        "Bangsamoro Autonomo
```

```
#remove unnecessary columns and mutate GDP to billions
rgdp_map <- rgdp_map %>%
  select(region, Year, GDP, geometry) %>%
  mutate(GDP = GDP / 1e9)
```

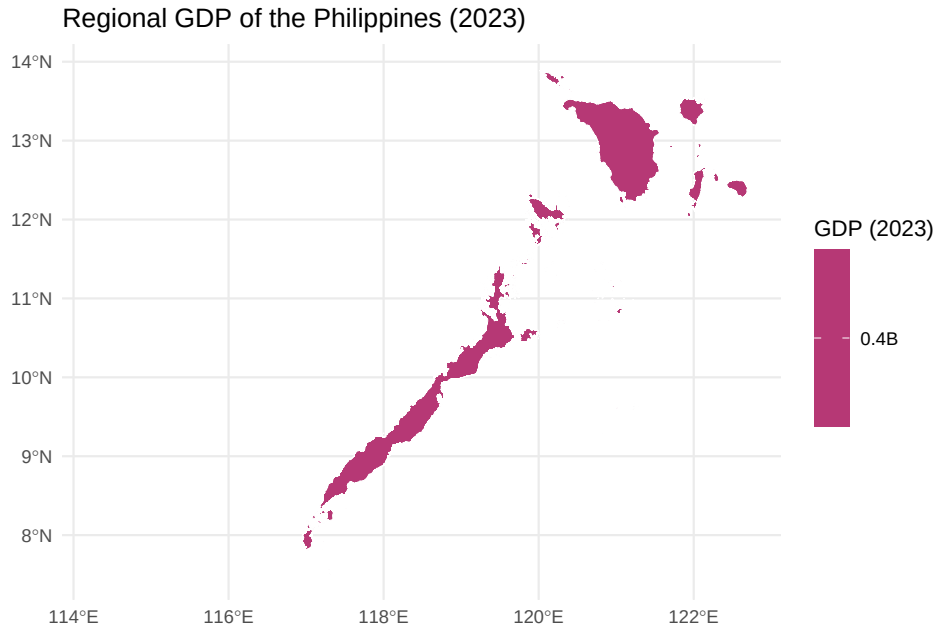
10.7.0.1 Choropleth Map

A Choropleth Map is a type of thematic map where regions are color-coded based on a statistical value (e.g., GDP, population, unemployment rate, etc.). It is commonly used in geographic data visualization to show spatial variations across different regions. For the static map, let us choose the latest year, 2023;

```
rgdp2023<-rgdp_map %>%
  filter(Year==2023)
```

```
library(scales)


# Modify the Choropleth Map
ggplot(rgdp2023) +
  geom_sf(aes(fill = GDP), color = "white") +  # Fill regions based on GDP
  scale_fill_viridis_c(
```

```
    option = "magma",
    name = "GDP (2023)",
    labels = label_number(accuracy = 0.1, suffix = "B", big.mark = ",")  # Fix rounding i
  ) +
  labs(title = "Regional GDP of the Philippines (2023)") +
  theme_minimal()
```




```
#Saving the images
#ggsave("rgdp2023.jpg", plot = last_plot(), width = 10, height = 8, dpi = 300)
#or plot = nameofplot, width and height are in inches and for high quality image, dpi = 3
```

10.7.0.2 Bubble Map

A Bubble Map is a type of thematic map that represents data values using circles (bubbles) placed over specific locations. The size of the bubble corresponds to the magnitude of the variable being visualized (e.g., GDP, population, sales, etc.). We use 2023 again.

The first step that we need to do is calculate the `centroids` using `st_centroid` from the `sf` package because this calculates the geometric center of each polygon region in the shapefile. These will be used as our bubble locations and to ensure that there will be no overlapping with borders.

```
phr_centroids<-st_centroid(rgdp2023)
```

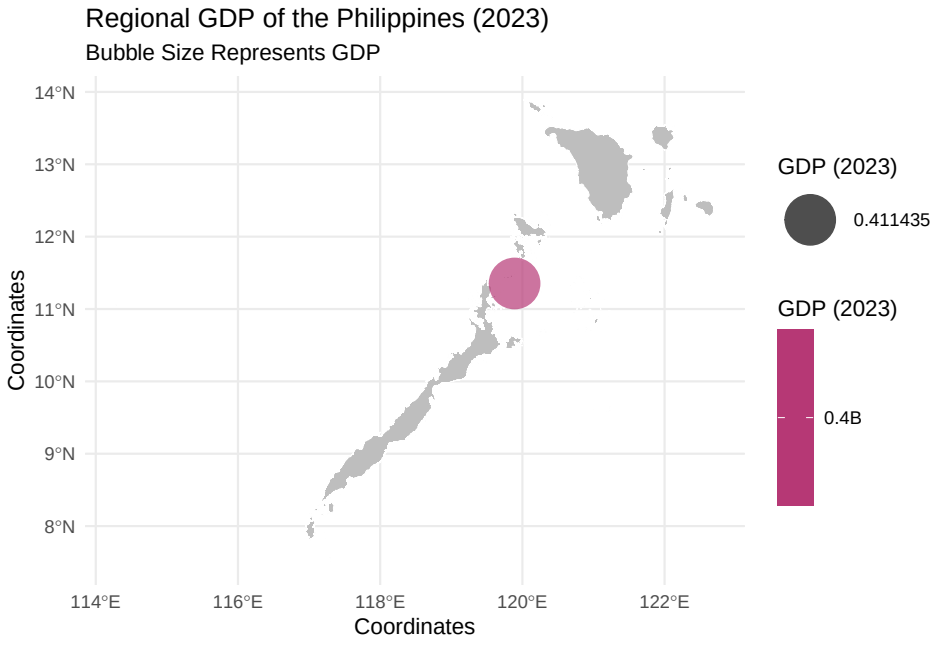
```
## Warning: st_centroid assumes attributes are constant over geometries
```

```
print(phr_centroids)
```

```
## Simple feature collection with 1 feature and 3 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 119.8902 ymin: 11.35282 xmax: 119.8902 ymax: 11.35282
## Geodetic CRS:  WGS 84
##            region Year      GDP                  geometry
## 1 MIMAROPA Region 2023 0.411435 POINT (119.8902 11.35282)
```

We now create our Bubble Map

```
ggplot()+
  geom_sf(data=rgdp2023, fill = "gray", color="white")+#base map
  geom_point(data = phr_centroids,
             aes(x = st_coordinates(geometry)[,1],
                 y = st_coordinates(geometry)[,2],
                 size = GDP,
                 color = GDP),
             alpha = 0.7) +
  scale_size(range = c(2, 15), name = "GDP (2023)") +  # Adjust bubble size
  scale_color_viridis_c(option = "magma", name = "GDP (2023)", labels = label_number(accu
  labs(title = "Regional GDP of the Philippines (2023)",
       subtitle = "Bubble Size Represents GDP",
       x="Coordinates",
       y="Coordinates")+
  theme_minimal()
```



As you can see, the base map is drawn using `geom_sf()` then the bubbles represent the GDP values. The bubble size and color represent GDP magnitude.

You will notice that `ggplot()` is empty, this creates an empty ggplot canvas where the map layers will be added using the `geom_sf`.

In the `aes` there's `x = st_coordinates(geometry)[,1]` same for y. These extract the longitude and latitude of each centroid.

The `scale_size()` range sets the bubble sizes between 2 and 15 units but you can edit this.

Now, `scale_viridis_c` has more things, the name sets the legend title for GDP colors. the labels ensures decimal precision and adds suffix B, plus, adds comma separators since we do not want the legend to contain scientific notations.

10.8 Animated Map

10.8.0.1 Animated Philippine Regional GDP Choropleth Map

We will now make an animated version of the Regional GDP Choropleth Map since we have data from 2000 to 2023. Let's convert to billions like what we did for the 2023.

It takes really long to render (it depends on the memory you have in your laptops) so, I will place the best strategy, that is creating static maps and combining them to form a video. Notice that when cleaning the environment, I chose to keep `rgdp_map` since we will still use this.

```
rm(list = setdiff(ls(), "rgdp_map"))
gc()
```

```
##            used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells  5671429 302.9    9801260 523.5  9801260 523.5
## Vcells 37653826 287.3   88064770 671.9 94237569 719.0
```

```
#for mp4
if (!require(av)) install.packages("av")
library(av)
```

We need to create a directory *within* our working directory where the images will be saved.

```
dir.create("ch_gdp_frames", showWarnings = FALSE)
```

We now generate static maps; this requires us to create a function/loop. You will notice that the `plot.background` was set twice. It is because when we use the `av`, the default is a black setting so it's better to have a white background set for the plot and the entire canvas.

```
for(year in 2000:2023){
  #filter data for the specific year
  data_year<-rgdp_map %>% filter(Year==year)

  #Generate the plot
  ip<-ggplot(data_year) +
  geom_sf(aes(fill = GDP), color = "white") +
  scale_fill_viridis_c(
    option = "magma",
    name = "GDP",
    labels = scales::label_number(accuracy = 0.1, suffix = "B", big.mark = ",")
  ) +
  labs(
    title = paste("Regional GDP of the Philippines"),
    subtitle = paste("Year:", year),
    fill = "GDP (Billion $)"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 16, face = "bold"),
```

```
    plot.subtitle = element_text(size = 14),
    panel.background = element_rect(fill = "white", color = NA),  # Set white background
    plot.background = element_rect(fill = "white", color = NA)   # Set white background
  )


  # Save the plot as an image
  ggsave(filename = paste0("ch_gdp_frames/gdp_", year, ".png"), plot = ip, width = 8, hei
}
```

You need to edit some parts, such as the `2000:2023` with whatever years you have; you need to filter data depending on how you called the time column. Then edit the `fill` and other things.

We now combine our static maps into MP4. If you want the transition for each year to be seen clearly, the framerate is set at 1 to have 1 frame per second and the frame is repeated twice. the `vfilter` makes the video HD

```
file_list <- list.files("ch_gdp_frames", pattern = "gdp_.*\\.png", full.names = TRUE)
rep_files <- rep(file_list, each = 2)  # Repeat each frame twice
av::av_encode_video(
  input = rep_files,
  output = "ch_rgdp_time.mp4",
  framerate = 1,
  vfilter = "scale=1280:720"
)
```

10.9 Practical: Advanced Visualizations

For the practical, you need to search and clean the dataset on your own. Please search in PSA OpenStat: *Number of Registered Live Births by Sex and by Usual Residence of Mother (Region, Province and Highly Urbanized City), Philippines: January - December 2013-2022.* Only gather regional data since the shapefile that is given for this practical is for Regions only. Please save each animation using `anim_save`

1. Using PSA OpenStat (2013-2022) data, create an animated bar chart showing the total number of live births per year. Differentiate Male vs. Female births using `fill`
2. Create an animated bar chart that compares the number of live births per region over time (2013-2022). Use `transition_states(year, wrap = FALSE)` to animate regional birth counts changing over time. Determine the region with the highest live births in each frame.
3. Select 3-5 regions and visualize their live birth trends (2013-2022) using an animated time series plot.
4. Using PSA OpenStat (2013-2022) data and a Philippines regional shapefile, create a static choropleth map of the recent number of live births per region. Color the regions based on `birth count`
5. Using PSA OpenStat (2013-2022) data and a Philippines regional shapefile, create an animated choropleth map showing how the number of live births per region changes over time. Color the regions based on `birth count`
6. How do birth trends differ across Philippine regions?
7. Which regions have experienced the highest increase or decrease in live

births from 2013 to 2022?

8. How does using `gganimate` with `sf` enhance your ability to analyze spatial(and time) data trends?

Chapter 11

Descriptive Statistics

For this section, we will make use of `Ch6PRactice.RData`. Since the file is in RData format, we will only need to load it in R.

11.1 Preliminaries

11.1.1 Install Packages

We install and load the "psych" package then load the tidyverse package

```
if(!("psych" %in% installed.packages()[,"Package"])) install.packages("psych")
if(!("gtsummary" %in% installed.packages()[,"Package"])) install.packages("gtsummary")
library(psych)
library(tidyverse)
library(gtsummary)
```

11.1.2 Load the dataset

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

```
##            used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells  5660104 302.3    9801260 523.5  9801260 523.5
## Vcells 16556980 126.4   70451816 537.6 94237569 719.0
```

```
load("Ch6Practice.RData")
```

We subset the data:

```
# Create a subset of the dataset with only female respondents
Dataset_CP_f <- Dataset_CP %>% filter(HH2a_Sex == "Female")
# Create a subset of the dataset with only male respondents
Dataset_CP_m <- Dataset_CP %>% filter(HH2a_Sex == "Male")
```

11.2 Frequency Table

Run a Frequency Table for the number of hours the respondents spend caring for children and/or grandchildren per week (variable **Q37a_HoursWeekChildren**). For both sex separately, report the mean and standard deviation.

11.2.1 Male

Calculate the descriptive statistics for the variable **Q37a_HoursWeekChildren**, using the function *describe* of the package *psych*:

```
psych::describe(Dataset_CP_m$Q37a_HoursWeekChildren)
```

```
##    vars   n  mean    sd median trimmed   mad min max range skew kurtosis   se
## X1    1 575 16.86 15.53     10   14.36 10.38   1 100    99 1.83     4.65 0.65
```

Notice how we directly use the *describe* function on the variable **Q37a_HoursWeekChildren** (called with the dollar sign operator, *$*, which works as follows: *name_dataset$name_variable* - notice that to access a specific variable, some functions can use the pipe operator while others require the dollar sign operator, as in this case). Notice also how we call the function *describe*: we use the :: operator, which tells R to look for a function within a specific package as follows

name_package::name_function

You could also simply call the function by only using its name, R will then search for that specific function across all the loaded R packages. However, if more packages have functions with the same name, specifying the name of the package with the :: operator helps assuring the usage of the correct function.

Report the Frequency Table for the variable **Q37a_HoursWeekChildren**, using the function `tbl_summary` of the package `gtsummary`:

Hours per Week with Children	**N = 1,723**[1]
Q37a_HoursWeekChildren	10 (5, 24)

[1]Median (Q1, Q3)

```
Dataset_CP_m %>%
  select(Q37a_HoursWeekChildren) %>%
  tbl_summary(
    statistic = list(all_categorical() ~ "{n} ({p}%)"),  # Show count & percentag
    missing = "no"  # Exclude missing values (set to "ifany" to include)
  ) %>%
  modify_header(label = "**Hours per Week with Children**")  # Customize header
```

The table shows that there are 1, 723 non-missing observations for the variable, `Q37a_HoursWeekChildren`

The median value of hours per week with children is 10 hours. (5,24) represents the interquartile range. Q1 (5): The first quartile (25th percentile) and the third quartile (75th percentile) are 5 and 24 hours respectively. This simply means that 50% of the responses lie between 5 and 24 hours per week.

When interpreting:

"The median hours per week spent with children of males is 10 hours, with half of the respondents reporting values between 5 and 24 hours. The total sample size is 1,723."

Moreover, in R there are other ways to access specific elements within a dataset. Datasets are organized in rows (which represent our observations)

and columns (which represent our variables). We can select a specific cell of a dataset as follows: *name_dataset[row, column]*, where *row* is the number(s) of row(s) we are interested in and *column* is either the number or the name of the column we are interested in. For instance, if we want to select the value of the variable **Q37a_HoursWeekChildren** for the third observation, we can use the following command

Dataset_CP[3,"Q37a_HoursWeekChildren"]

If we do not specify a value for the rows (or columns), we select all the rows (or columns) of the specified columns (or rows). For instance, the following command selects all the observations for the variable **Q37a_HoursWeekChildren**

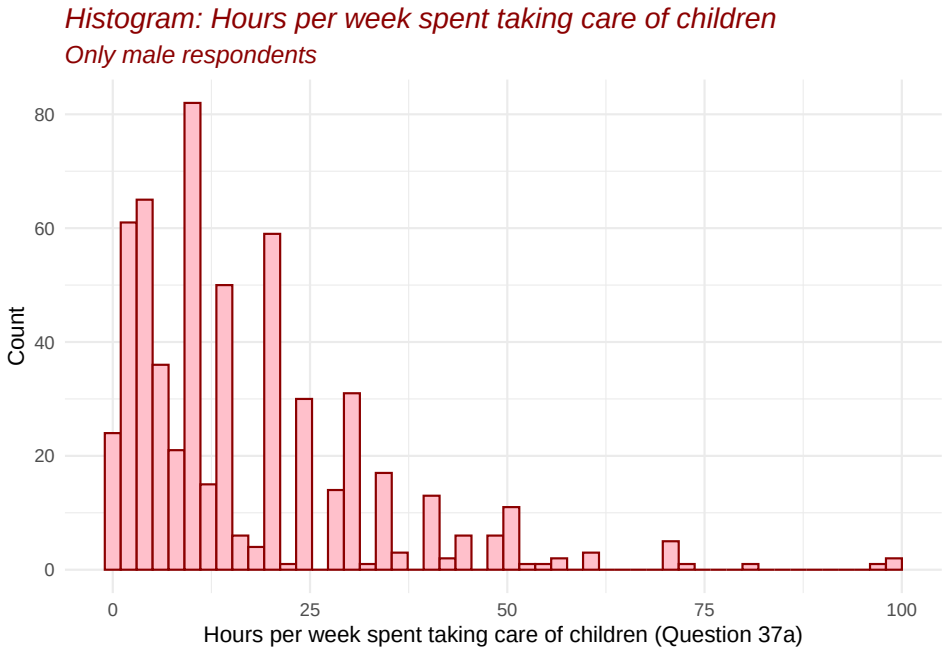
Dataset_CP[,"Q37a_HoursWeekChildren"]

11.3 Histogram

Plot the histogram for the variable **Q37a_HoursWeekChildren**

```
ggplot(data = Dataset_CP_m, aes(x = Q37a_HoursWeekChildren)) +
  geom_histogram(bins = 50,
                 color = "darkred",
                 fill = "pink") +
  theme_minimal()+
  labs(title = "Histogram: Hours per week spent taking care of children", subtitle = "Onl
  xlab("Hours per week spent taking care of children (Question 37a)") +
  ylab("Count") +
  theme(plot.title = element_text(color = "darkred",
```

```
                                    size = 14,
                                    face = "italic"),
      plot.subtitle = element_text(color = "darkred", size = 12, face = "italic
```



- Depending on the data, different number of bins are more or less appropriate. In this case, we indicate that we want our data to be split in 50 intervals. You could indicate a different number or you could also leave the *bins* option out of the code and let R select it (R will give a warning (not error) about the number of bins which has been selected). Each type of graph has its own set of options that can be set, use the help function to learn more about them.

- Next, we call a pre-defined layout of the graph called *theme_minimal*. There are many pre-defined layouts and you are free to choose

whichever you like the most, or even create your own layout (which is partially what we do with the last layer).

- In the fourth layer we use *labs* in order to define the title of the graph, we add a subtitle and we specify the name of the axes. There are more options that you can define within *labs*, use *help(labs)* to find out what these are.

- Finally, we edit the appearance of the title and subtitle using *theme* and introducing the elements *plot.title = element_text()* and *plot.subtitle = element_text()*, where we specify their *color*, *size* and *face*. With that, we are able to change the format of the title and subtitle, however, it is not always necessary. If you want to find out more options that you can specify within theme, you can use *help("theme")* or search in Google.

11.4 Summary Table

To summarize, we can build a table showing the mean and the standard deviation for the variable **Q37a_HoursWeekChildren** for the two groups, separately. To do so, we use two features:

First, in the menu bar of your Quarto document (on top), we click on "Table" and then select "Insert Table". A new window will appear and ask us the number of rows and columns our table should include. Once the table has been created, we can fill in its cells with whatever we wish to present in the table.

Second, we use inline code, which allows us to embed directly into the text

the output of an R code (you can find more information about inline coding here).

A chunk of code, as we have seen so far, is separated from the text of our Quarto document and does not allow us to build a readable text combining words with output from R codes which in one linear sentence. So, if we want to say:

The number of male respondents in our dataset is equal to

```
Dataset_CP_m %>% summarize(n = n())
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1  1723
```

We can clearly see that this is not ideal when using a chunk of code.

To better integrate this into our text we can use inline code. To do so, in the menu bar of our Quarto document (on top), we click on "</>", then we type r and add a space and then write the line of code we would like to execute in order to include its output into the text. In our previous example, this could be done by writing as an inline code "r Dataset_CP_m %>% summarize(n = n())", as follows:

The number of male respondents in our dataset is equal to 1723

This could be useful, for instance, if we wish to present some properties of

the data without us manually copy-pasting the numbers from the console to the text. At the same time, we can use this to populate our table, as we indeed do below.

Note:

In a Quarto document, you can run the code which is included in a chunk of code with the green play button. This allows you to execute and visualize the output of that specific command. Unfortunately, inline codes do not have the same option. Inline codes will be executed only when the Quarto document is rendered. This means that you cannot directly see the result of inline codes before you render your Quarto document nor you can verify if that code is correct (and if it is not, the rendering will not work).

To overcome this issue, you can copy-paste the code of your inline code into the Console and run it from there. This will allow you to verify if the code is correct and what is the output.

We have created here a table summarizing some of the descriptive statistics for the male respondents in our dataset. Take a look at this table:

	Male	Code
N_total	1723	Dataset_CP_m %>% summarize (n=n())
N_missing	1148	Dataset_CP_m%>% summarize(n_na=sum(is.na(Q37a_HoursWeekChildren)))
N_valid	575	Dataset_CP_m%>%summarize(n_valid=sum(!is.na(Q

	Male	Code
Mean	16.8591304347826	Dataset_CP_m%>%summarize(mean = mean(Q37a_HoursWeekChildren, na.rm=T))
Median	10	Dataset_CP_m%>%summarize(median = median(Q37a_HoursWeekChildren, na.rm=T))
Std. Deviation	15.5294252315733	Dataset_CP_m%>% summarize(sd=sd(Q37a_HoursWeekChildren, na.rm=T))

Notice how we use the pipe operator on the dataset to compute the descriptive statistics with the *summarize* function and

- the *n* function to calculate the total number of observations;
- the *is.na* function to calculate the number of observations with missing values;
- the *!* operator combined with the *is.na* function to calculate the opposite of the number of observations with missing values, i.e., the number of observations with valid values;
- the *mean* function to calculate the mean, with the *na.rm* option set to TRUE to tell R that there are missing values and it should compute

the mean without considering the missing values;

- the *median* function to calculate the median, with the *na.rm* option set to TRUE to tell R that there are missing values and it should compute the median without considering the missing values.

- the *sd* function to calculate the standard deviation, with the *na.rm* option set to TRUE to tell R that there are missing values and it should compute the standard deviation without considering the missing values.

11.5 Interpretation of Descriptive Statistics:

Descriptive statistics serve as tools to summarize and organize data into meaningful insights, making it easier to understand and communicate findings. The following are the key aspects of descriptive statistics:

11.5.0.1 Definition

Descriptive statistics involve summarizing, organizing, and simplifying data in a way that is understandable and interpretable. This can include numerical summaries, tables, and graphical representations to describe the distribution, central tendency, and variability of data.

11.5.1 Key Components of Descriptive Statistics

1. **Measures of Central Tendency**:

- **Mean**: The average value, useful for symmetric distributions without outliers.
- **Median**: The middle value in the data, efficient even with skewness and outliers.
- **Mode**: The most frequent value, helpful for categorical or multimodal distributions.

2. **Measures of Variability**:

- **Range**: Difference between the maximum and minimum values.
- **Standard Deviation (SD)**: Indicates the average spread of data around the mean.
- **Interquartile Range (IQR)**: Captures the spread of the middle 50% of the data.

3. **Data Distribution**:

- Histograms and frequency distributions help identify the shape (e.g., symmetric, skewed, bimodal) and any potential outliers in the dataset.

4. **Data Representation**:

- Graphical methods like bar charts, line graphs, pie charts, and boxplots make patterns and trends in the data more visible and easier to interpret.

11.5.2 When to Use Mean or Median

- The mean is preferred when the data is symmetrically distributed without extreme values.
- The median is better for skewed distributions or datasets with significant outliers, as it is not affected by extreme values.
- Example: In income data, where a few individuals earn significantly more than the majority, the median provides a more realistic "typical" income.

11.5.3 Purpose of Descriptive Statistics

- To summarize large datasets into manageable insights.
- To provide a foundation for inferential statistics, which goes beyond describing the sample to drawing conclusions about the population.
- To describe relationships between variables, if applicable, through measures like correlation coefficients.

11.5.4 Limitations

- Descriptive statistics only describe the data at hand; they do not infer or predict trends in the population.
- Without considering measures of variability (e.g., SD or IQR), central tendency measures (mean, median) can be misleading.
- Descriptive statistics are limited in making generalizations beyond the sample data.

11.6 Practical: Descriptive Statistics

You will still use the `Dataset_CP` data for the practical.

11.6.1 Descriptive Statistics Questions

- **Q37b_HoursWeekHousework**:

    1. What are the mean, median, and standard deviation for the time spent on housework (`Q37b_HoursWeekHousework`) for the entire dataset?
    2. Compare the mean and median for males and females. Is the mean or median a better representation of central tendency? Why?
    3. What does the standard deviation tell you about variability in housework hours?

- **Q7_HoursWeekWork**:

    1. What is the range of hours worked per week (`Q7_HoursWeekWork`) for males and females?
    2. Are there any noticeable outliers in the data? How could these affect the mean?

- **Q37c_HoursWeekElderly**:

    1. Plot a histogram for hours spent caring for the elderly (`Q37c_HoursWeekElderly`).

2. Is the distribution symmetric, skewed, or bimodal? What might explain the shape of the distribution?

- In the dataset, males and females spend different amounts of time on housework (`Q37b_HoursWeekHousework`). What societal or cultural factors might explain this difference? How could this information be used to inform policy?
- How might understanding variability in working hours (`Q7_HoursWeekWork`) help employers or policymakers design better work-life balance policies?
- After calculating descriptive statistics and creating visualizations, what challenges did you face in interpreting the data? How did the choice of tools (e.g., `psych`, `gtsummary`) affect your approach?

Chapter 12

Hypothesis Testing

12.1 Steps:

1. State null and alternative hypotheses
2. Choose a test (one-tailed or two-tailed)
3. Do the test
4. Present your results
5. Conclusion → reject the null or fail to reject
6. Make a conclusion about your data (From our data, we have evidence that…)

12.2 Difference between one-tailed test and two-tailed test

A **two-tailed test** checks whether a sample mean or proportion is significantly different from a given value in either direction (higher or lower).

- Null Hypothesis (H_0): There is no difference between the sample statistic and the population value.
- Alternative Hypothesis (H_α): The sample statistic is either higher or lower than the population value.

A **one-tailed** test checks if a sample statistic is significantly greater or smaller than a given value in one direction only.

- Null Hypothesis (H_0): There is no difference OR the difference is in the opposite direction.
- Alternative Hypothesis (H_α): The sample statistic is either higher or lower, but not both.

12.2.1 Deciding between One-Tailed versus Two-Tailed

1. A central bank claims that the country's average annual inflation rate is 3%. An economist wants to test whether this is incorrect.
2. A government introduces a new minimum wage policy, and policymakers want to check whether it has decreased employment levels.
3. A researcher wants to compare the GDP growth rate of two neighboring countries to see if they are different.

4. An economist wants to determine whether male workers earn more than female workers in the same industry.
5. A researcher wants to test whether developing countries receive less Foreign Direct Investment (FDI) than developed countries.

12.3 Test of Proportions

A test of proportion is used when we want to check whether the proportion of a certain event (e.g., unemployment rate, poverty rate) in a sample is different from a known proportion in the population.

12.3.1 One-Sample Test of Proportion

We use this test when we want to see if the proportion of a sample is different from the population proportion.

Example: Unemployment Rate

A government report claims that 8% of the labor force is unemployed. We collect a sample of 500 workers and find that 50 are unemployed. Is the actual unemployment rate significantly different from 8%?

```
prop.test(x = 50, n = 500, p = 0.08, correct = FALSE)
```

```
##
##  1-sample proportions test without continuity correction
##
## data:  50 out of 500, null probability 0.08
## X-squared = 2.7174, df = 1, p-value = 0.09926
```

```
## alternative hypothesis: true p is not equal to 0.08
## 95 percent confidence interval:
##  0.07667756 0.12942191
## sample estimates:
##   p
## 0.1
```

This employs `prop.test` then inside the parenthesis, we have x=Number of successes, n=total sample size and p=hypothesized population proportion. The `correct=FALSE` adjusts the t-statistic. The rule is to set it to TRUE for small sample sizes and FALSE for big sample sizes.

- If the p-value < 0.05, we reject the null hypothesis, meaning the unemployment rate in our sample is significantly different from 8%.
- If the p-value > 0.05, we fail to reject the null hypothesis, meaning we do not have enough evidence to say the unemployment rate differs from 8%.

If we believe the true unemployment rate is higher than 8%

```
prop.test(x = 50, n = 500, p = 0.08, alternative = "greater", correct = FALSE)
```

```
##
##  1-sample proportions test without continuity correction
##
## data:  50 out of 500, null probability 0.08
## X-squared = 2.7174, df = 1, p-value = 0.04963
```

```
## alternative hypothesis: true p is greater than 0.08
## 95 percent confidence interval:
##  0.08003919 1.00000000
## sample estimates:
##   p
## 0.1
```

Since it is one-tailed, `alternative = "greater"` tests the direction. The alternative is changed if it is in the opposite direction.

If we believe the true unemployment rate is lower than 8%

```
prop.test(x = 50, n = 500, p = 0.08, alternative = "less", correct = FALSE)
```

```
##
##  1-sample proportions test without continuity correction
##
## data:  50 out of 500, null probability 0.08
## X-squared = 2.7174, df = 1, p-value = 0.9504
## alternative hypothesis: true p is less than 0.08
## 95 percent confidence interval:
##  0.0000000 0.1242664
## sample estimates:
##   p
## 0.1
```

12.3.2 Two-Sample Test of Proportion

We use this when comparing proportions between two groups (e.g., unemployment rates in two different regions).

12.3.2.1 Example: Employment Rate Comparison

A researcher wants to compare the employment rates between urban and rural areas.

- In an urban area, 450 out of 500 people are employed.
- In a rural area, 420 out of 500 people are employed.

We test whether the employment rates are significantly different.

```
prop.test(x = c(450, 420), n = c(500, 500), correct = FALSE)
```

```
##
##  2-sample test for equality of proportions without continuity correction
##
## data:  c(450, 420) out of c(500, 500)
## X-squared = 7.9576, df = 1, p-value = 0.004789
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  0.01847836 0.10152164
## sample estimates:
## prop 1 prop 2
##   0.90   0.84
```

The x= number of successes in the two groups, n=total number of sample sizes in the two groups.

Interpreting the Results:

- If p-value < 0.05: Employment rates differ significantly between urban and rural areas.
- If p-value > 0.05: There is no significant difference in employment rates.

If we believe urban employment rate is higher than rural:

```
prop.test(x = c(450, 420), n = c(500, 500), alternative = "greater", correct = FALSE)
```

```
##
##  2-sample test for equality of proportions without continuity correction
##
## data:  c(450, 420) out of c(500, 500)
## X-squared = 7.9576, df = 1, p-value = 0.002394
## alternative hypothesis: greater
## 95 percent confidence interval:
##  0.02515394 1.00000000
## sample estimates:
## prop 1 prop 2
##   0.90   0.84
```

If we believe urban employment rate is lower than rural:

```
prop.test(x = c(450, 420), n = c(500, 500), alternative = "less", correct = FALSE
```

```
##
##  2-sample test for equality of proportions without continuity correction
##
## data:  c(450, 420) out of c(500, 500)
## X-squared = 7.9576, df = 1, p-value = 0.9976
## alternative hypothesis: less
## 95 percent confidence interval:
##  -1.00000000  0.09484606
## sample estimates:
## prop 1 prop 2
##   0.90   0.84
```

12.3.3 Real-World Example:

We are going to use the `wage1` dataset from Wooldridge package but we do not want to do anything to the original dataset (though technically we can) but, better safe than sorry.

```
library(wooldridge)
data(wage1)
```

```
sample<-wage1
```

Is the proportion of female workers greater than 50%?

```
prop.test(sum(sample$female), nrow(sample), p = 0.5, alternative = "greater", correct = F
```

```
##
##  1-sample proportions test without continuity correction
##
## data:  sum(sample$female) out of nrow(sample), null probability 0.5
## X-squared = 0.92015, df = 1, p-value = 0.8313
## alternative hypothesis: true p is greater than 0.5
## 95 percent confidence interval:
##  0.443458 1.000000
## sample estimates:
##         p
## 0.4790875
```

12.4 T-Tests

A t-test is used to compare means (e.g., wages, GDP growth).

Types of T-Tests:

1. One-Sample T-Test: Compares a sample mean to a known value.
2. Independent-Samples T-Test: Compares means of two different groups.
3. Paired-Samples T-Test: Compares before and after effects within the same group.

12.4.1 One-Sample T-Test

A report states that the average monthly wage in a country is 20,500. We take a sample of 100 workers and test if their wages differ.

```
set.seed(123)
wages<-rnorm(100, mean=21000, sd=500)
t.test(wages, mu=20500)
```

```
##
##  One Sample t-test
##
## data:  wages
## t = 11.946, df = 99, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 20500
## 95 percent confidence interval:
##  20954.64 21135.76
## sample estimates:
## mean of x
##   21045.2
```

The `rnorm` generates a random sample of 100 wages. Then the `t.test` performs a one-sample t-test by comparing if the sample mean (21000) is different from the hypothesized mean (20500)

Interpreting the results:

- If p-value < 0.05: The average wage is significantly different from 20,500.

- If p-value > 0.05: There is no significant difference in wages.

12.4.2 Real-World Example

Is the average hourly wage significantly different from $5 per hour?

```
t.test(sample$wage, mu = 5)
```

```
##
##  One Sample t-test
##
## data:  sample$wage
## t = 5.5649, df = 525, p-value = 4.186e-08
## alternative hypothesis: true mean is not equal to 5
## 95 percent confidence interval:
##  5.579768 6.212437
## sample estimates:
## mean of x
##  5.896103
```

12.4.3 Independent-Samples T-Test

We compare male and female wages in the same industry.

```
set.seed(123)
male_wage<-rnorm(50, mean=23000, sd=600)
female_wage<-rnorm(50, mean=22500, sd=500)
```

```r
t.test(male_wage, female_wage, var.equal=TRUE)
```

```
##
##  Two Sample t-test
##
## data:  male_wage and female_wage
## t = 4.4149, df = 98, p-value = 2.603e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  246.3177 648.5583
## sample estimates:
## mean of x mean of y
##  23020.64  22573.20
```

This performs a two-sample t-test assuming equal variances.

Interpreting the results:

- If p-value < 0.05, wages differ significantly.
- If p-value > 0.05, there is no significant difference.

12.4.3.1 Checking variance before running the T-Test

Variance measures how spread out the data is around the mean.

- Low variance $\rightarrow$ Data points are close to the mean.
- High variance $\rightarrow$ Data points are spread out from the mean.

```
var.test(male_wage, female_wage)
```

```
##
##  F test to compare two variances
##
## data:  male_wage and female_wage
## F = 1.5057, num df = 49, denom df = 49, p-value = 0.1556
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.8544445 2.6533137
## sample estimates:
## ratio of variances
##           1.505692
```

If variance ratio is close to 1, assume equal variances (`var.equal = TRUE`).

12.4.4 Real-World Example

Do men and women earn different wages?

```
#check variance
var.test(sample$wage[sample$female == 1], sample$wage[wage1$female == 0])
```

```
##
##  F test to compare two variances
##
## data:  sample$wage[sample$female == 1] and sample$wage[wage1$female == 0]
```

```
## F = 0.36954, num df = 251, denom df = 273, p-value = 4.813e-15
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.2900223 0.4714405
## sample estimates:
## ratio of variances
##          0.3695359
```

Interpreting the variance ratio:

- Null Hypothesis (H_0): The variances of male and female wages are equal
- Alternative Hypothesis (H_α): The variances of male and female wages are not equal

```
t.test(sample$wage ~ sample$female, var.equal = FALSE)
```

```
##
##  Welch Two Sample t-test
##
## data:  sample$wage by sample$female
## t = 8.44, df = 456.33, p-value = 4.243e-16
## alternative hypothesis: true difference in means between group 0 and group 1 i
## 95 percent confidence interval:
##  1.926971 3.096690
## sample estimates:
## mean in group 0 mean in group 1
```

```
##        7.099489        4.587659
```

~ Means "wage depends on gender" (separates the dependent and independent variable)

12.4.5 Paired-Samples T-Test

A new minimum wage policy was introduced. We analyze its impact on worker wages before and after implementation.

```r
set.seed(0)
wage_before <- rnorm(50, mean = 32000, sd = 400)
wage_after <- wage_before + rnorm(50, mean = 2500, sd = 100)

t.test(wage_before, wage_after, paired = TRUE)
```

```
##
##  Paired t-test
##
## data:  wage_before and wage_after
## t = -206.45, df = 49, p-value < 2.2e-16
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -2526.496 -2477.785
## sample estimates:
## mean difference
##       -2502.141
```

This means that wage_after was simply the wages after the policy were increased by approx. 2500 per worker. The `t.test` simply performs a paired t-test that is why `paired=TRUE`

Interpreting the Results:

- If p-value < 0.05: The minimum wage policy significantly increased wages.
- If p-value > 0.05: The policy had no significant effect.

Did wages increase?

```
t.test(wage_before, wage_after, paired = TRUE, alternative = "greater")
```

```
##
##  Paired t-test
##
## data:  wage_before and wage_after
## t = -206.45, df = 49, p-value = 1
## alternative hypothesis: true mean difference is greater than 0
## 95 percent confidence interval:
##  -2522.46      Inf
## sample estimates:
## mean difference
##       -2502.141
```

Chapter 13

Ordinary Least Squares

13.1 General Description and notation of a Linear Regression

In R, the `set.seed()` function is used to set the random number generator's seed. Random number generators in computer programming are not truly random; they are pseudo-random, meaning they generate sequences of numbers that appear random but are actually determined by an initial starting point called a seed. If you set the seed to a specific value, you can reproduce the same sequence of random numbers every time you run your program. This is important in many data analysis and simulation tasks, as it allows for the results to be reproducible.

Note: we will use the `cat()` function today. The `cat()` function is used for printing or concatenating objects to the console. It is primarily used to display text, numbers, or other R objects on the screen or to write them to a file. The name "cat" stands for "concatenate" or "concatenation," as the

function can be used to combine and display multiple objects.

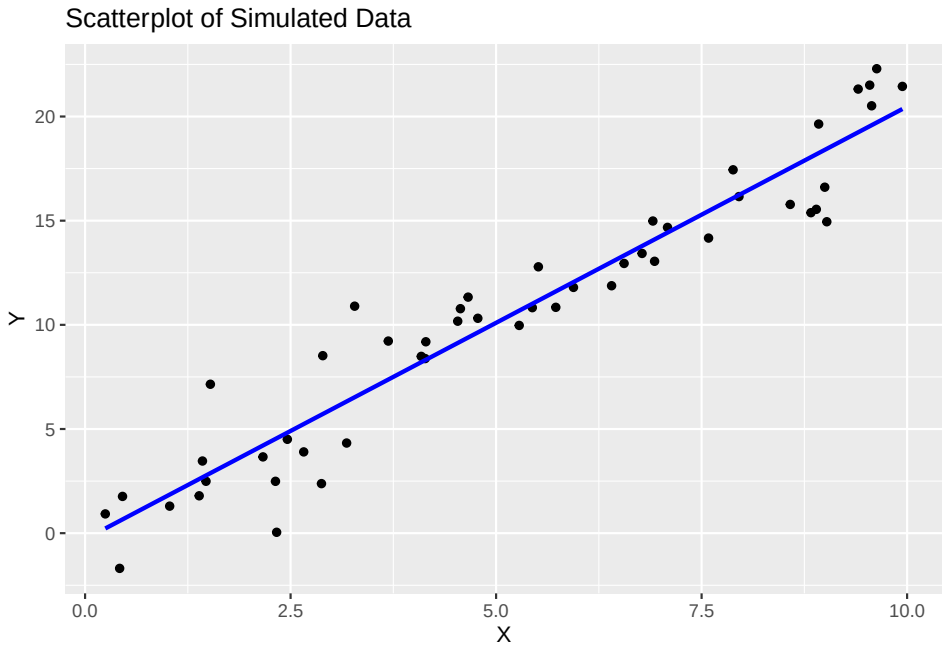
```
library(tidyverse)
```

```
rm(list = setdiff(ls(), "sample"))
gc()
```

```
##            used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells  5658236 302.2    9801260 523.5  9801260 523.5
## Vcells 11623454  88.7   56361453 430.1 94237569 719.0
```

```
# Generate example data
set.seed(123)
n <- 50
x <- runif(n, 0, 10)
y <- 2 * x + rnorm(n, 0, 2)

# Create a scatterplot
ggplot(data = data.frame(x, y), aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  labs(title = "Scatterplot of Simulated Data",
       x = "X", y = "Y")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



13.1.1 The notation

```
cat("Linear Regression Notation:\n")
```

```
## Linear Regression Notation:
```

```
cat("Y =   +  *X + \n")
```

```
## Y =   +  *X +
```

```
cat("Y: Dependent variable (response)\n")
```

```
## Y: Dependent variable (response)
```

```
cat("X: Independent variable (predictor)\n")
```

```
## X: Independent variable (predictor)
```

```
cat(" : Intercept (constant)\n")
```

```
##  : Intercept (constant)
```

```
cat(" : Coefficient for X (slope)\n")
```

```
##  : Coefficient for X (slope)
```

```
cat(" : Error (residuals)\n")
```

```
##  : Error (residuals)
```

13.2 Running and reporting OLS Regression

In R, the **`lm()`** function is used to fit linear regression models. Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables by finding the best-fitting linear equation. The **`lm()`** function is a fundamental tool for performing linear regression analysis in R.

Here's how to use the **`lm()`** function in R:

model <- lm(dependent_variable ~ independent_variable1 + independent_variable2, data = your_data_frame)

Let's break down the components of this function:

1. `dependent_variable`: This is the variable you want to predict or model. It is the response variable, typically numeric.

2. `independent_variable1`, `independent_variable2`, ...: These are the variables that you believe are related to the dependent variable. They are the predictor variables, which can be numeric or categorical.

3. `data`: This argument specifies the data frame in which the variables are located. You should specify the name of your data frame here.

The `lm()` function returns a linear regression model object, which contains information about the estimated coefficients and other statistical details of the model.

After fitting the model, you can access various information and summaries, such as:

- `summary(model)`: This function provides a detailed summary of the regression model, including coefficients, standard errors, t-values, p-values, R-squared, and more.

- `coef(model)`: This function returns the estimated coefficients of the model.

- `predict(model, newdata)`: You can use this function to make predictions with the model, where `newdata` is a data frame containing the values of the independent variables for which you want to make predictions.

```r
# Fit an OLS regression model
model <- lm(y ~ x)

# Display the model summary
summary(model)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5116 -1.1157 -0.1313  1.0985  4.3723
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.2847     0.5442  -0.523    0.603
## x             2.0764     0.0913  22.743   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.881 on 48 degrees of freedom
## Multiple R-squared:  0.9151, Adjusted R-squared:  0.9133
## F-statistic: 517.2 on 1 and 48 DF,  p-value: < 2.2e-16
```

13.3 Interpretation of Coefficients

For numeric independent variables, a positive coefficient means that an increase in the independent variable is associated with an increase in the dependent variable, and a negative coefficient means the opposite.

For categorical independent variables (e.g., dummies for categories), the coefficients represent the difference in the dependent variable for that category compared to the reference category.

```
# Extract and interpret coefficients
cat("Intercept ( ): ", coef(model)[1], "\n")
```

```
## Intercept ( ):  -0.2847054
```

```
cat("Coefficient for X ( ): ", coef(model)[2], "\n")
```

```
## Coefficient for X ( ):  2.076349
```

13.4 Goodness of Fit (F-Statistics and Adjusted R-squared)

The null hypothesis (H0) for the F-test is that all coefficients in the model are equal to zero, meaning that none of the independent variables (wt and cyl) have any effect on the dependent variable. The small p-value (<0.05) means we can reject the null hypothesis.

Note: to extract p-value of F-statistic you have to use the function pf() that gives the area, for the F distribution, to the left of the value x for two given degrees of freedom, df1 and df2.

```
# Adjusted R-squared
cat("Adjusted R-squared: ", summary(model)$adj.r.squared, "\n")
```

```
## Adjusted R-squared:  0.9133112
```

```
# Print the F-statistic and p-value
cat("F-statistic:", summary(model)$fstatistic[1], "\n")
```

```
## F-statistic: 517.2401
```

```
p_val = pf(summary(model)$fstatistic[1],summary(model)$fstatistic[2],summary(mode
cat("p-value:", p_val, "\n")
```

```
## p-value: 2.364413e-27
```

13.5 Assumption Diagnostics

In R, when you create a plot of a linear regression model (an OLS model) using the **`plot()`** function, it produces a set of diagnostic plots to help you assess the assumptions and goodness of fit of the model. These diagnostic plots are typically used to check if the model is a good fit for the data and whether the assumptions underlying linear regression are met. Here's an explanation of each output produced by **`plot(model)`**:

- **Residuals vs. Fitted Values (Partial Regression Plot)**:
  - This plot shows the relationship between the residuals (the differences between the observed values and the predicted values) and the fitted values (the predicted values from the model).
  - It is used to check for linearity, which is one of the key assumptions of linear regression. Ideally, you want to see a random scattering of points around a horizontal line, with no clear pattern.
  - If you see a pattern, such as a curve or funnel shape, it may indicate a violation of the linearity assumption.
- **Normal Q-Q (Quantile-Quantile) Plot**:
  - This plot compares the distribution of the residuals to a theoretical normal distribution. If the residuals are normally distributed, the points should roughly follow a straight line.
  - Deviations from a straight line may indicate departures from normality in the residuals. For example, if the points bend upwards or downwards, it may indicate skewness or heavy-tailed residuals.
- **Scale-Location (Spread-Location) Plot**:
  - This plot shows the square root of the absolute residuals against the fitted values. It is also known as a "spread vs. location" plot.
  - It is used to check for homoscedasticity, which means that the variability of the residuals is roughly constant across different levels of the fitted values.

  - Ideally, you want to see a random scattering of points around a horizontal line with roughly equal spread.
  - If the spread changes systematically with the fitted values (e.g., a funnel shape), it may indicate heteroscedasticity, which can lead to unreliable parameter estimates and hypothesis tests.

- **Residuals vs. Leverage Plot (Cook's Distance Plot)**:

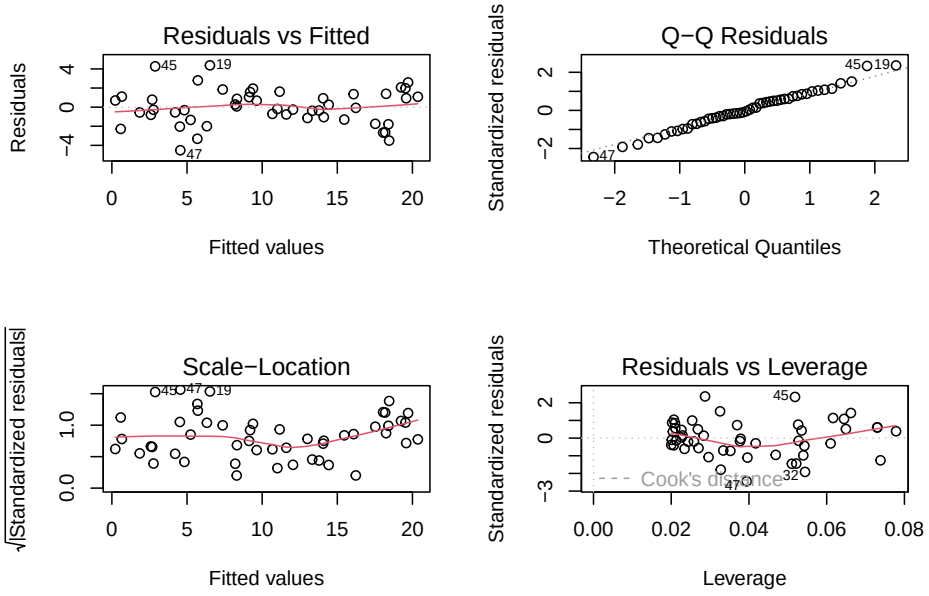
  - This plot displays the standardized residuals against the leverage values (a measure of how much an observation affects the regression coefficients). It is often used to identify influential data points.
  - Data points that are far from the other points in this plot have high leverage, and points above a certain threshold may be influential.
  - Influential points can have a significant impact on the model's parameters, and it's important to examine them for potential outliers or errors.

- **Cook's Distance Plot**:

  - Cook's distance is a measure of the impact of each observation on the model's coefficients. This plot shows the Cook's distance for each observation.
  - Observations with a high Cook's distance are potential outliers or influential points that can significantly affect the model. It is a useful tool for identifying data points that should be investigated further.

- **Multicollinearity**
  - **Correlation Matrix**: A simple way to start is by looking at the correlation matrix of the predictors. High correlation coefficients (close to -1 or 1) between pairs of predictors indicate potential multicollinearity. This can be done using the `cor()` function.

```
# Check OLS assumptions
par(mfrow = c(2, 2))
plot(model)
```



13.6 Real-World Example

We are going to use the `sample` dataset

```
sample_sub<-sample %>% select(c(wage,educ,exper,tenure))
head(sample_sub)
```

```
##   wage educ exper tenure
## 1 3.10   11     2      0
## 2 3.24   12    22      2
## 3 3.00   11     2      0
## 4 6.00    8    44     28
## 5 5.30   12     7      2
## 6 8.75   16     9      8
```

Create a scatterplot

```
# Create a scatterplot
ggplot(data = sample_sub, aes(x = sample_sub$educ, y =sample_sub$wage)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  labs(title = "Scatterplot of Wage Data",
       x = "Education", y = "Wage")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```


```
# Fit an OLS regression model
model.1 <- lm(wage ~ educ, data=sample_sub)


# Display the model summary
summary(model.1)
```

```
##
## Call:
## lm(formula = wage ~ educ, data = sample_sub)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.3396 -2.1501 -0.9674  1.1921 16.6085
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.90485    0.68497  -1.321    0.187
## educ         0.54136    0.05325  10.167   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.378 on 524 degrees of freedom
## Multiple R-squared:  0.1648, Adjusted R-squared:  0.1632
## F-statistic: 103.4 on 1 and 524 DF,  p-value: < 2.2e-16
```

13.6.0.1 Coefficients:

```
# Extract and interpret coefficients
cat("Intercept ( ): ", coef(model.1)[1], "\n")
```

```
## Intercept ( ):  -0.9048516
```

```
cat("Coefficient for X ( ): ", coef(model.1)[2], "\n")
```

```
## Coefficient for X ( ):  0.5413593
```

13.6.0.2 Adjusted R-Squared and Goodness of Fit

```
# Adjusted R-squared
cat("Adjusted R-squared: ", summary(model)$adj.r.squared, "\n")
```

```
## Adjusted R-squared:  0.9133112
```

```
# Print the F-statistic and p-value
cat("F-statistic:", summary(model)$fstatistic[1], "\n")
```

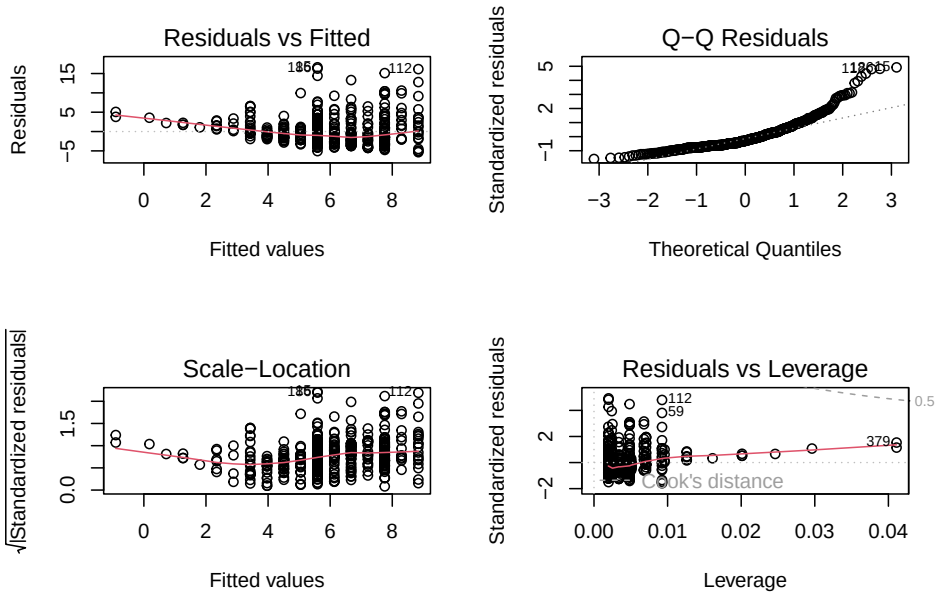
```
## F-statistic: 517.2401
```

```
p_val = pf(summary(model)$fstatistic[1],summary(model)$fstatistic[2],summary(model)$fstati
cat("p-value:", p_val, "\n")
```

```
## p-value: 2.364413e-27
```

13.6.0.3 Assumption Diagnostics

```
# Check OLS assumptions
par(mfrow = c(2, 2))
plot(model.1)
```



13.7 Practical: Hypothesis Testing and OLS

For this practical, we will make use of a dataset we used before: `Ch4PracticeA.xlsx`

13.8 Hypothesis Testing

1. Clean the dataset by removing missing values.
2. Is the **average health expenditure** per household significantly different from P15?

3. Do **households led by females** have different **poverty index scores** compared to male-headed households?
4. Is the proportion of **households with dirt floors** different from **50%**?

13.9 OLS

Examine relationships between household characteristics and economic outcomes.

1. Does education impact household poverty levels?
    - What is the coefficient of education?
    - Is the relationship statistically significant (p-value < 0.05)?
    - What is the R-squared value, and what does it tell us?
    - Check Assumption Diagnostics
2. Does health expenditure influence household poverty?
    - Does higher spending on healthcare correlate with higher or lower poverty levels?
    - What does this mean for healthcare affordability in low-income households?
    - Check Assumption Diagnostics

What were the main findings from the hypothesis tests?

How did education and health expenditures impact poverty levels?

What policies could improve household welfare based on your results?

Chapter 14

Multivariate Regression with Dummy Variables

14.0.0.1 Preliminaries

```
library(tidyverse)
library(lmtest)
```

```
library(wooldridge)
data(wage1)
sample1<-wage1
```

14.1 Dummy Variables

- Binary variables defined as 0 or 1;

- D=1 if the criterion is satisfied
- D=0 if not
  - Convey qualitative info
  - Examples: female, married, graduated, passed
  - Example: variable female where female=1 for females, female=0 for males
- "gender" is not a good variable name because it is not clear if gender=1 for male or femal
- Can be an independent variable or dependent variable in a probit/logit model

Code	Definition
`data <- data %>% mutate(response_dummy = if_else(response == "yes", 1, 0))`	This is when you generate a new variable in your dataset. You can change "yes" to female or married, etc.

Code	Definition
`Dataset_CP_modified <- Dataset_CP %>%mutate(Q37a_HoursWeekChildren_level = case_when (Q37a_HoursWeekChildren <= 5 ~ "low",Q37a_HoursWeekChildren > 5 & Q37a_HoursWeekChildren <= 15 ~ "medium",Q37a_HoursWeekChildren > 15 ~ "high"))`	This is when you have numerous categories to represent dummy variables

Regression Models:

$\text{wage} = \beta_0 + \epsilon$

Since there is only the intercept, there are no independent variables:

```
null<-lm(wage~1, data=sample1)
summary(null)
```

```
##
## Call:
## lm(formula = wage ~ 1, data = sample1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.3661 -2.5661 -1.2461  0.9839 19.0839
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.896      0.161   36.62   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.693 on 525 degrees of freedom
```

This shows the average wage.

Now, let's look at when the independent variable is represented by a binary variable (1,0)

wage $= \beta_0 + \delta_0$female $+ \epsilon$

- The coefficient is the difference in wage for females compared to males
- This model has different intercepts for females ($\beta_0 + \delta_0$) and males β_0
- The intercept for females is the average wage for females and the intercept for males is the average wage for males
- The regression t-test for significance of the coefficient on female is identical to the t-test for significant differences in wage between the female group and the male group, with the same mean, t-statistic, and p-value.

```
single<-lm(wage~sample1$female, data = sample1)
summary(single)
```

```
##
## Call:
## lm(formula = wage ~ sample1$female, data = sample1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.5995 -1.8495 -0.9877  1.4260 17.8805
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      7.0995     0.2100  33.806  < 2e-16 ***
## sample1$female  -2.5118     0.3034  -8.279 1.04e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.476 on 524 degrees of freedom
## Multiple R-squared:  0.1157, Adjusted R-squared:  0.114
## F-statistic: 68.54 on 1 and 524 DF,  p-value: 1.042e-15
```

We can plot the regression line;

```
sample1 %>%
  ggplot(aes(x=female, y=wage))+
  geom_point()+
  geom_smooth(method="lm", se=FALSE)+
  theme_minimal()+
  labs(x="Gender", y="Wage",
```

```
       title="Scatter Plot with Regression Line")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

The figure shows the slope `-2.51` and the intercepts: 7.10 and 4.59

Results:

Variables	Wage	Wage
female		`-2.51***`
intercept	`5.90***`	`7.10***`

We introduced a female dummy variable (1=female, 0 otherwise (known as **intercept dummy**)

This specification says that the intercept (or the average difference in wages between females and males) is different.

Intercept, β_0 , measures the intercept of the default group (males) and $\beta_0+\delta_0$ is the intercept for females

$\beta_0 = 5.90$ is the average value of wage.

$\delta_0 = -2.51$ is the effect of the indicator variable female on wage

- Females have 2.51 lower wages than males. The interpretation of the coefficient is with respect to the reference/base category of males
- The intercept or average wage for males is $\beta_0 = 7.10$
- The average wage for females is $\beta_0 + \delta_0 = 7.10 + (-2.51) = 4.59$

14.1.0.1 Dummy variable trap

- This refers to the problem that not all categories can be included in the regression and one category needs to be left out, which is called a base or reference category.
- For example, male and female cannot be both included in the regression because of perfect collinearity
- Including male instead of female in the regression:

  wage $= \beta_0 + \delta_0(1 - male) + \epsilon =$
- wage = $(\beta_0 + \delta_0)$ - $\delta_0$male + ϵ
- The coefficient $-\delta_0$ on male has the same magnitude and significance, but opposite sign from the coefficient on female.

- The intercept in the model with male is $(\beta_0 + \delta_0)$, which can also be obtained from the model with female (average wage of females)
- A regression can be estimated with both male and female but with no constant (this approach is not commonly used)

14.1.1 Interactions with another dummy variable

Interaction term - multiply variable by dummy variable

– Interaction terms for variables *female* and *married* can be done in two different ways.

(1) include *female* and *married* and `female*married` in the regression

$$\text{wage} = \beta_0 + \beta_1 female + \beta_2 married + \beta_3 female * married + \epsilon$$

(2) Create four categories: `female*single`, `male*single`, `female*married`, and `male*married`; include 3 of them in the regression model (`male*single` is the reference category).

- **Note:** Choice of reference category depends on you but commonly, the most number of observations in the category is chosen.

```
sample_modified <- sample1 %>%
  mutate(female_married = if_else(female == 1 & married == 1, 1, 0),
         male_single = if_else(female == 0 & married == 0, 1, 0),
         male_married = if_else(female == 0 & married == 1, 1, 0),
         female_single = if_else(female==1 & married == 0, 1,0))
head(sample_modified)
```

```
##   wage educ exper tenure nonwhite female married numdep smsa northcen south west c
## 1 3.10   11     2      0        0      1       0      2    1        0     0    1
## 2 3.24   12    22      2        0      1       1      3    1        0     0    1
## 3 3.00   11     2      0        0      0       0      2    0        0     0    1
## 4 6.00    8    44     28        0      0       1      0    1        0     0    1
## 5 5.30   12     7      2        0      0       1      1    0        0     0    1
## 6 8.75   16     9      8        0      0       1      0    1        0     0    1
##   services profserv profocc clerocc servocc    lwage expersq tenursq female_marrie
## 1        0        0       0       0       0 1.131402       4       0
## 2        1        0       0       0       1 1.175573     484       4
## 3        0        0       0       0       0 1.098612       4       0
## 4        0        0       0       1       0 1.791759    1936     784
## 5        0        0       0       0       0 1.667707      49       4
## 6        0        1       1       0       0 2.169054      81      64
##   female_single
## 1             1
## 2             0
## 3             0
## 4             0
## 5             0
## 6             0
```

Regression model:

$wage = \beta_0 + \beta_1 femalesingle + \beta_2 femalemarried + \beta_3 malemarried + \epsilon$

You need to drop the reference category variable which we already did when we dropped `malesingle`

```
interaction<-lm(wage~female_married+female_single+male_married, data=sample_modi
summary(interaction)
```

```
##
## Call:
## lm(formula = wage ~ female_married + female_single + male_married,
##     data = sample_modified)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7530 -1.7327 -0.9973  1.2566 17.0184
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.1680     0.3614  14.299  < 2e-16 ***
## female_married  -0.6021     0.4645  -1.296    0.195
## female_single   -0.5564     0.4736  -1.175    0.241
## male_married     2.8150     0.4363   6.451 2.53e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.352 on 522 degrees of freedom
## Multiple R-squared:  0.181,  Adjusted R-squared:  0.1763
## F-statistic: 38.45 on 3 and 522 DF,  p-value: < 2.2e-16
```

Regression model:

$$wage = \beta_0 + \beta_1 female + \beta_2 married + \beta_3 femalemarried + \epsilon$$

```r
int1<-lm(wage~female+married+female_married, data=sample_modified)
summary(int1)
```

```
##
## Call:
## lm(formula = wage ~ female + married + female_married, data = sample_modified)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7530 -1.7327 -0.9973  1.2566 17.0184
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.1680     0.3614  14.299  < 2e-16 ***
## female          -0.5564     0.4736  -1.175    0.241
## married          2.8150     0.4363   6.451 2.53e-10 ***
## female_married  -2.8607     0.6076  -4.708 3.20e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.352 on 522 degrees of freedom
## Multiple R-squared:  0.181,  Adjusted R-squared:  0.1763
## F-statistic: 38.45 on 3 and 522 DF,  p-value: < 2.2e-16
```

Let's have pretty regression results table:

```
if(!("stargazer" %in% installed.packages()[,"Package"])) install.packages("starga
```

```
library(stargazer)
# Create a regression table
stargazer(null, interaction, int1, #regression
          type = "text", title = "Regression Results",
          dep.var.labels = "Wage",
          #covariate.labels = c("B1", "B2", "BN"), label your independent variab
          omit.stat = c("f", "ser"), digits = 3) # Omits F-statistic and standar
```

```
##
## Regression Results
## ============================================
##                       Dependent variable:
##                   ---------------------------
##                               Wage
##                     (1)       (2)       (3)
## --------------------------------------------
## female                                -0.556
##                                       (0.474)
##
## married                              2.815***
##                                       (0.436)
##
## female_married             -0.602   -2.861***
##                            (0.464)   (0.608)
##
```

```
## female_single             -0.556
##                           (0.474)
##
## male_married             2.815***
##                           (0.436)
##
## Constant       5.896***  5.168***  5.168***
##                 (0.161)   (0.361)   (0.361)
##
## ------------------------------------------
## Observations      526       526       526
## R2               0.000     0.181     0.181
## Adjusted R2      0.000     0.176     0.176
## ==========================================
## Note:            *p<0.1; **p<0.05; ***p<0.01
```

Interpretation of Results:

Model 1: The baseline wage when all other variables are absent. The reference group is single males.

Model 2:

- Female, Single: -0.556 (not significant) –> Single females earn 0.556 less than single males, but the result is not statistically significant.
- Male, Married: 2.815^{***} –> Married males earn 2.815 more than single males
- Constant: 5.168^{***} –> The wage for single males

Model 3:

- Female: -0.556 (not significant) –> Being female alone does not significantly impact wages
- Married: 2.815*** –> Being married increases wages
- Female x Married Interaction: -2.861*** –> Being a married female results in a decrease of 2,861 on wages.
- Constant: 5.168*** –> The wage for single males

Model Fit:

Model 1: No explanatory power

Model 2 and 3: The models explain 18.1% of the variation in wages.

14.1.2 Several related dummy variables

- Regression with several related dummy variables needs to have one reference category.
- For example: Dummies for regions: northcentral, south, west. So, east is the reference category. Thus, $east = 1 - northcentral - south - west$

$$wage = \beta_0 + \beta_1 northcentral + \beta_2 south + \beta_3 west + \epsilon$$

```
reg1<-lm(wage~northcen+south+west, data=sample_modified)
summary(reg1)
```

```
##
```

```
## Call:
## lm(formula = wage ~ northcen + south + west, data = sample_modified)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.083 -2.387 -1.097  1.157 18.610
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.3697     0.3380  18.848   <2e-16 ***
## northcen     -0.6593     0.4651  -1.418   0.1569
## south        -0.9828     0.4316  -2.277   0.0232 *
## west          0.2436     0.5154   0.473   0.6366
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.671 on 522 degrees of freedom
## Multiple R-squared:  0.0175, Adjusted R-squared:  0.01185
## F-statistic: 3.099 on 3 and 522 DF,  p-value: 0.02646
```

Region	Average Wage
Northcentral	5.71
South	5.39
West	6.61
East	6.37

Wages in the south are 0.98 lower than wages in the east, and wages in the northcentral and west are not significantly different from those in the east region.

14.1.3 Dummy Variable and Continuous Variable in Regression

$wage = \beta_0 + \beta_1 educ + \epsilon$

This model has the same intercept and slope on education for females and males

$wage = \beta_0 + \beta_1 educ + \delta_0 female + \epsilon$

- The effect of female on wage $= \delta_0$
- The same slope for females and males $= \beta_1$
- Intercept for males $= \beta_0$
- Intercept for females $= \beta_0 + \delta_0$

```
withedu<-lm(wage~educ+female, data=sample_modified)
summary(withedu)
```

```
##
## Call:
## lm(formula = wage ~ educ + female, data = sample_modified)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -5.9890 -1.8702 -0.6651  1.0447 15.4998
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.62282    0.67253   0.926    0.355
## educ         0.50645    0.05039  10.051  < 2e-16 ***
## female      -2.27336    0.27904  -8.147 2.76e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.186 on 523 degrees of freedom
## Multiple R-squared:  0.2588, Adjusted R-squared:  0.256
## F-statistic: 91.32 on 2 and 523 DF,  p-value: < 2.2e-16
```

Results:

Variables	wage	female dummy wage	male dummy wage
educ	0.54***	0.51***	0.51***
female		-2.27***	
male			2.27***
Intercept	-0.91	0.62	-1.65**

Plotting the graph:

```
ggplot(sample_modified, aes(x = educ, y = wage, color = as.factor(female))) + #differenti
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE) +
```

```
  labs(color = "Gender") +
  theme_minimal() +
  ggtitle("Regression of Wage on Education, Differentiated by Gender")
```

14.1.4 Interaction terms with non-dummy variables

$wage = \beta_0 + \beta_1 educ + \delta_0 female + \delta_1 female * educ + \epsilon$

- This model has different slopes on education and intercepts for females and males
- slope for males = β_1
- slope for females = $\beta_1 + \delta_1$
- intercept for males = β_0

- intercept for females $= \beta_0 + \delta_0$

```
model <- lm(wage ~ educ + female + female:educ, data = sample_modified)
summary(model) #you can create interaction term when regressing by using ':'
```

```
##
## Call:
## lm(formula = wage ~ educ + female + female:educ, data = sample_modified)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.1611 -1.8028 -0.6367  1.0054 15.5258
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.20050    0.84356   0.238    0.812
## educ         0.53948    0.06422   8.400 4.24e-16 ***
## female      -1.19852    1.32504  -0.905    0.366
## educ:female -0.08600    0.10364  -0.830    0.407
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.186 on 522 degrees of freedom
## Multiple R-squared:  0.2598, Adjusted R-squared:  0.2555
## F-statistic: 61.07 on 3 and 522 DF,  p-value: < 2.2e-16
```

Interpretation of Results:

- Slope for males is 0.54
- Slope for females is 0.45
- Intercept for males is 0.20
- Intercept for females is -1.00
- The coefficient on female and on the interaction term between female and education are not significant. So, the intercepts and the slopes of returns to education are not significantly different for females and males.

14.1.5 F-test for differences across groups

F-test is used to test whether the returns to education, experience, and tenure are the same for two groups (males and females).

H_0 : The coefficients on interaction terms are all zero, meaning that the impact of non-dummy variables on dependent variables is the same for the two groups.

H_a : At least one of the coefficients is not zero, meaning, the impact is different for the two groups.

Step 1: Fit the Full and Reduced Models

```
full<-lm(wage~educ+exper+tenure+female+female:educ+female:exper+female:tenure, da
```

This model includes interactiono terms, allowing the effects of continuous variables to be different for females.

```
reduced<-lm(wage~educ+exper+tenure+female, data=sample_modified)
```

This model assumes education, experience, and tenure have the same impact for males and females but allows for general difference in wage levels through female coefficient.

Step 2: Perform the F-Test

```
result<-anova(reduced,full)
print(result)
```

```
## Analysis of Variance Table
##
## Model 1: wage ~ educ + exper + tenure + female
## Model 2: wage ~ educ + exper + tenure + female + female:educ + female:exper +
##     female:tenure
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    521 4557.3
## 2    518 4394.2  3    163.06 6.4074 0.0002878 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This compares the full and reduced model; If the p-value<0.05, reject the null. In this case, Females have significantly different wages than males.

Chapter 15

Formal Tests on Assumptions

15.1 Multicollinearity

Multicollinearity occurs when independent variables in a regression model are highly correlated, making coefficient estimates unreliable.

1. Variance Inflation Factor (VIF)
    - If VIF > 10, multicollinearity may be a problem.

```
if(!("car" %in% installed.packages()[,"Package"])) install.packages("car")
```

```
library(car)
vif(full)
```

```
## there are higher-order terms (interactions) in this model
## consider setting type = 'predictor'; see ?vif
```

```
##          educ         exper        tenure        female   educ:female  exper:f
##      1.918370      3.283682      2.121188     30.508011     25.140337      5.1
```

How to fix?

1. Remove or combine highly correlated variables
2. Do Principal Component Analysis or Factor Analysis

15.2 Heteroscedasticity

The variance of residuals is not constant across observations.

1. Breusch-Pagan Test
    - H_0 : Homoscedasticity
    - H_a : Heteroscedasticity
2. White Test
    - Detects both heteroscedasticity and model misspecification

```
#Breusch-Pagan Test
library(lmtest)
bptest(full)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  full
## BP = 46.768, df = 7, p-value = 6.195e-08
```

Heteroscedasticity present.

```
#White Test
library(car)
ncvTest(full)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 158.2722, Df = 1, p = < 2.22e-16
```

Heteroscedasticity present.

How to fix?

1. Use robust standard errors.

```
if(!("sandwich" %in% installed.packages()[,"Package"])) install.packages("sandwich")
```

```
library(sandwich)
library(lmtest)

coeftest(full, vcov = vcovHC(model, type = "HC1"))  # Huber-White robust SEs)
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) -3.531140   0.871723 -4.0508 5.887e-05 ***
```

```
## educ          0.677180   0.073396  9.2264 < 2.2e-16 ***
## female        2.075290   1.460808  1.4206   0.15602
## educ:female -0.213688   0.123818 -1.7258   0.08498 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2. Transform the dependent variable (log transformation)
3. Weighted Least Squares

15.3 Specification Errors

The model omits relevant variables, includes irrelevant ones or has the wrong functional form.

1. Ramsey RESET Test
    - Tests for omitted variables or incorrect functional form.
    - H_0 : The model is correctly specified
    - H_a : The model may be misspecified

```
library(lmtest)
resettest(full, power = 2:3) #tests for quadratic and cubic functional form erro
```

```
##
##  RESET test
##
## data:  full
## RESET = 10.012, df1 = 2, df2 = 516, p-value = 5.421e-05
```

Model is misspecified.

How to Fix?

1. Include relevant missing variables
2. Try polynomial or interaction terms if relationships are nonlinear
3. Check for categorical variable misclassification

15.4 Alternative Functional Forms

15.4.1 Log-Log Model

- The slope coefficient β_1 measures the elasticity of Y with respect to X.
- The percentage change in Y for a given percentage change in X.
- The simplest way to decide whether the log-log model fits the data is to plot the scatterplot of `lnY` against `X` and see if the points lie approx. on a straight line.

```
library(readxl)
CEOSAL1 <- read_excel("CEOSAL1.xls")
ceovars<-c("salary","sales")
CEOSAL2<-CEOSAL1[ceovars]

#Create the log of a variable using the log function
CEOSAL2<-CEOSAL2%>%mutate(lsales=log(sales),lsalary=log(salary))
```

```
log2<-lm(lsalary~lsales, data=CEOSAL2)
summary(log2)
```

```
##
## Call:
## lm(formula = lsalary ~ lsales, data = CEOSAL2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.01038 -0.28140 -0.02723  0.21222  2.81128
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.82200    0.28834  16.723  < 2e-16 ***
## lsales       0.25667    0.03452   7.436  2.7e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5044 on 207 degrees of freedom
## Multiple R-squared:  0.2108, Adjusted R-squared:  0.207
## F-statistic:  55.3 on 1 and 207 DF,  p-value: 2.703e-12
```

```
#You can also do it in the regression
#log2<-lm(log(salary)~log(sales),data=CEOSAL2)
```

15.4.2 Log-Lin Model

- Useful in finding out the rate of growth of certain economic variables.
- The slope coefficient measures the constant proportional or relative change in Y for a given absolute change in the value of the regressor.

  $\beta_1 = \frac{\text{relative change in regressand}}{\text{absolute change in regressor}}$
- If we multiply the relative change in Y by 100, it will give the percentage change or growth rate

```
loglin<-lm(lsalary~sales,data = CEOSAL2)
summary(loglin)
```

```
##
## Call:
## lm(formula = lsalary ~ sales, data = CEOSAL2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.44220 -0.29159 -0.02837  0.28323  2.72487
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.847e+00  4.500e-02 152.138  < 2e-16 ***
## sales       1.498e-05  3.553e-06   4.217  3.7e-05 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5448 on 207 degrees of freedom
## Multiple R-squared:  0.07912,    Adjusted R-squared:  0.07467
## F-statistic: 17.79 on 1 and 207 DF,  p-value: 3.696e-05
```

15.4.3 Lin-Log Model

- The absolute change in Y is equal to the slope times the relative change in X. If the relative change in X is multiplied by 100, then the equation above gives the absolute change in Y for a percentage change in X.

```
linlog<-lm(salary~lsales, data=CEOSAL2)
summary(linlog)
```

```
##
## Call:
## lm(formula = salary ~ lsales, data = CEOSAL2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1072.1  -447.6  -222.8    41.7 13702.5
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -898.93     771.50  -1.165  0.24529
## lsales        262.90      92.36   2.847  0.00486 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1349 on 207 degrees of freedom
## Multiple R-squared:  0.03767,    Adjusted R-squared:  0.03302
## F-statistic: 8.103 on 1 and 207 DF,  p-value: 0.004863
```

15.5 Practical: Multivariate (Dummy) Regression and Formal Tests of Assumptions

For this problem set, we are going to replicate the findings of the study entitled, "Development from Representation? A Study of Quotas for the Scheduled Castes in India" by Francesca Jensenius published in the American Economic Journal: Applied Economics. This study looks into the effects of an electoral quota implemented in 1974 for scheduled castes (SC), a historically marginalised group in India. Because of the scheduled castes' weak position in society, they have difficulty in securing many political seats. For this problem set, we refer to "reserved constituencies" as the constituencies where only members of the scheduled caste can run for elections while the whole electorate votes regardless of their caste. We refer to "general constituencies" as the constituencies that did not have a quota. We use the `quotas.csv and quotas codebook` in Visualizations Module.

Preliminaries:

To be uniform, create a variable where a constituency that is a reserved one is labelled "1" and a general constituency is labelled "0"

1. Conduct a descriptive statistics using psych package on the data; create a histogram of the newly created variable.
2. Conduct an OLS regression with the following model. The regression outcome is **the percentage of SCs in rural areas in AC living in a village with an educational institution (P_educVD01_sc). The main variable of interest is the type of constituency. The control factors are tot_pop71_true, SC_percent71_true, Plit71, Plit71_SC, P_W71, P_al71_SC, P_al71_nonSC. Create a regression line with only the main variable of interest.**
3. Interpret only the main variable of interest and the R-squared. Tell me why you think the control factors were added in the context of Econometrics and in terms of economics.
4. With the regression results, conduct a variance inflation factor test.
5. Interpret your results and what the test measures.
6. With the regression results, conduct a Breusch-Pagan test or White's test.
7. Interpret your results and what the test measures.
8. With the regression results, conduct a Ramsey Reset test.
9. Interpret your results and what the test measures.

Chapter 16

Additional Topics

This lecture introduces using OLS regression to predict outcome and key steps in working with primary survey data, panel data and time series forecasting using R.

16.1 Predicting Outcome using OLS

16.1.1 Simulated Data

```
set.seed(123)
x<-runif(30, min=10000, max=70000)
y<-12000+0.76*x+rnorm(30, mean=0, sd=5000)
sdf<-data.frame(Y=y, X=x)
head(sdf)
```

```
##          Y        X
```

```
## 1 41648.10 27254.65
## 2 58035.97 57298.31
## 3 28416.26 34538.62
## 4 63372.37 62981.04
## 5 60121.35 66428.04
## 6 16338.26 12733.39
```

16.1.2 OLS

```
sim_ols<-lm(Y~X, data=sdf)
summary(sim_ols)
```

```
##
## Call:
## lm(formula = Y ~ X, data = sdf)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10714.7  -3601.4     97.7   3528.1   9406.1
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.556e+04  2.494e+03   6.237 9.69e-07 ***
## X           6.825e-01  5.245e-02  13.012 2.15e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 4938 on 28 degrees of freedom
## Multiple R-squared:  0.8581, Adjusted R-squared:  0.853
## F-statistic: 169.3 on 1 and 28 DF,  p-value: 2.153e-13
```

16.1.3 Use `predict()`

Let us say we want to find out what Y is given a certain value of X, like say 50000

```
new<-data.frame(X=50000)
predict(sim_ols, newdata=new)
```

```
##        1
## 49683.44
```

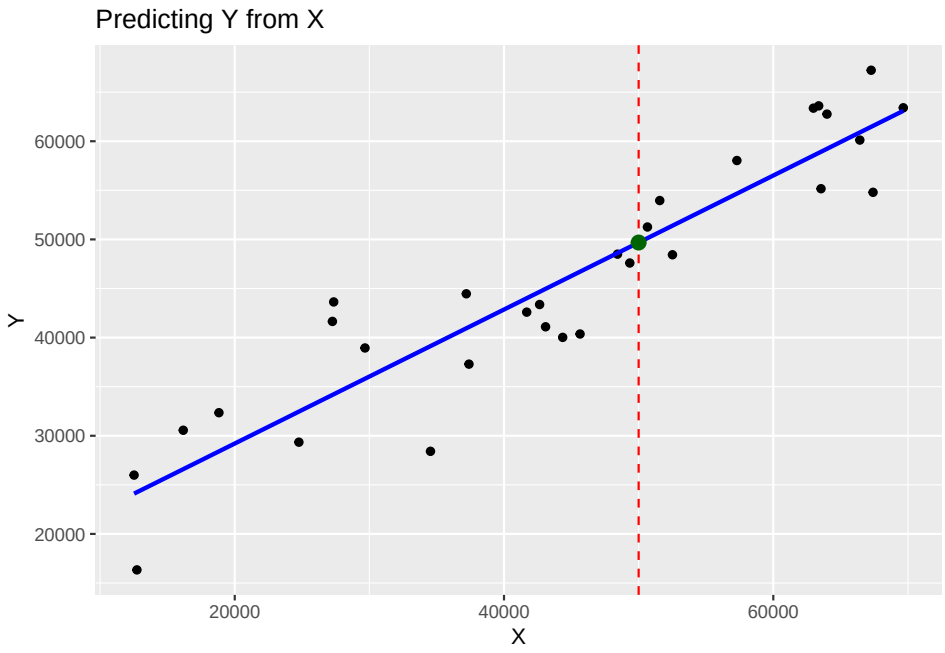
You need to create a new data frame for this.

16.1.4 Extra: Visualization

```
library(ggplot2)

ggplot(sdf, aes(x = X, y = Y)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  geom_vline(xintercept = 50000, linetype = "dashed", color = "red") +
  geom_point(data = new, aes(x = 50000, y = predict(sim_ols, new)), color = "darkgreen", s
```

```
  labs(title = "Predicting Y from X",
       x = "X", y = "Y")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



16.1.5 Real Data

We will use the `wooldridge` package and use `wage1`

```
library(wooldridge)
data(wage1)
```

We will do a multiple regression model

```r
wage<-lm(wage~educ+exper+tenure, data=wage1)
summary(wage)
```

```
##
## Call:
## lm(formula = wage ~ educ + exper + tenure, data = wage1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.6068 -1.7747 -0.6279  1.1969 14.6536
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.87273    0.72896  -3.941 9.22e-05 ***
## educ         0.59897    0.05128  11.679  < 2e-16 ***
## exper        0.02234    0.01206   1.853   0.0645 .
## tenure       0.16927    0.02164   7.820 2.93e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.084 on 522 degrees of freedom
## Multiple R-squared:  0.3064, Adjusted R-squared:  0.3024
## F-statistic: 76.87 on 3 and 522 DF,  p-value: < 2.2e-16
```

Say we want to see if someone had 16 years of education, 5 years of experience and 3 years of tenure, what will their wage be?

```
new2<-data.frame(
  educ=16,
  exper=5,
  tenure=3
)
predict(wage, newdata=new2)
```

```
##        1
## 7.33021
```

16.2 Designing and Testing a Survey

16.2.1 Calculating Sample Size

When conducting a survey, it's important to determine how many responses you need. Sample size depends on the population size, margin of error, confidence level and type of estimate: proportion or mean.

Sample Size Formula for Proportions

$$n = \frac{Z^2 \cdot p \cdot (1-p)}{E^2}$$

Where:

Z=Z-score (1.96 for 95% confidence)

p= estimated population proportion (use 0.5 if unknown)

E=desired margin of error

Scenario 1: You want to study how noise pollution affects life satisfaction among DLSU students. The student population is approximately 20,000. You want ± 5% precision at 95% confidence.

Please install `samplingbook` package since this will help us calculate sample size.

```
library(samplingbook)


sample.size.prop(e=0.05, N=20000, P=0.5, level=0.95)
```

```
##
## sample.size.prop object: Sample size for proportion estimate
## With finite population correction: N=20000, precision e=0.05 and expected proportion P=
##
## Sample size needed: 377
```

e is the ± 5% precision or the margin of error; it is the maximum acceptable difference between your sample estimate and the true population value.

Sample Size Formula for Means

$$n = \left(\frac{Z \cdot S}{E}\right)^2$$

Where:

S=estimated population standard deviation

E=margin of error

Scenario 2: You want to estimate the average life satisfaction score (1-5 scale), with ±0.2 precision, assuming SD=1.1

```
sample.size.mean(e=0.2, N=20000, S=1.1, level=0.95)
```

```
##
## sample.size.mean object: Sample size for mean estimate
## With finite population correction: N=20000, precision e=0.2 and standard devia
##
## Sample size needed: 116
```

16.2.2 Testing Survey Reliability - Cronbach's Alpha

Cronbach's alpha measures how consistent survey items (questions) work together to measure the same concept or construct.

When you ask multiple questions about a topic like noise pollution or life satisfaction, you want those questions to consistently reflect the same concept. Cronbach's alpha helps you test the reliability of your questionnaire before analyzing the results.

Aim for 0.70 or higher to be considered acceptable in social science research.

Example Questionnaire (5-point Likert Scale):

1. I am frequently disturbed by noise in my study areas.
2. Noise pollution negatively affects my concentration.
3. I feel more satisfied with life when my environment is quiet.
4. I find it difficult to relax due to noise around campus.
5. A quieter environment improves my overall well-being.

```r
library(psych)

# Simulate responses from 100 students
set.seed(123)
survey_data <- data.frame(
  Q1 = sample(1:5, 100, replace = TRUE),
  Q2 = sample(1:5, 100, replace = TRUE),
  Q3 = sample(1:5, 100, replace = TRUE),
  Q4 = sample(1:5, 100, replace = TRUE),
  Q5 = sample(1:5, 100, replace = TRUE)
)

# Calculate Cronbach's alpha
alpha(survey_data)
```

```
## Warning in alpha(survey_data): Some items were negatively correlated with the first pri
## should be reversed.
## To do this, run the function again with the 'check.keys=TRUE' option

## Some items ( Q1 Q2 ) were negatively correlated with the first principal component and
## probably should be reversed.
## To do this, run the function again with the 'check.keys=TRUE' option

##
## Reliability analysis
## Call: alpha(x = survey_data)
##
```

```
##   raw_alpha std.alpha G6(smc) average_r   S/N  ase mean   sd median_r
##       0.034     0.026    0.05    0.0052 0.026 0.15    3 0.65   0.0018
##
##     95% confidence boundaries
##          lower alpha upper
## Feldt    -0.30  0.03  0.30
## Duhachek -0.26  0.03  0.33
##
##  Reliability if an item is dropped:
##    raw_alpha std.alpha G6(smc) average_r    S/N alpha se  var.r   med.r
## Q1     0.072     0.072   0.080    0.0191  0.078      0.15 0.0097  0.0071
## Q2     0.034     0.029   0.047    0.0074  0.030      0.16 0.0102  0.0013
## Q3    -0.114    -0.127  -0.075   -0.0291 -0.113      0.18 0.0074 -0.0024
## Q4     0.083     0.081   0.065    0.0215  0.088      0.15 0.0015  0.0035
## Q5     0.039     0.028   0.058    0.0071  0.029      0.16 0.0153  0.0189
##
##  Item statistics
##      n raw.r std.r  r.cor  r.drop mean  sd
## Q1 100  0.41  0.41 -0.077 -0.0219  2.9 1.4
## Q2 100  0.48  0.45  0.066  0.0096  3.0 1.5
## Q3 100  0.55  0.54  0.521  0.1130  3.0 1.5
## Q4 100  0.37  0.41 -0.022 -0.0357  3.0 1.3
## Q5 100  0.44  0.45  0.019  0.0046  2.9 1.4
##
## Non missing response frequency for each item
##       1    2    3    4    5 miss
## Q1 0.21 0.20 0.23 0.17 0.19    0
```

```
## Q2 0.25 0.20 0.14 0.17 0.24    0
## Q3 0.21 0.24 0.16 0.14 0.25    0
## Q4 0.17 0.20 0.23 0.26 0.14    0
## Q5 0.22 0.19 0.22 0.19 0.18    0
```

Warning: Reverse-coded Items "Some items were negatively correlated with the first principal component…" –> This means Q1 and Q2 may be phrased in the opposite direction of the rest.

If students agree with Q3–Q5 but disagree with Q1–Q2, it will weaken reliability.

Fix: Recode these items or rerun with `check.keys = TRUE` to automatically detect and reverse them.

```
alpha(survey_data, check.keys = TRUE)
```

```
## Warning in alpha(survey_data, check.keys = TRUE): Some items were negatively correlate
##  This is indicated by a negative sign for the variable name.

##
## Reliability analysis
## Call: alpha(x = survey_data, check.keys = TRUE)
##
##   raw_alpha std.alpha G6(smc) average_r  S/N  ase mean  sd median_r
##        0.2      0.21     0.2     0.051 0.27 0.13    3 0.7   0.0095
##
##     95% confidence boundaries
##          lower alpha upper
```

```
## Feldt    -0.07   0.2  0.43
## Duhachek -0.04   0.2  0.45
##
##  Reliability if an item is dropped:
##     raw_alpha std.alpha G6(smc) average_r   S/N alpha se  var.r  med.r
## Q1-     0.151     0.161   0.145     0.046 0.191     0.14 0.0076 0.0071
## Q2-     0.191     0.196   0.169     0.057 0.244     0.13 0.0063 0.0095
## Q3      0.201     0.203   0.169     0.060 0.255     0.13 0.0041 0.0463
## Q4      0.039     0.041   0.036     0.010 0.042     0.16 0.0019 0.0058
## Q5      0.250     0.263   0.228     0.082 0.357     0.12 0.0073 0.1064
##
##  Item statistics
##       n raw.r std.r r.cor r.drop mean  sd
## Q1- 100  0.50  0.50 0.241 0.1086  3.1 1.4
## Q2- 100  0.50  0.48 0.178 0.0718  3.0 1.5
## Q3  100  0.48  0.47 0.174 0.0615  3.0 1.5
## Q4  100  0.56  0.59 0.474 0.2231  3.0 1.3
## Q5  100  0.41  0.42 0.018 0.0085  2.9 1.4
##
## Non missing response frequency for each item
##       1    2    3    4    5 miss
## Q1 0.21 0.20 0.23 0.17 0.19    0
## Q2 0.25 0.20 0.14 0.17 0.24    0
## Q3 0.21 0.24 0.16 0.14 0.25    0
## Q4 0.17 0.20 0.23 0.26 0.14    0
## Q5 0.22 0.19 0.22 0.19 0.18    0
```

Output Table 1: General Reliability Info

Column	Meaning
`raw_alpha`	Actual Cronbach's alpha (approx. 0.03 –> very low reliability)
`std.alpha`	Standardized alpha (if items are on different scales – not needed here)
`G6(smc)`	Guttman's lambda 6 (alternative to alpha)
`average_r`	Average correlation between all pairs of items
`S/N`	Signal-to-noise ratio (higher is better)
`ase`	Approx. standard error of alpha

Interpretation: Raw alpha = 0.03 means very poor reliability (items don't measure the same concept).

This is likely due to uncorrelated or negatively correlated items as the warning mentioned.

Output Table 2: Confidence Intervals

Method	Lower	Alpha	Upper
Feldt	-0.30	0.03	0.30
Duhachek	-0.26	0.03	0.33

It further strengthens the unreliability of the questions.

Output Table 3: Alpha if Item Dropped

Item	raw_alpha	average_r	Notes
Q3	-0.11	-0..03	Hurts reliability the most
Q4	0.08	0.02	Slight improvement
Q1,Q2,Q5	~0.03-0.07	~0.01-0.02	Not helping much

Use this to identify items that may be off-topic.

Output Table4: Correlations and Item Stats

Column	Meaning
`r.cor`	Correlation of each item with total score (ideal: >0.3)
`r.drop`	Correlation of each item excluding itself from the total
`mean`	Mean response
`sd`	Standard deviation

`r.cor` and `r.drop` should be positive and >0.3

Now, let us use a Likert dataset called `bfi` from the package `psych`.

```
library(psych)
data(bfi)


# Select 5 items
agree_items <- bfi[, c("A1", "A2", "A3", "A4", "A5")]


agree_items <- na.omit(agree_items)
alpha_result <- alpha(agree_items)
```

```
## Warning in alpha(agree_items): Some items were negatively correlated with the first pri
## should be reversed.
## To do this, run the function again with the 'check.keys=TRUE' option

## Some items ( A1 ) were negatively correlated with the first principal component and
## probably should be reversed.
## To do this, run the function again with the 'check.keys=TRUE' option
```

```
print(alpha_result)
```

```
##
## Reliability analysis
## Call: alpha(x = agree_items)
##
##   raw_alpha std.alpha G6(smc) average_r  S/N   ase mean   sd median_r
##       0.43      0.46    0.53      0.14 0.84 0.017  4.2 0.74     0.32
##
##     95% confidence boundaries
##          lower alpha upper
## Feldt      0.4  0.43  0.46
## Duhachek   0.4  0.43  0.46
##
##  Reliability if an item is dropped:
##    raw_alpha std.alpha G6(smc) average_r  S/N alpha se  var.r med.r
## A1      0.72      0.73    0.67     0.397 2.64   0.0089 0.0066 0.375
## A2      0.28      0.30    0.39     0.096 0.42   0.0223 0.1106 0.079
## A3      0.17      0.20    0.31     0.060 0.25   0.0253 0.1015 0.079
```

```
## A4      0.25      0.30    0.44     0.098 0.43   0.0233 0.1614 0.103
## A5      0.21      0.23    0.36     0.071 0.31   0.0242 0.1322 0.093
##
##  Item statistics
##       n raw.r std.r r.cor r.drop mean  sd
## A1 2709 0.055 0.021 -0.40  -0.31  2.4 1.4
## A2 2709 0.631 0.665  0.58   0.37  4.8 1.2
## A3 2709 0.728 0.743  0.72   0.48  4.6 1.3
## A4 2709 0.689 0.661  0.50   0.37  4.7 1.5
## A5 2709 0.701 0.718  0.64   0.45  4.6 1.3
##
## Non missing response frequency for each item
##       1    2    3    4    5    6 miss
## A1 0.33 0.30 0.14 0.12 0.08 0.03    0
## A2 0.02 0.05 0.05 0.20 0.37 0.31    0
## A3 0.03 0.06 0.08 0.20 0.36 0.27    0
## A4 0.05 0.08 0.07 0.16 0.24 0.41    0
## A5 0.02 0.07 0.09 0.22 0.35 0.25    0
```

16.3 Panel Data Models (Fixed vs. Random Effects)

We'll use the **plm** package to analyze panel data — where the same individuals (or units) are observed across multiple time periods.

16.3.1 Simulated Panel Dataset

Example: Suppose we surveyed 5 students in 3 consecutive years (2020-2022), we want to analyze how a variable `x` affects a variable `y`

```
library(plm)
set.seed(123)
panel_sim <- data.frame(
  id = rep(1:5, each = 3), #student
  year = rep(2020:2022, times = 5), #year
  x = rnorm(15, mean = 10),
  y = rnorm(15, mean = 50)
)

# Convert to panel format
pdata_sim <- pdata.frame(panel_sim, index = c("id", "year")) #needed so that plm can unde
```

16.3.2 Fixed Effects vs. Random Effects

Fixed Effects: This means the model only uses variation within each individual over time to estimate the effect of x on y.

In the code, it contains `"within"` which removes all time-invariant characteristics of each individual like gender.

When you are interested in **how changes within individuals** (over time) affect the outcome.

Random Effects: This assumes that individual-specific effects are random and uncorrelated with predictors. It allows estimation of time-invariant

variables like gender or region.

When you are interested in **comparing differences across individuals** and want to include time-invariant traits like gender or location.

```
#Fixed Effects Model
fe<-plm(y~x, data=pdata_sim, model = "within")
summary(fe)
```

```
## Oneway (individual) effect Within Model
##
## Call:
## plm(formula = y ~ x, data = pdata_sim, model = "within")
##
## Balanced Panel: n = 5, T = 3, N = 15
##
## Residuals:
##      Min.   1st Qu.    Median   3rd Qu.      Max.
## -1.122145 -0.725146 -0.017663  0.605705  1.312598
##
## Coefficients:
##   Estimate Std. Error t-value Pr(>|t|)
## x -0.84218    0.37470 -2.2476   0.0512 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    15.31
## Residual Sum of Squares: 9.8057
```

```
## R-Squared:      0.35952
## Adj. R-Squared: 0.003691
## F-statistic: 5.05187 on 1 and 9 DF, p-value: 0.051201
```

Interpretation: How does a change in x over time affect y for the specific individual?

```
#Random Effects Model
re<-plm(y~x, data=pdata_sim, model = "random")
summary(re)
```

```
## Oneway (individual) effect Random Effect Model
##    (Swamy-Arora's transformation)
##
## Call:
## plm(formula = y ~ x, data = pdata_sim, model = "random")
##
## Balanced Panel: n = 5, T = 3, N = 15
##
## Effects:
##                  var std.dev share
## idiosyncratic 1.090   1.044     1
## individual    0.000   0.000     0
## theta: 0
##
## Residuals:
##     Min.  1st Qu.   Median  3rd Qu.     Max.
```

```
## -1.63076 -0.83589  0.11735  0.72396  1.60535
##
## Coefficients:
##             Estimate Std. Error z-value Pr(>|z|)
## (Intercept) 55.85115    3.23285 17.2761  < 2e-16 ***
## x           -0.60062    0.31741 -1.8923  0.05845 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    16.712
## Residual Sum of Squares: 13.103
## R-Squared:      0.21596
## Adj. R-Squared: 0.15564
## Chisq: 3.5807 on 1 DF, p-value: 0.058455
```

Some notes on the results:

- A balanced model simply means that each individual has the same number of time observations
- FE and RE: When the residuals are centered around 0, it is a good thing.
- FE: A 1-unit increase in x is associated with a decrease of 0.84 in y for the same individual over time, with slight statistical significance
- FE: When it comes to the R-squared, 36% of the within-individual variation in y is explained by x.

- RE: The variance has the idiosyncratic, individual and theta. The effects suggests whether the random effects component (individual differences) add any explanatory power and since it's 0, it doesn't.
- RE: A 1-unit increase in x is associated with a 0.60 increase in y, with slight statistical significance.
- RE: When it comes to the R-squared, 21.6% of variation in y is explained by x.

Note: The individual effect variance is 0, and theta = 0, this is already a clue that Random Effects model adds nothing. We need to check

16.3.3 Hausman Test

If the p-value is greater than 0.05, we use the Random Effects but if p-value is less than 0.05, we use Fixed Effects.

```
phtest(fe, re)
```

```
##
##  Hausman Test
##
## data:  y ~ x
## chisq = 1.4717, df = 1, p-value = 0.2251
## alternative hypothesis: one model is inconsistent
```

The Hausman Test says that Random Effects is acceptable, though we saw that the variance (individual=0, theta=0), so what do we do? We can use pooled OLS.

Let's try a more realistic example with the dataset `Grunfeld` from the plm package

```
library(plm)
data("Grunfeld")
head(Grunfeld)
```

```
##   firm year   inv  value capital
## 1    1 1935 317.6 3078.5     2.8
## 2    1 1936 391.8 4661.7    52.6
## 3    1 1937 410.6 5387.1   156.9
## 4    1 1938 257.7 2792.2   209.2
## 5    1 1939 330.8 4313.2   203.4
## 6    1 1940 461.2 4643.9   207.2
```

We have 10 US firms over 20 years (1935-1954) with variables like inv (investment), value, and capital.

```
panel<-pdata.frame(Grunfeld, index = c("firm", "year")) #we need to identify the
```

```
#Fixed Effects
fe_ex<-plm(inv~value+capital, data = panel, model = "within")
summary(fe_ex)
```

```
## Oneway (individual) effect Within Model
##
## Call:
## plm(formula = inv ~ value + capital, data = panel, model = "within")
```

```
##
## Balanced Panel: n = 10, T = 20, N = 200
##
## Residuals:
##       Min.    1st Qu.     Median    3rd Qu.       Max.
## -184.00857  -17.64316    0.56337   19.19222  250.70974
##
## Coefficients:
##         Estimate Std. Error t-value  Pr(>|t|)
## value   0.110124   0.011857  9.2879 < 2.2e-16 ***
## capital 0.310065   0.017355 17.8666 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    2244400
## Residual Sum of Squares: 523480
## R-Squared:      0.76676
## Adj. R-Squared: 0.75311
## F-statistic: 309.014 on 2 and 188 DF, p-value: < 2.22e-16
```

Let's discuss the results together.

```
#Random Effects
re_ex<-plm(inv~value+capital, data=panel, model="random")
summary(re_ex)
```

```
## Oneway (individual) effect Random Effect Model
```

```
##    (Swamy-Arora's transformation)
##
## Call:
## plm(formula = inv ~ value + capital, data = panel, model = "random")
##
## Balanced Panel: n = 10, T = 20, N = 200
##
## Effects:
##                   var std.dev share
## idiosyncratic 2784.46   52.77 0.282
## individual    7089.80   84.20 0.718
## theta: 0.8612
##
## Residuals:
##      Min.   1st Qu.    Median   3rd Qu.      Max.
## -177.6063  -19.7350    4.6851   19.5105  252.8743
##
## Coefficients:
##               Estimate Std. Error z-value Pr(>|z|)
## (Intercept) -57.834415  28.898935 -2.0013  0.04536 *
## value         0.109781   0.010493 10.4627  < 2e-16 ***
## capital       0.308113   0.017180 17.9339  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    2381400
## Residual Sum of Squares: 548900
```

```
## R-Squared:      0.7695
## Adj. R-Squared: 0.76716
## Chisq: 657.674 on 2 DF, p-value: < 2.22e-16
```

Let's discuss the results together.

```
phtest(fe_ex, re_ex)
```

```
##
##  Hausman Test
##
## data:  inv ~ value + capital
## chisq = 2.3304, df = 2, p-value = 0.3119
## alternative hypothesis: one model is inconsistent
```

Let's discuss the results together.

16.4 Lagged Values and Preliminaries in Forecasting

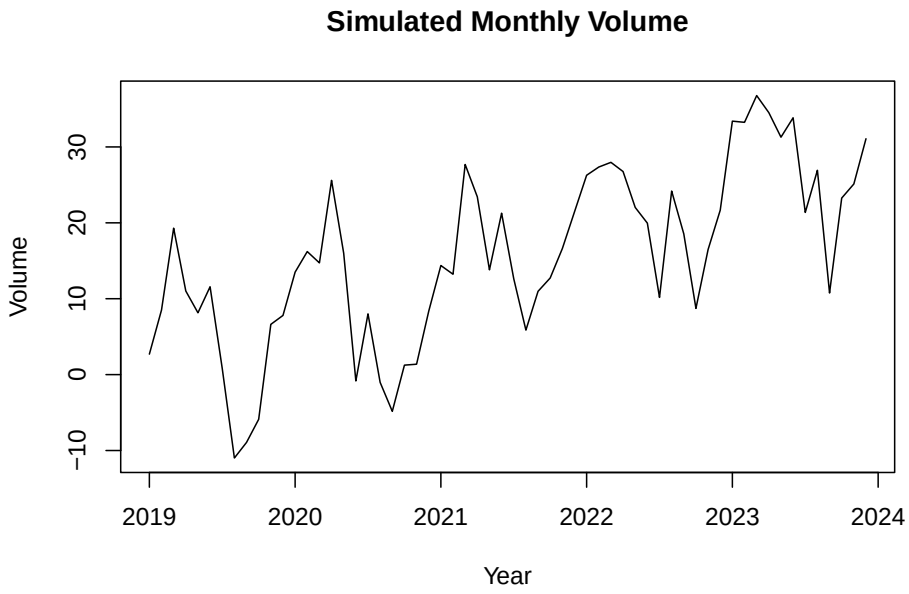
Lagged values are simply past values of a time series. So, if we have `Lag 1`, it means previous time period then so on and so forth. Lagged values help capture temporal dependence which is important in time series forecasting. For this section, install and load the `forecast` package and load the tidyverse package and the zoo package.

```
library(tidyverse)
library(forecast)
library(zoo)
```

16.4.1 Simulated Time Series

We will create a simulated monthly volume data with some trend and noise.

```
set.seed(123)
n<-60

#Simulate a trend + seasonal + random noise time series
time<-1:n
trend <- 0.5 * time #creates a linear trend that means as time increases, the sa
seasonality <- 10 * sin(2 * pi * time / 12) #this adds ups and downs each year t
noise <- rnorm(n, mean = 0, sd = 5) #adds random fluctuations so it closely rese
volume <- trend + seasonality + noise #this makes the sales data

sim_ts <- ts(volume, start = c(2019, 1), frequency = 12) #this creates the time
plot(sim_ts, main = "Simulated Monthly Volume", ylab = "Volume", xlab = "Year")
```



Now, the next step is to create lagged values to understand past-dependence.

16.4.1.1 Create Tibble

```
# Create prettier data frames with tibble
datats <- tibble(
  time = as.yearmon(time(sim_ts)),
  volume = as.numeric(sim_ts),
  lag1 = lag(volume, 1),
  lag2 = lag(volume, 2)
)


head(datats, 10)
```

```
## # A tibble: 10 x 4
##    time       volume    lag1    lag2
##    <yearmon>   <dbl>   <dbl>   <dbl>
##  1 Jan 2019    2.70   NA      NA
##  2 Feb 2019    8.51    2.70   NA
##  3 Mar 2019   19.3     8.51    2.70
##  4 Apr 2019   11.0    19.3     8.51
##  5 May 2019    8.15   11.0    19.3
##  6 Jun 2019   11.6     8.15   11.0
##  7 Jul 2019    0.805  11.6     8.15
##  8 Aug 2019  -11.0     0.805  11.6
##  9 Sept 2019  -8.93  -11.0     0.805
## 10 Oct 2019   -5.89   -8.93  -11.0
```

The lagged values are useful for building autoregressive models.

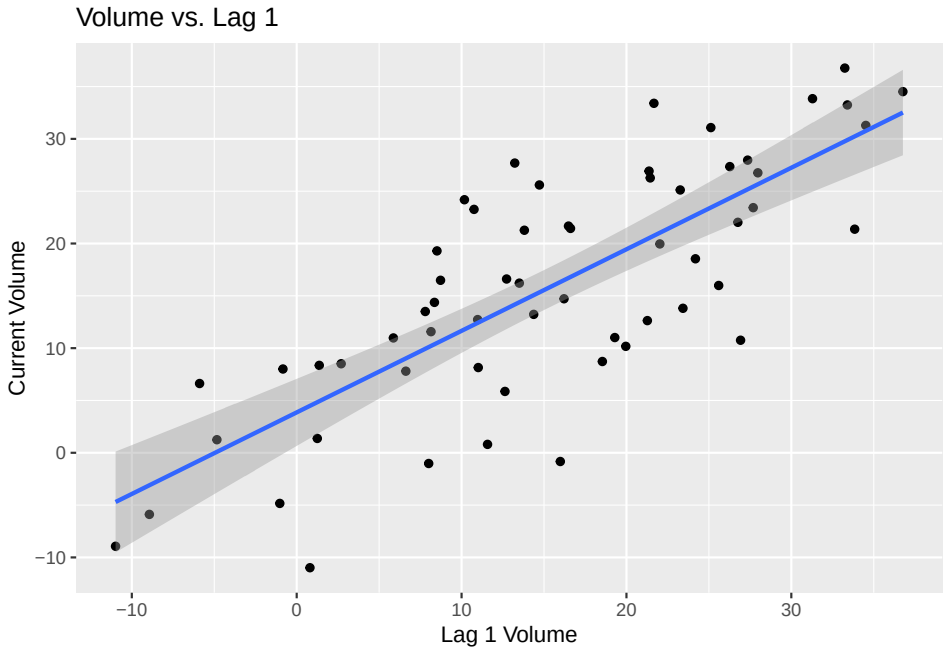
16.4.1.2 Scatterplot

```
ggplot(datats, aes(x = lag1, y = volume)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "Volume vs. Lag 1", x = "Lag 1 Volume", y = "Current Volume")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 1 row containing non-finite outside the scale range (`stat_sm
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range (`g
```

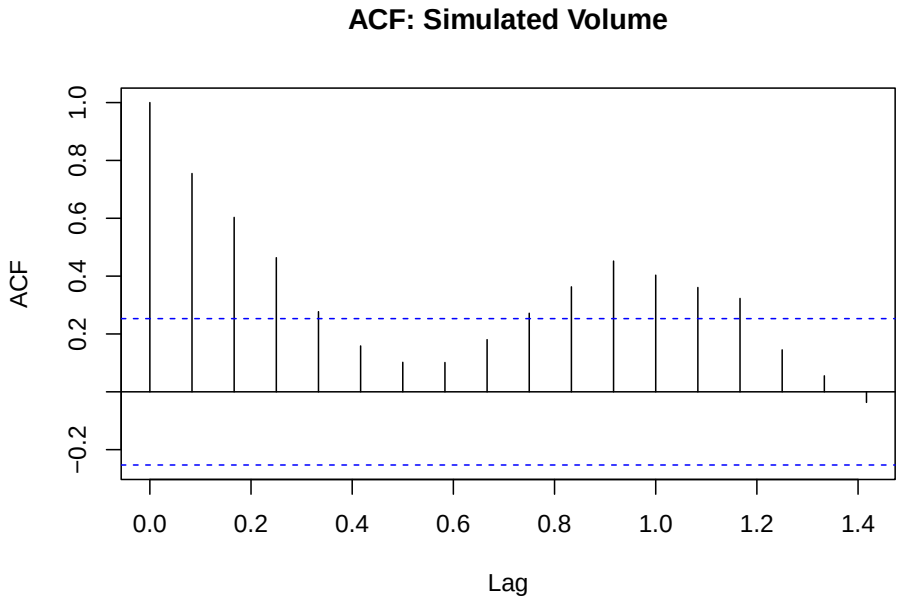


There should be a pattern to see whether the data is good for forecasting. The scatterplot does show a pattern but, if you want to be sure, we can use the `auto.arima()` in the forecast package (which will be done later).

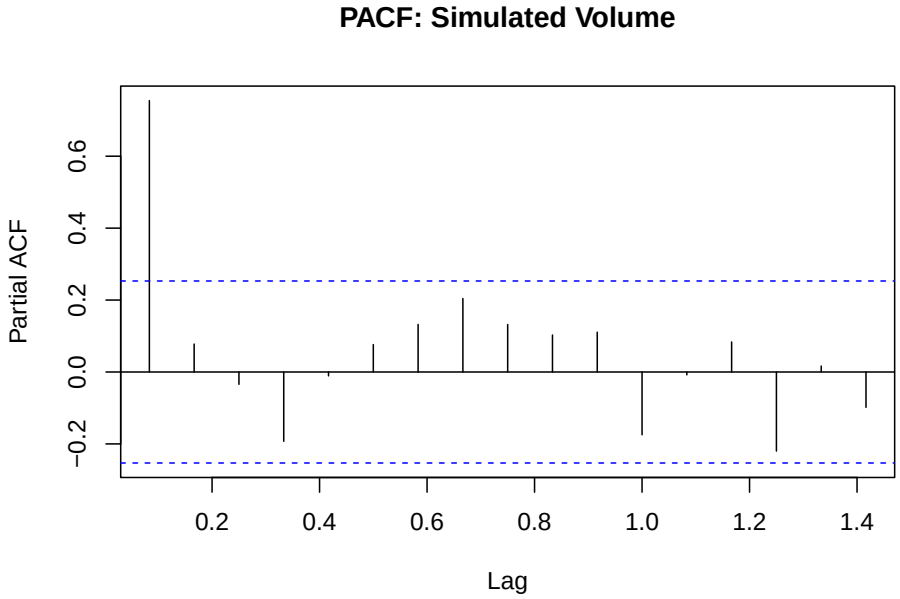
Let's see if the data has seasonality, it is necessary to difference and what the AR order (p) that will be used.

16.4.1.3 ACF and PACF

```
# ACF and PACF
acf(sim_ts, main = "ACF: Simulated Volume")
```




```
pacf(sim_ts, main = "PACF: Simulated Volume")
```



In the ACF: If bars cross the blue lines, it means the autocorrelation at that lag is statistically significant (meaning, these past values influence the present). We also do not see a sudden cutoff because if we do, we will have the MA process (we use the AR or ARIMA).

In the PACF: We look at bars outside the blue lines so we see it at Lag 1 which means that there is strong direct relationship between current value and the previous value. When all other lags are within the blue lines, it is typical of the AR(1) process.

16.4.1.4 Stationarity (ADF test)

We need to install and load the `tseries` package for the Augmented-Dickey Fuller test to check if our data is stationary. If p-value is greater than 0.05, we need differencing.

```
library(tseries)
adf.test(sim_ts)
```

```
## Warning in adf.test(sim_ts): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  sim_ts
## Dickey-Fuller = -5.9266, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

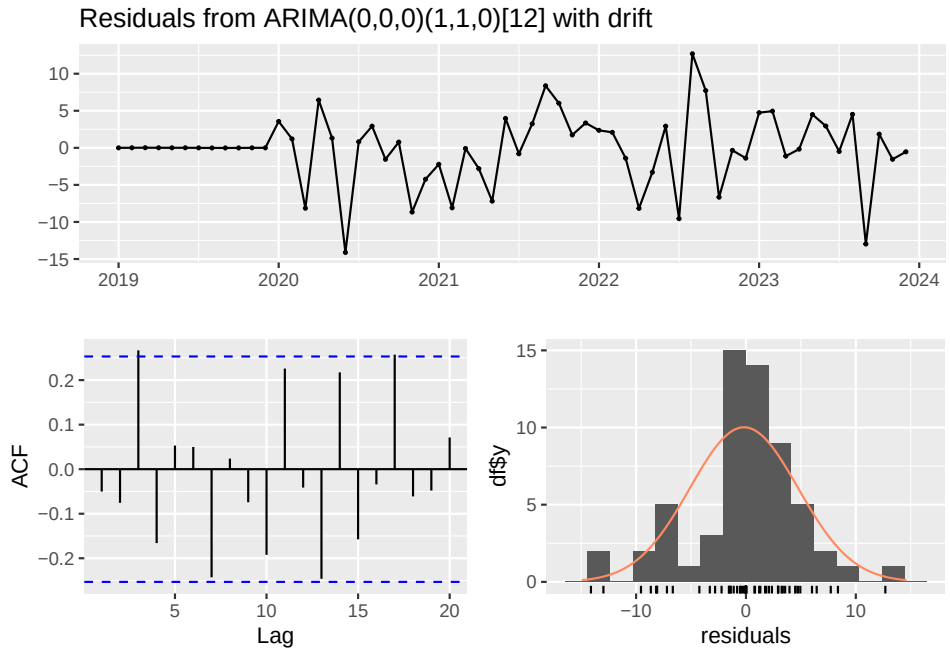
16.4.1.5 auto.arima

```
tsfit<-auto.arima(sim_ts)
summary(tsfit)
```

```
## Series: sim_ts
## ARIMA(0,0,0)(1,1,0)[12] with drift
##
## Coefficients:
##          sar1   drift
##       -0.6482  0.5105
## s.e.   0.1053  0.0445
##
## sigma^2 = 31.18:  log likelihood = -152.91
```

```
## AIC=311.82   AICc=312.37   BIC=317.43
##
## Training set error measures:
##                      ME     RMSE      MAE      MPE     MAPE      MASE         ACF1
## Training set -0.1763034 4.889015 3.344342 12.61809 63.78359 0.4208828 -0.05025177
```

```
#check residuals to assess model quality
checkresiduals(tsfit)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,0)(1,1,0)[12] with drift
## Q* = 18.699, df = 11, p-value = 0.06671
```

```
##
## Model df: 1.   Total lags used: 12
```

The code automatically selects the best ARIMA(p,d,q) model. It checks for stationarity, determines how many differences are needed (d), tests different combinations of autoregressive (p) and moving average (q) terms.

Some statements to help you understand what the p,d,q are:

p: If volume was high last month, they might still be high this month

d: Did the volume change, like, what was the movement? An increase or decrease?

q: Lol, the volume was not what I expected. Let's correct that this time around!

So together, the ARIMA model (let's say ARIMA (1,1,1) is:

"I will predict today's volume based on last month's volume and the mistake I made in last month's prediction but I will do that after adjusting the data for trend."

Now, when it comes to the results, the best model is ARIMA(0,0,0)(1,1,0)[12] with drift. This means that the model doesn't rely on lagged values (0,0,0) and the model relies on the previous year's value (AR term at lag 12). The model captures a linear trend over time too.

As for the plots, I will only summarize what you need to see:

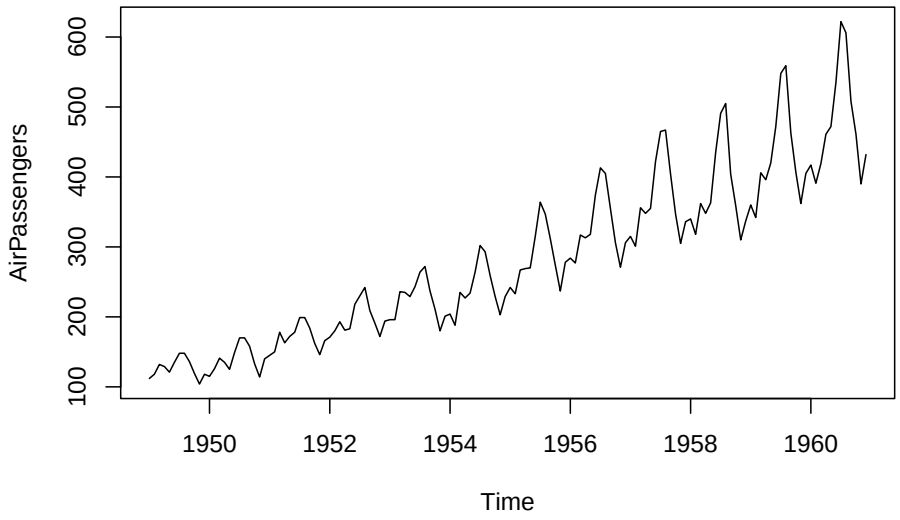
- Top: Shows the residuals which should fluctuate randomly around zero. There shouldn't be a pattern.

- Left: ACF of Residuals where what you want is that all bars are within the blue dashed lines (95% confidence interval)
- Right: Histogram of Residuals where it shows the distribution of errors and you want the red curve (normal fit) to overlap on the histogram nicely.

16.4.2 Real Data

We use a dataset from Base R - monthly airline passengers from 1949-1960.

```
data("AirPassengers")
plot(AirPassengers)
```



```
datats2 <- tibble(
  time = as.yearmon(time(AirPassengers)),      # Convert time index to year-month
  passengers = as.numeric(AirPassengers),      # Convert time series to numeric
  lag1 = lag(passengers, 1),                   # Add Lag 1: previous month
  lag12 = lag(passengers, 12)                  # Add Lag 12: same month last year
)

# View the first few rows
head(datats2, 15)
```

```
## # A tibble: 15 x 4
##    time      passengers  lag1 lag12
##    <yearmon>      <dbl> <dbl> <dbl>
##  1 Jan 1949         112    NA    NA
##  2 Feb 1949         118   112    NA
##  3 Mar 1949         132   118    NA
##  4 Apr 1949         129   132    NA
##  5 May 1949         121   129    NA
##  6 Jun 1949         135   121    NA
##  7 Jul 1949         148   135    NA
##  8 Aug 1949         148   148    NA
##  9 Sept 1949        136   148    NA
## 10 Oct 1949         119   136    NA
## 11 Nov 1949         104   119    NA
## 12 Dec 1949         118   104    NA
## 13 Jan 1950         115   118   112
## 14 Feb 1950         126   115   118
```

```
## 15 Mar 1950          141    126    132
```

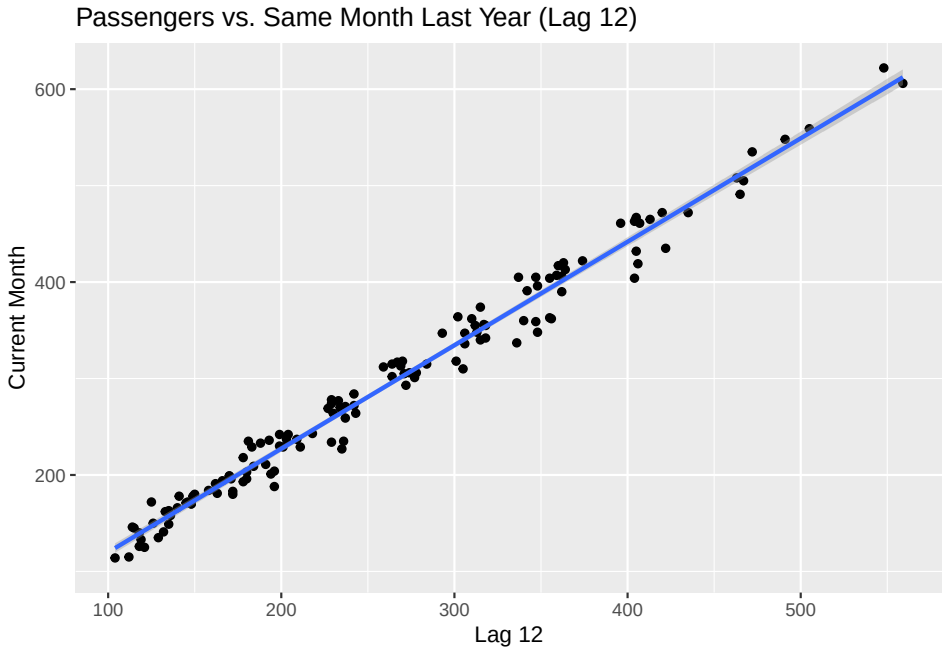
We don't have data for `lag12`'s first 12 rows.

```
ggplot(datats2, aes(x = lag12, y = passengers)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "Passengers vs. Same Month Last Year (Lag 12)",
       x = "Lag 12", y = "Current Month")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

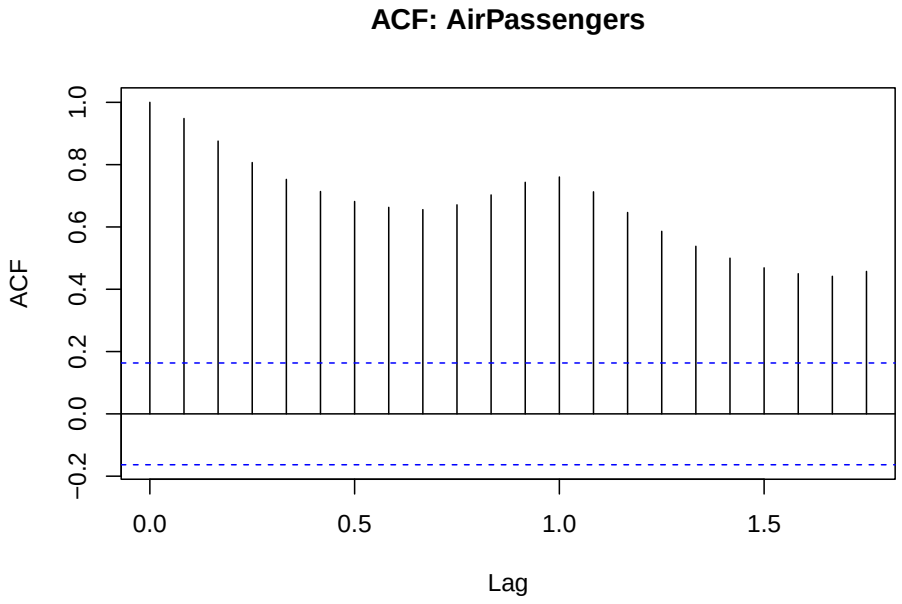
```
## Warning: Removed 12 rows containing non-finite outside the scale range (`stat_smooth()
```

```
## Warning: Removed 12 rows containing missing values or values outside the scale range (
```

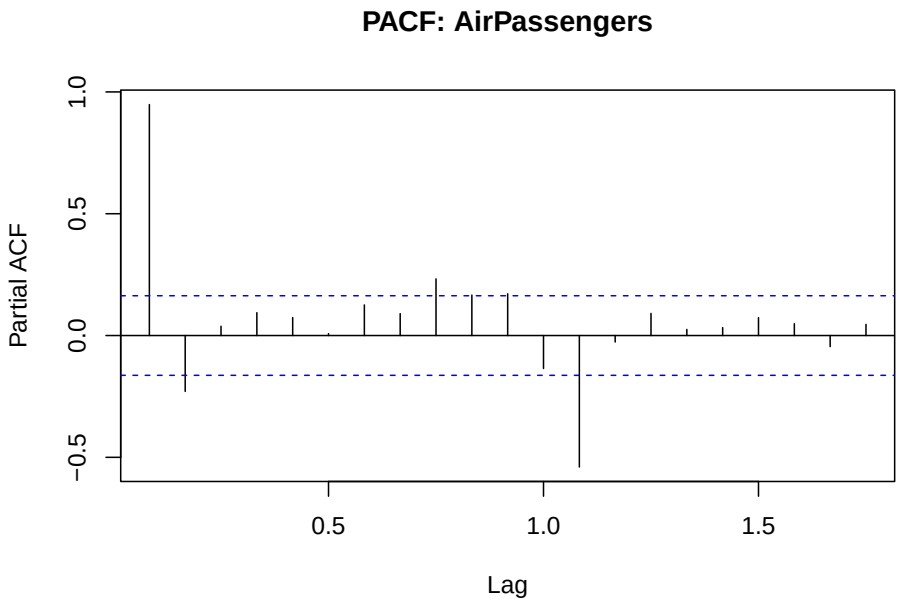


16.4.2.1 ACF and PACF

```
acf(AirPassengers, main = "ACF: AirPassengers")
```




```
pacf(AirPassengers, main = "PACF: AirPassengers")
```



We use the dataset, not the one where we made the tibble. so, we see that there is high autocorrelation at many lags and no sudden cutoff. We also see a spike at lag 1 in the PACF.

16.4.2.2 Stationarity

```
adf.test(AirPassengers)
```

```
## Warning in adf.test(AirPassengers): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  AirPassengers
```

```
## Dickey-Fuller = -7.3186, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

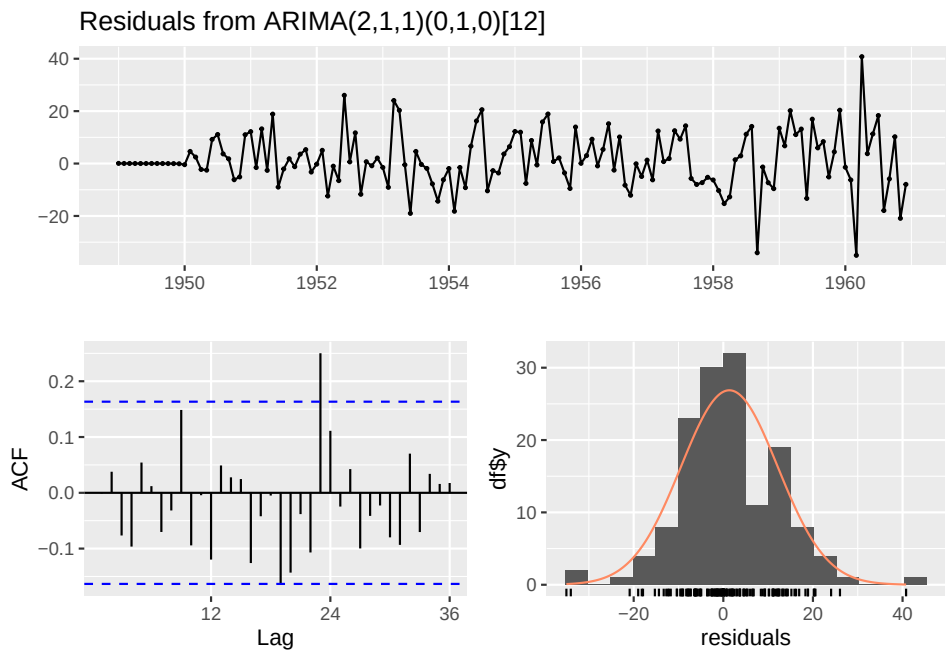
Though the ACF/PACF suggest that it's non-stationary, the ADF might return stationarity due to seasonal effects. So, let us confirm what needs to be done through the auto.arima.

16.4.2.3 auto.arima

```
tsfit2<-auto.arima(AirPassengers)
summary(tsfit2)
```

```
## Series: AirPassengers
## ARIMA(2,1,1)(0,1,0)[12]
##
## Coefficients:
##          ar1     ar2      ma1
##       0.5960  0.2143  -0.9819
## s.e.  0.0888  0.0880   0.0292
##
## sigma^2 = 132.3:  log likelihood = -504.92
## AIC=1017.85   AICc=1018.17   BIC=1029.35
##
## Training set error measures:
##                     ME     RMSE     MAE      MPE     MAPE     MASE         ACF1
## Training set 1.3423 10.84619 7.86754 0.420698 2.800458 0.245628 -0.00124847
```

```
#check residuals to assess model quality

checkresiduals(tsfit2)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,1)(0,1,0)[12]
## Q* = 37.784, df = 21, p-value = 0.01366
##
## Model df: 3.   Total lags used: 24
```

We see that seasonal differencing is employed since the best model is ARIMA(2,1,1)(0,1,0)[12]. The trend and seasonality were present (2,1,1)(0,1,0) and were addressed through differencing.

The results show that not all bars are within the blue lines so we need to check or adjust lag terms (especially the lag that is not within the bounds).

16.5 The End!