



Data Analysis

By

Jay Chaturbhai Kakadiya
Student Id: 2230044

Submitted to the University of Wales Trinity Saint David

Module: Advanced Statistical Programming
Lecturer: Dr. Nik Whitehead

Word Count: 2535

Abstract

This document presents an R script, named '**climatewales.r**', designed to process and analyse climate data from the UK Meteorological Office contained in the Excel file '**WalesClimateData.xlsx**'. The dataset encompasses information on maximum and minimum temperatures (in °C), hours of sunshine, millimetres of rainfall, and the count of days with rainfall ≥ 1 mm, spanning the entire duration of available records. The data is organized monthly, with mean values provided for each season and the entire year.

The primary objective of the script is to facilitate the generation of insights to address specific questions outlined in a subsequent report. These questions focus on visualizing the evolution of average seasonal maximum temperatures over a substantial period (1892 to 2022), assessing the correlation between total annual rainfall and annual maximum temperature, conducting a t-test to discern differences in minimum temperatures during distinct periods, and employing ANOVA to identify statistically significant variations in the annual count of wet days across decades from 1891 to 2020.

Table of Contents

1.	Introduction.....	5
2.	Import Packages and Read Data	6
3.	Data Preprocessing.....	7
3.1	‘handle_missing_data’ Function	7
3.2	Main Script	7
4	Average Seasonal Maximum Temperatures (1892-2022): A Decadal Perspective.....	8
5	Correlation Between Annual Rainfall and Maximum Temperature	11
6	Shapiro test and t-test	13
7	ANOVA Test and post-hoc tests (Tukey's HSD)	16
7.1	Analysis	18
8	Reflection	20
8.1	Challenges	20
8.2	Learning	20
8.3	Improvement for Next Time.....	20
9	Conclusion	21
10	References	22
11	Appendix.....	23

Table of Figures

Figure 1: Load the Packages and Read Data.....	6
Figure 2: Data Preprocessing.....	7
Figure 3: Calculate Average Seasonal Maximum Temperature	8
Figure 4: Extract Every tenth Year	9
Figure 5: Plot the Multiple Bar Graph	9
Figure 6: Task 1 Main Script	10
Figure 7: Average Seasonal Maximum Temperature per Decade (1892 - 2022).....	10
Figure 8: Correlation and Scatterplot.....	11
Figure 9: Annual Maximum Temperature vs Annual Rainfall	12
Figure 10: Shapiro test and t-test	13
Figure 11: Main Script of t-test.....	13
Figure 12: Boxplot of period 1890-1910	14
Figure 13: Boxplot of period 2000-2020	14
Figure 14: Results of Shapiro Test and t-test.....	15
Figure 15: ANOVA and Post-hoc tests.....	16
Figure 16: Summary of ANOVA Test	17
Figure 17: Result of Post-hoc test_1	17
Figure 18: Result of Post-hoc test_2	18
Figure 19: Boxplot of Annual numbers of Wet Days by Decades	19

1. Introduction

Understanding historical climate patterns is crucial for discerning potential trends and anomalies that may have far-reaching implications. The dataset under consideration, extracted from **‘WalesClimateData.xlsx’**, provides a comprehensive set of variables, enabling an in-depth analysis of key climatic aspects. To unravel the nuances within this dataset, an R script named **‘climatewales.r’** has been developed, aligning with the data analysis tasks laid out for this project.

In the subsequent sections of this document, we delve into the intricacies of the R script, elucidating the methodology employed to handle data processing and analysis. The tasks at hand include visualizing changes in average seasonal maximum temperatures over an extended period, exploring the correlation between total annual rainfall and annual maximum temperature, investigating potential differences in minimum temperatures during distinct time intervals, and conducting a comprehensive analysis of the annual count of wet days across complete decades.

The outcomes of these analyses are pivotal for extracting meaningful insights into climate trends, contributing to a broader understanding of the climatic conditions observed in Wales over the past century. Through the ensuing sections, the document details the structure and functionality of the R script, providing a comprehensive foundation for subsequent reporting on the specific questions posed for analysis.

Note: In the R script, all the functions and the main script are separated, but in this document, for better visualisation and understanding, the function and main script of that particular function are combined.

2. Import Packages and Read Data

The first step of this task is to install the relevant libraries if it is not already installed and read the data from the Excel file. Figure 1 shows the R script of it.

Figure 1: Load the Packages and Read Data

```
# Install the readxl, dplyr, tidyr, ggplot2 packages if not already installed
# Load the packages
library(readxl)
library(dplyr)
library(tidyr)
library(ggplot2)

# Read data from Excel sheet
read_excel_sheet <- function(file, sheet_name) {
  readxl::read_excel(file, sheet = sheet_name)
}

# *****----- Main Script -----*****
# Store the file name
walesclimate <- "Wales Climate Data.xlsx"
# Get data from different sheets
max_temp_data <- read_excel_sheet(walesclimate, "Max Temp")
min_temp_data <- read_excel_sheet(walesclimate, "Min Temp")
sunshine_data <- read_excel_sheet(walesclimate, "Sunshine")
rainfall_mm_data <- read_excel_sheet(walesclimate, "Rainfall")
rainfall_days_data <- read_excel_sheet(walesclimate, "Rain Days")
```

Code Explanation

The provided R code initiates the process of analyzing climate data from an Excel file named "Wales Climate Data.xlsx." The code first ensures that essential packages, including **readxl**, **dplyr**, **tidyr**, and **ggplot2**, are installed.[1] These packages are crucial for data manipulation, transformation, and visualization.

The core functionality of the script lies in reading data from various sheets within the Excel file. A custom function named **read_excel_sheet** is defined to streamline this process. This function takes two parameters - the file name (**file**) and the sheet name (**sheet_name**). It utilizes the **read_excel** function from the **readxl** package[1] to read the specified sheet from the Excel file.

The script then proceeds with the main data acquisition phase. The file name, "Wales Climate Data.xlsx," is stored in a variable named **walesclimate**. Subsequently, data is fetched from different sheets within the Excel file, each corresponding to specific climate metrics. The data is read into separate data frames: **max_temp_data** for maximum temperature, **min_temp_data** for minimum temperature, **sunshine_data** for hours of sunshine, **rainfall_mm_data** for millimeters of rainfall, and **rainfall_days_data** for the count of days with rainfall ≥ 1 mm.

3. Data Preprocessing

Data preprocessing is a crucial step in the data analysis pipeline that involves cleaning and transforming raw data into a format suitable for analysis. This phase is essential for ensuring the accuracy, reliability, and effectiveness of subsequent data analyses.

The data preprocessing steps we used are depicted in Figure 2.

Figure 2: Data Preprocessing

```
# ----- Data Pre-Processing Function -----  
# Handle missing data  
handle_missing_data <- function(data) {  
  data[data == '---'] <- NA # Replace '---' to NAs  
  data <- as.numeric(data) # Convert character to Number  
  data[is.na(data)] <- mean(data, na.rm = TRUE) # Replace NAs with Mean  
  data <- format(round(as.numeric(data), 2), nsmall = 2) # Round the Data  
  data <- as.numeric(data) # Convert Character to Number  
  return(data)  
}  
  
# *****----- Main Script -----*****  
# Data Pre-Processing  
max_temp_data$win <- handle_missing_data(max_temp_data$win)  
min_temp_data$win <- handle_missing_data(min_temp_data$win)  
sunshine_data$win <- handle_missing_data(sunshine_data$win)  
rainfall_mm_data$win <- handle_missing_data(rainfall_mm_data$win)  
rainfall_days_data$win <- handle_missing_data(rainfall_days_data$win)
```

Code Explanation

3.1 'handle_missing_data' Function

Real-world datasets often contain missing values, which can adversely impact the analysis. The **handle_missing_data** function in the provided code addresses missing data by replacing specific placeholder values ('---') with NA (Not Available).

Inconsistent data types within a dataset can lead to errors during analysis. The code converts character data to numeric format, ensuring uniformity and facilitating numerical computations.
[2]

Subsequently, missing numeric values are imputed with the mean of the corresponding column.
[2] This ensures that missing data does not hinder the overall analysis, and the imputation method is chosen judiciously to minimize bias.

Precision in data is essential for maintaining accuracy and readability. The code rounds numeric values to two decimal places using the **round** function.[2] This step is particularly relevant when dealing with measurements or values that don't require excessive precision. Then we convert the data into numeric form and return the pre-processed data.

3.2 Main Script

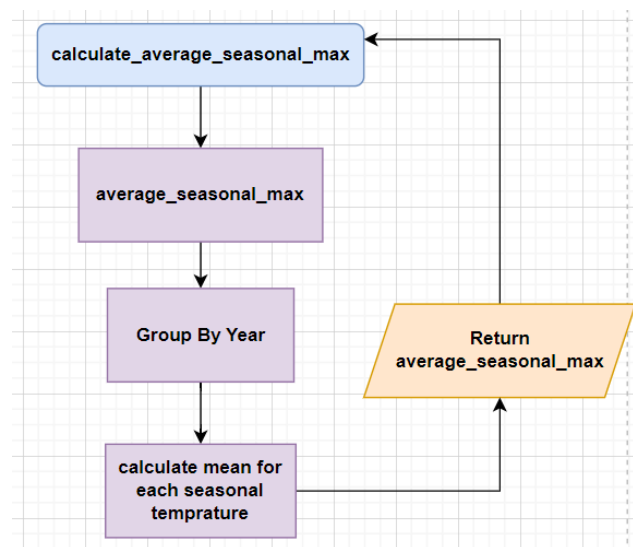
The **handle_missing_data** function is applied to specific columns (**win**) in different climate-related data frames (**max_temp_data**, **min_temp_data**, **sunshine_data**, **rainfall_mm_data**, **rainfall_days_data**).

4 Average Seasonal Maximum Temperatures (1892-2022): A Decadal Perspective

This task 1 includes 3 functions. The first function calculates the average seasonal maximum temperature, second function extract every tenth year for plotting and the finale function is for creating and display multiple bar graph.

Figure 3 shows the flowchart of first function, which is calculate the average seasonal maximum temperature.

Figure 3: Calculate Average Seasonal Maximum Temperature



The function uses the %>% (pipe) operator from the dplyr package to create a sequence of data manipulations. It groups the input data by the 'year' column using the **group_by** function. It then uses the **summarize** function to calculate the mean of the seasonal temperature columns (spr, sum, aut, win) for each year. The result is a data frame (**average_seasonal_max**) where each row represents a unique year, and columns represent the average seasonal maximum temperatures for spring (spr), summer (sum), autumn (aut), and winter (win). Then, The function returns the summarized data frame.

Figure 4 shows the flowchart of the second function which extract every tenth year for plotting. As we can see in Figure 4, The function extracts every tenth row from the input data frame, effectively selecting every tenth year. The selection is based on the row indices using the **seq** function with a **step size of 10, starting from the 9th row** (from 1892 onwards) to avoid extracting the first row (from 1884). The function returns the data frame containing every tenth year.

Figure 4: Extract Every tenth Year

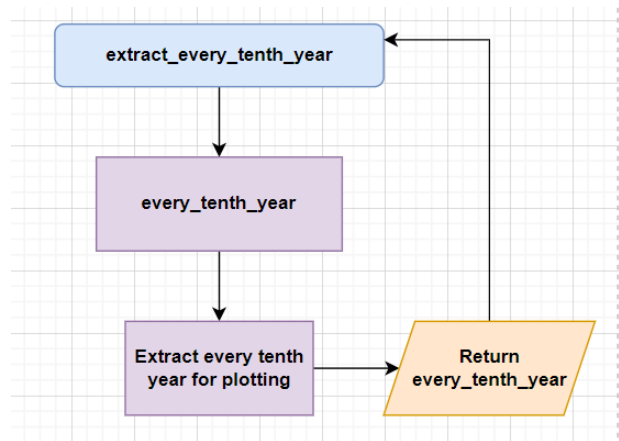
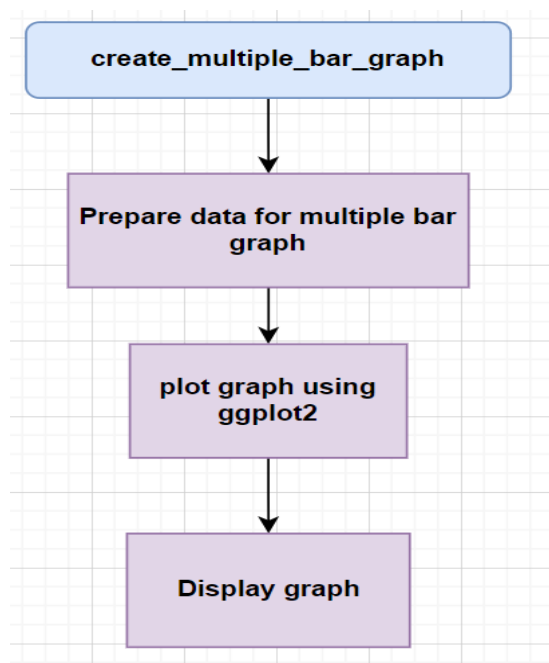


Figure 5 shows the flowchart of third function, which create a multiple bar graph.

Figure 5: Plot the Multiple Bar Graph



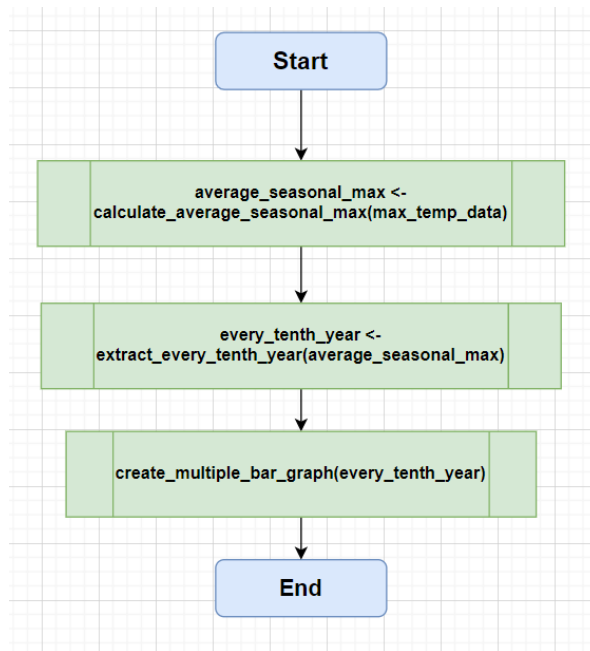
The function uses the gather function from the **tidyr** package to reshape the input data frame into a long format suitable for plotting[1]. It then uses **ggplot2** to create a multiple bar graph (**multiple_bar_plot**) with the reshaped data[1].

The function prints and displays the created multiple bar graph.

Note: This function does not explicitly return an object as it is designed for visualization rather than data manipulation.

Flowchart of the main script of this section is shown in Figure 6.

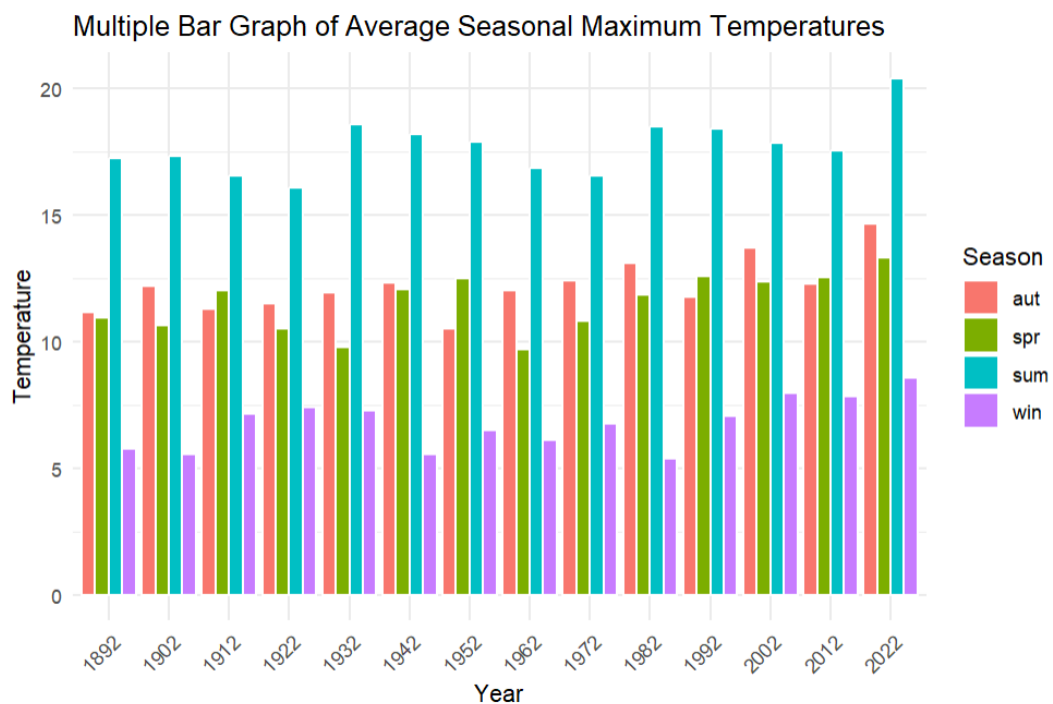
Figure 6: Task 1 Main Script



In the main script, the average seasonal maximum temperatures are computed from the **max_temp_data**, resulting in the **average_seasonal_max** data frame. Subsequently, every tenth year is extracted from this dataset, and a multiple bar graph is created using **ggplot2** to visually represent the trends in seasonal temperatures over time.

Finale output of this section can be shown in Figure 7. The x-axis represents the year, the y-axis represents the temperature, and different bars represent different seasons.

Figure 7: Average Seasonal Maximum Temperature per Decade (1892 - 2022)

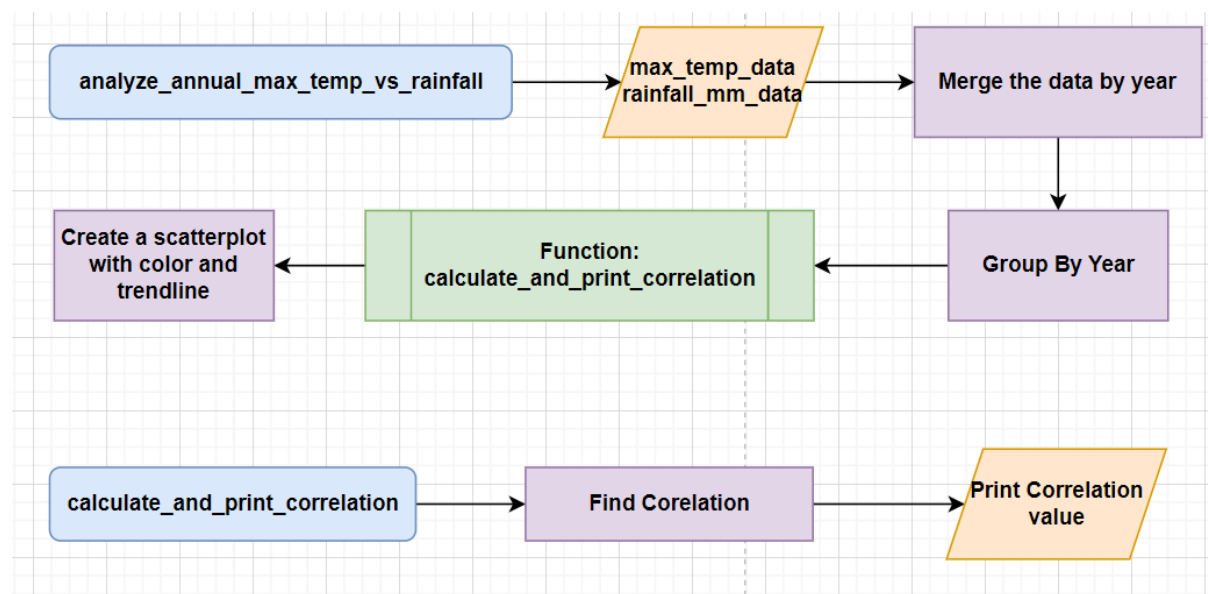


5 Correlation Between Annual Rainfall and Maximum Temperature

In this task, the script computes the correlation between the total annual rainfall and the annual maximum temperature by making a plot.

Figure 8 shows the flowchart of 2 functions, which calculate the correlation and create a scatterplot with support line.

Figure 8: Correlation and Scatterplot



The **max_temp_data** and **rainfall_mm_data** - two data frames containing information on annual maximum temperature and annual rainfall, respectively. Firstly, function merges the two data frames based on the 'year' column, including only the years where both datasets have complete observations (`all = FALSE`). Secondly, it Calls another function **calculate_and_print_correlation** to calculate and print the correlation coefficient between the annual maximum temperature and annual rainfall. Lastly, it creates scatterplot with color and trendline using **plot** function.

In main script by writing below script, we can run the code and see the correlation value:

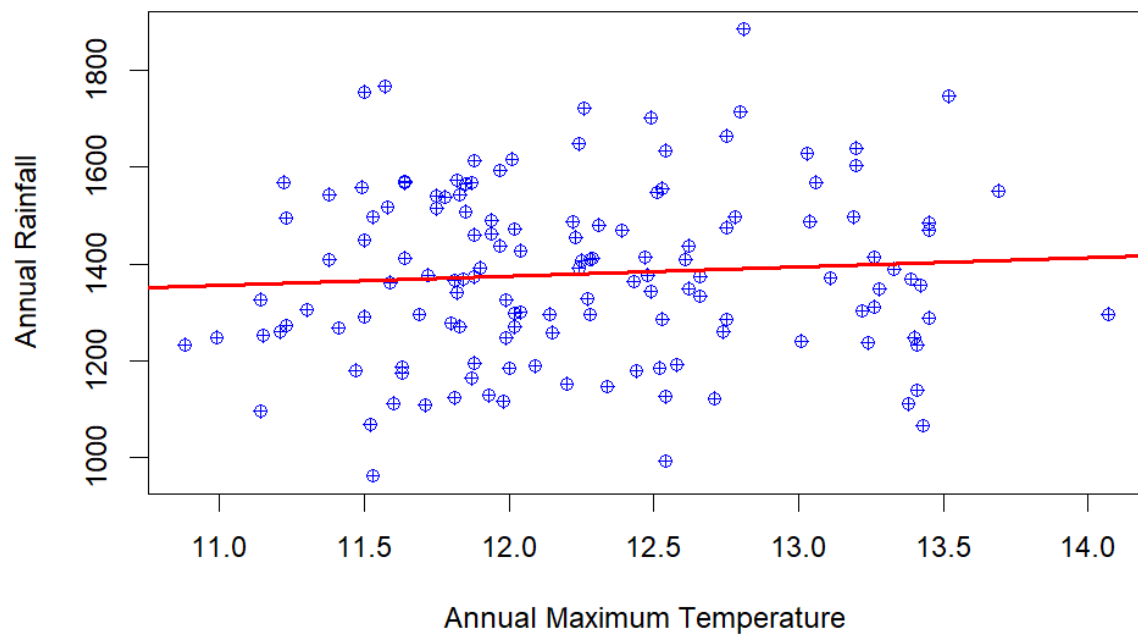
```
analyze_annual_max_temp_vs_rainfall(max_temp_data, rainfall_mm_data)
```

The correlation coefficient value of annual maximum temperature and annual rainfall is **0.0725**.

Figure 9 shows the scatterplot with a trendline.

Figure 9: Annual Maximum Temperature vs Annual Rainfall

Scatterplot of Annual Maximum Temperature vs. Annual Rainfall



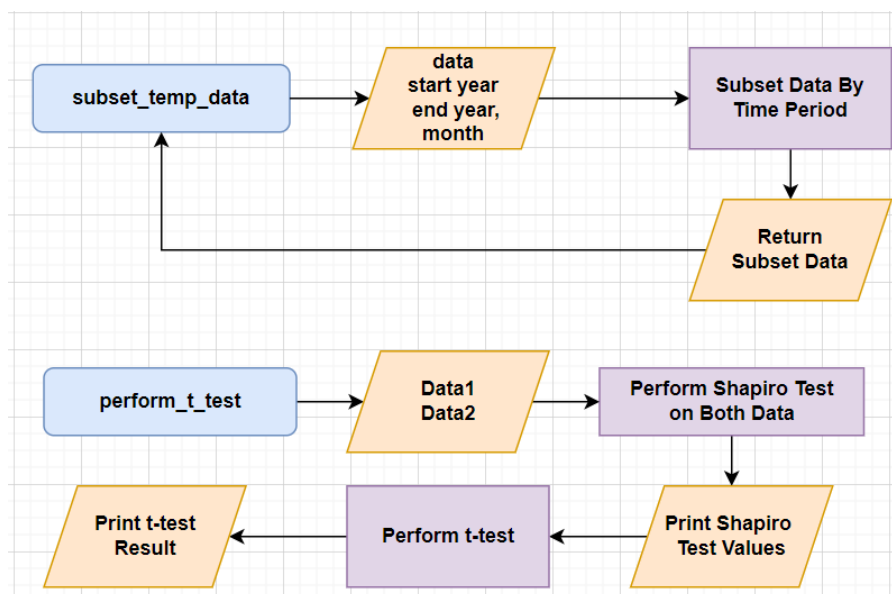
6 Shapiro test and t-test

A t-test is a statistical method used to compare the means of two groups and determine if there is a significant difference between them. It's particularly useful when dealing with small sample sizes.

In this task we perform t-test to identify whether there is a significant difference in minimum temperatures between August 1890-1910 and August 2000-2020.

Figure 10 shows the flowchart of the functions we used to perform Shapiro test and t-test.

Figure 10: Shapiro test and t-test



subset_temp_data function takes input data, start year, end year and month. It Subsets the input data based on the specified time period (from start_year to end_year inclusive) and returns the subset of data corresponding to the specified month.

In **perform_t_test** function, input 2 data and perform Shapiro tests on both the data. Then we print the p-value of both data. After that we conduct the t-test on both data and prints the result of the t-test. [3]

The main body of this task is shown in Figure 11.

Figure 11: Main Script of t-test

```
# Subset the data for the two time periods
min_temp_1890_1910 <- subset_temp_data(min_temp_data, 1890, 1910, "aug")
min_temp_2000_2020 <- subset_temp_data(min_temp_data, 2000, 2020, "aug")

# Boxplot for minimum temperature of August between 1890 and 1910
boxplot(min_temp_1890_1910, main = "Minimum Temperature of August between 1890 and 1910")
# No Outliers. No Need To apply filter on data

# Boxplot for minimum temperature of August between 2000 and 2020
boxplot(min_temp_2000_2020, main = "Minimum Temperature of August between 2000 and 2020")
# No Outliers. No Need To apply filter on data
# Perform Shapiro test and t-test
perform_t_test(min_temp_1890_1910, min_temp_2000_2020)
```

The code uses the `subset_temp_data` function to extract minimum temperature data specifically for August in the years 1890-1910 and 2000-2020.

Two boxplots are generated to visually compare the minimum temperatures for August in the specified time periods. The first boxplot represents the period from 1890 to 1910, and the second boxplot represents the period from 2000 to 2020.

The code concludes by calling the `perform_t_test` function, which conducts Shapiro tests to assess the normality of the data and performs a t-test to determine if there is a significant difference in minimum temperatures between the two specified time periods.

Figure 12 and Figure 13 show the boxplots.

Figure 12: Boxplot of period 1890-1910

Minimum Temperature of August between 1890 and 1910

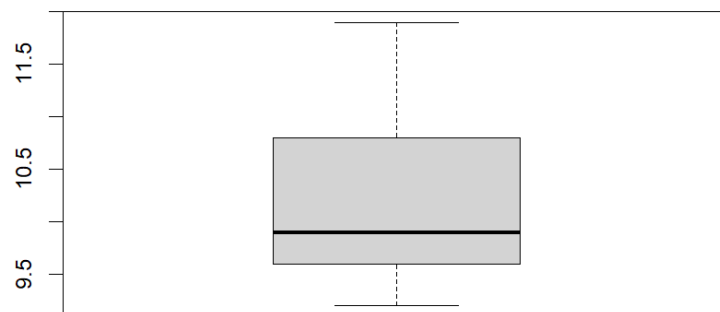
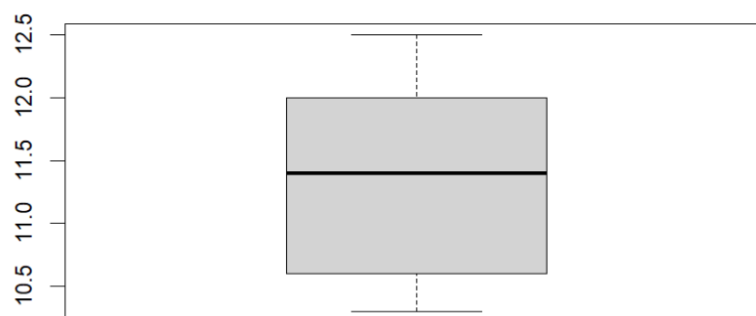


Figure 13: Boxplot of period 2000-2020

Minimum Temperature of August between 2000 and 2020



As we can see in both boxplots, there is no outliers are present in either dataset, suggesting that further data filtering is unnecessary.

Figure 14 shows the result of Shapiro test and t-test.

The p-value for the Shapiro test on the data sample from 1890-1910 is greater than the common significance level of 0.05 which is 0.05. [3] Therefore, we do not have enough evidence to reject the null hypothesis of normality. The data appears to be reasonably normally distributed.

Figure 14: Results of Shapiro Test and t-test

```
[1] "Shapiro Test p-value (1890-1910): 0.0522694417128314"
[1] "Shapiro Test p-value (2000-2020): 0.136542565918646"

Welch Two Sample t-test

data: data1 and data2
t = -4.6707, df = 39.819, p-value = 3.388e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.5555841 -0.6158445
sample estimates:
mean of x mean of y
 10.24286  11.32857
```

Similarly, the p-value for the Shapiro test on the data sample from 2000-2020 is greater than 0.05. This indicates that, based on the Shapiro test, we do not have enough evidence to reject the null hypothesis of normality for this dataset.

The p-value for the Welch Two Sample t-test is very small (p-value = 3.388e-05), well below the common significance level of 0.05. [3] Therefore, we reject the null hypothesis that there is no difference in the means of the two groups. The small p-value suggests that there is a significant difference in minimum temperatures between the two time periods (1890-1910 and 2000-2020). [3]

7 ANOVA Test and post-hoc tests (Tukey's HSD)

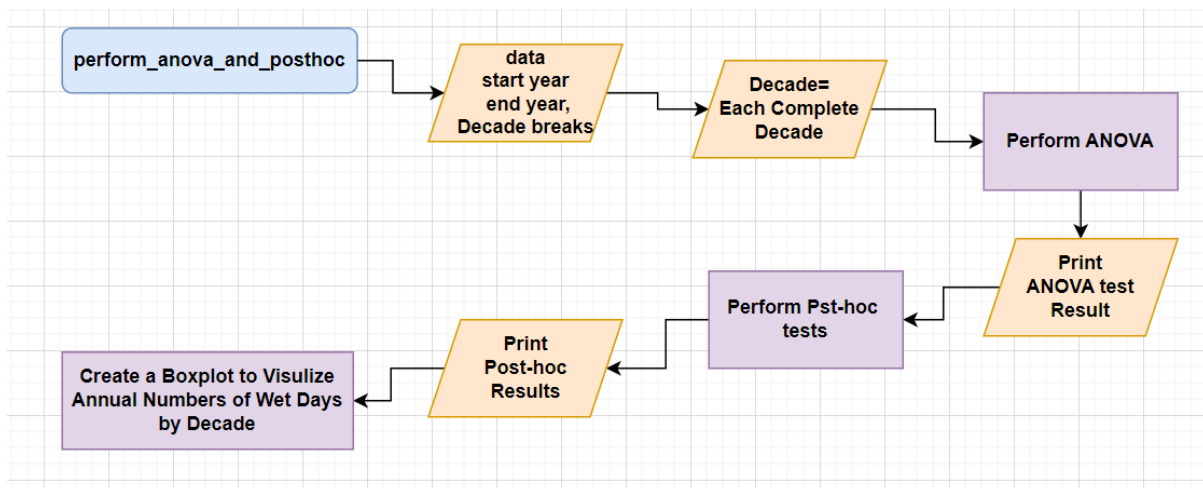
To find out if there are statistically significant differences in the annual numbers of wet days in each whole decade beginning in 1891 and ending in 2020, we conduct an ANOVA test in this task.

ANOVA (Analysis of Variance) is a statistical method used to analyze the differences among group means in a sample. [3] It assesses whether there are any statistically significant differences between the means of three or more independent groups.

Post-hoc tests, such as Tukey's Honestly Significant Difference (HSD), are performed after an ANOVA when the null hypothesis is rejected, indicating that there are significant differences between groups.

The following Figure 15 shows the flowchart of the function we used to perform ANOVA and post-hoc tests.

Figure 15: ANOVA and Post-hoc tests



The function takes inputs: **data** (the dataset), **start_year**, **end_year**, and **decade_breaks** (indicating breaks for each complete decade).

A new variable 'Decade' is created by cutting the 'year' variable in the dataset into discrete intervals based on the specified decade breaks. This variable represents each complete decade.

An ANOVA (Analysis of Variance) test is performed using the **aoV** function. The dependent variable is '**ann**' (annual numbers of wet days), and the factor is the newly created '**Decade**' variable and print the summary of ANOVA test.

To identify where the differences lie, post-hoc tests (specifically, Tukey's HSD) are performed on the ANOVA results using the **TukeyHSD** function and print the result of this tests.

Finally, a box plot is created to visually represent the annual numbers of wet days by decade. This plot provides a graphical summary of the distribution of the data, highlighting potential variations between decades.

Below R script is used to call this function and complete this task:

```
decade_breaks <- seq(1890, 2020, by = 10)
perform_anova_and_posthoc(rainfall_days_data, 1891, 2020, decade_breaks)
```

Figure 16 shows the summary of ANOVA test.

Figure 16: Summary of ANOVA Test

	Df	Sum Sq	Mean Sq	F value	Pr(>F)						
as.factor(Decade)	12	4571	380.9	1.632	0.0917						
Residuals	117	27298	233.3								

Signif. codes:	0	'***'	0.001	'**'	0.01	'*'	0.05	'.'	0.1	' '	1

The p-value of ANOVA test is 0.0917, suggests that there is a marginally significant difference in the mean annual numbers of wet days across the different decades.

The result of post-hoc test is shown in Figures 17 and 18.

Figure 17: Result of Post-hoc test_1

	diff	lwr	upr	p adj
2-1	8.64	-14.465216	31.745216	0.9878973
3-1	8.61	-14.495216	31.715216	0.9882490
4-1	13.00	-10.105216	36.105216	0.7902320
5-1	1.83	-21.275216	24.935216	1.0000000
6-1	-0.48	-23.585216	22.625216	1.0000000
7-1	1.64	-21.465216	24.745216	1.0000000
8-1	-2.66	-25.765216	20.445216	1.0000000
9-1	-8.29	-31.395216	14.815216	0.9915163
10-1	-4.81	-27.915216	18.295216	0.9999614
11-1	0.91	-22.195216	24.015216	1.0000000
12-1	-4.13	-27.235216	18.975216	0.9999927
13-1	7.18	-15.925216	30.285216	0.9977055
3-2	-0.03	-23.135216	23.075216	1.0000000
4-2	4.36	-18.745216	27.465216	0.9999868
5-2	-6.81	-29.915216	16.295216	0.9986172
6-2	-9.12	-32.225216	13.985216	0.9810434
7-2	-7.00	-30.105216	16.105216	0.9981977
8-2	-11.30	-34.405216	11.805216	0.9082776
9-2	-16.93	-40.035216	6.175216	0.3985863
10-2	-13.45	-36.555216	9.655216	0.7504497
11-2	-7.73	-30.835216	15.375216	0.9954495
12-2	-12.77	-35.875216	10.335216	0.8093308
13-2	-1.46	-24.565216	21.645216	1.0000000
4-3	4.39	-18.715216	27.495216	0.9999857
5-3	-6.78	-29.885216	16.325216	0.9986751
6-3	-9.09	-32.195216	14.015216	0.9815449
7-3	-6.97	-30.075216	16.135216	0.9982704
8-3	-11.27	-34.375216	11.835216	0.9098560
9-3	-16.90	-40.005216	6.205216	0.4014807
10-3	-13.42	-36.525216	9.685216	0.7531939
11-3	-7.70	-30.805216	15.405216	0.9956077
12-3	-12.74	-35.845216	10.365216	0.8117557
13-3	-1.43	-24.535216	21.675216	1.0000000
5-4	-11.17	-34.275216	11.935216	0.9149889
6-4	-13.48	-36.585216	9.625216	0.7476932
7-4	-11.36	-34.465216	11.745216	0.9050675
8-4	-15.66	-38.765216	7.445216	0.5270823
9-4	-21.29	-44.395216	1.815216	0.1029212
10-4	-17.81	-40.915216	5.295216	0.3180896
11-4	-12.09	-35.195216	11.015216	0.8602873
12-4	-17.13	-40.235216	5.975216	0.3795226
13-4	-5.82	-28.925216	17.285216	0.9997122

Figure 18: Result of Post-hoc test_2

6-5	-2.31	-25.415216	20.795216	1.0000000
7-5	-0.19	-23.295216	22.915216	1.0000000
8-5	-4.49	-27.595216	18.615216	0.9999818
9-5	-10.12	-33.225216	12.985216	0.9574575
10-5	-6.64	-29.745216	16.465216	0.9989190
11-5	-0.92	-24.025216	22.185216	1.0000000
12-5	-5.96	-29.065216	17.145216	0.9996329
13-5	5.35	-17.755216	28.455216	0.9998801
7-6	2.12	-20.985216	25.225216	1.0000000
8-6	-2.18	-25.285216	20.925216	1.0000000
9-6	-7.81	-30.915216	15.295216	0.9950042
10-6	-4.33	-27.435216	18.775216	0.9999878
11-6	1.39	-21.715216	24.495216	1.0000000
12-6	-3.65	-26.755216	19.455216	0.9999982
13-6	7.66	-15.445216	30.765216	0.9958116
8-7	-4.30	-27.405216	18.805216	0.9999887
9-7	-9.93	-33.035216	13.175216	0.9630713
10-7	-6.45	-29.555216	16.655216	0.9991878
11-7	-0.73	-23.835216	22.375216	1.0000000
12-7	-5.77	-28.875216	17.335216	0.9997366
13-7	5.54	-17.565216	28.645216	0.9998272
9-8	-5.63	-28.735216	17.475216	0.9997957
10-8	-2.15	-25.255216	20.955216	1.0000000
11-8	3.57	-19.535216	26.675216	0.9999986
12-8	-1.47	-24.575216	21.635216	1.0000000
13-8	9.84	-13.265216	32.945216	0.9655307
10-9	3.48	-19.625216	26.585216	0.9999989
11-9	9.20	-13.905216	32.305216	0.9796542
12-9	4.16	-18.945216	27.265216	0.9999921
13-9	15.47	-7.635216	38.575216	0.5469608
11-10	5.72	-17.385216	28.825216	0.9997592
12-10	0.68	-22.425216	23.785216	1.0000000
13-10	11.99	-11.115216	35.095216	0.8670436
12-11	-5.04	-28.145216	18.065216	0.9999363
13-11	6.27	-16.835216	29.375216	0.9993873
13-12	11.31	-11.795216	34.415216	0.9077476

7.1 Analysis

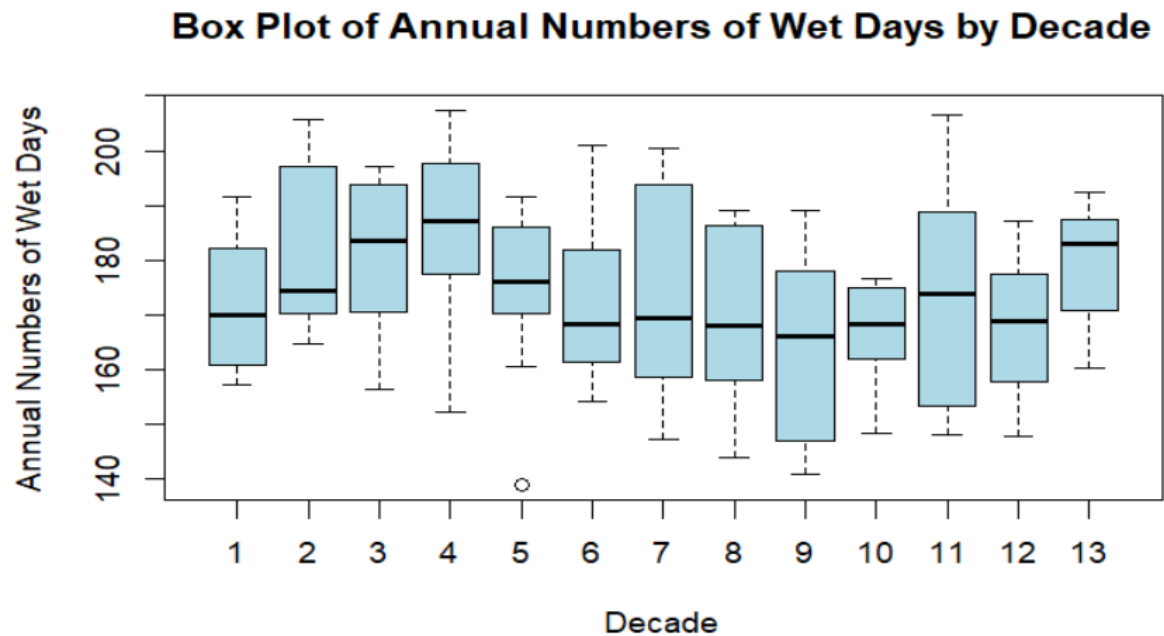
The '**p adj**' column provides adjusted p-values after considering multiple comparisons. If this value is less than the significance level (commonly 0.05), it suggests a significant difference between the corresponding groups.

- **Non-Significant Comparison:** Many of the comparisons show p-values close to 1.0, indicating a lack of significant differences between the corresponding decades. [3] For example, "2-1," "3-1," "4-1," and several others have high p-values.
- **Significant Comparison:** Some comparisons have lower p-values, suggesting significant differences. For instance, "9-4" has a p-value of 0.1029, indicating a borderline significance, while "8-4" has a p-value of 0.5270, suggesting no significant difference. [3]

This analysis provides insights into which pairs of decades exhibit significant differences in mean annual numbers of wet days.

Figure 19 shows the boxplot annual numbers of wet days by decades. This plot provides a graphical summary of the distribution of the data, highlighting potential variations between decades.

Figure 19: Boxplot of Annual numbers of Wet Days by Decades



8 Reflection

8.1 Challenges

Statistical Interpretation: Accurately interpreting statistical results, especially for complex analyses like ANOVA and post-hoc tests, requires a good grasp of statistical concepts. Misinterpretation can lead to incorrect conclusions.

8.2 Learning

Data Preprocessing Importance: The provided code emphasizes the significance of data preprocessing, including handling missing values and ensuring data types are appropriate. This step is crucial for the accuracy of subsequent analyses.

Statistical Testing: The tasks involve various statistical tests, such as correlation, t-tests, ANOVA, and post-hoc tests. Understanding the assumptions and implications of each test is crucial for drawing meaningful conclusions.

8.3 Improvement for Next Time

Interactive Analysis: Exploring the data interactively with tools like Jupyter Notebooks could be beneficial. It allows for a more dynamic exploration of the data and results.

9 Conclusion

In conclusion, this comprehensive data analysis, conducted through R programming, has provided valuable insights into the climate patterns in Wales. By employing various statistical tests and visualizations, we explored the changing trends in maximum temperatures, correlations between rainfall and temperature, differences in minimum temperatures across distinct time periods, and the variability in annual wet days. The meticulous data preprocessing and application of diverse statistical techniques contribute to the robustness of our findings.

10 References

- [1]R, “CRAN Packages By Name,” *R-project.org*, 2017.
https://cran.r-project.org/web/packages/available_packages_by_name.html
- [2]M. Yoosefzadeh Najafabadi, A. Heidari, and I. Rajcan, “AllInOne Pre-processing: A comprehensive preprocessing framework in plant field phenotyping,” *SoftwareX*, vol. 23, p. 101464, Jul. 2023, doi: <https://doi.org/10.1016/j.softx.2023.101464>.
- [3]A. Hazra and N. Gogtay, “Biostatistics series module 3: Comparing groups: Numerical variables,” *Indian Journal of Dermatology*, vol. 61, no. 3, p. 251, 2016, doi: 10.4103/0019-5154.182416.
<https://www.proquest.com/docview/1813897755?accountid=130472&parentSessionId=dK7Oojc448WERBEhdrFnua0GCkuEDv4mvApA2KBYkaw%3D&pq-origsite=primo&sourcetype=Scholarly%20Journals>.

11 Appendix

Flowchart Tool: <https://app.diagrams.net/>

Code Generator Tool: <https://chat.openai.com/>