



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Advanced Statistical Programming (ACCA7022 BIRM)

Student Name: Tasin Islam

Student Number: 2435184

Module Code: ACCA7022_BIRM

Module: Advanced Statistical Programming

Assignment Title: Data Analysis

Lecturer: Dr Aliyu Sambo

Word Count: 600

Submission Deadline: 14th November 2025

Port Talbot Hourly Air Quality Analysis

In this project, I have analysed the Port Talbot Hourly Air Quality dataset using R, focusing on the identification of trends, relationships, and statistical differences among the air pollutants. I wrote the whole code myself, line by line, based on my own understanding of data analysis in R. Throughout the work, the libraries applied were: tidyverse for cleaning and transforming data, lubridate for date time, rstatix for running statistical tests, ggplot2 for visualizations.

Approach and Rationale

First, I loaded all the necessary libraries, such as `library(rstatix)`, `library(readr)`, etc., to prepare the environment for data processing and statistical tests. Then, I read the dataset using `read_csv()` and handled missing data by setting `na = c("No data", "NA", "")` to avoid disrupting the analysis with non-numeric entries. Further, I changed all long headers into shorter, cleaner forms using `rename()` to make working with this dataset easier. For example, I renamed Nitrogen oxides as nitrogen dioxide to `NOx` and PM2.5 particulate matter (Hourly measured) to `PM2.5`.

I then combined the date and time into a single variable using `mutate()` and `dmy_hms()`. The creation of this new datetime column allowed me to order the dataset chronologically by using `arrange(datetime)`. This part was initially fiddly as the dates were not in one consistent format within the CSV file, but the issue was solved by setting `tz = "UTC"` explicitly for consistency.

Then, I computed the percent of missing data for each pollutant using a simple function inside `sapply()`. This provided an overview of data completeness.

Visualisations

Visualisations were used in communicating the trends and relationships clearly. I first created a line graph for `NOx` levels between July 2015 and November 2024 using the `displayPollutantGraph()` function: `displayPollutantGraph("NOx", 7, 2015, 11, 2024)`

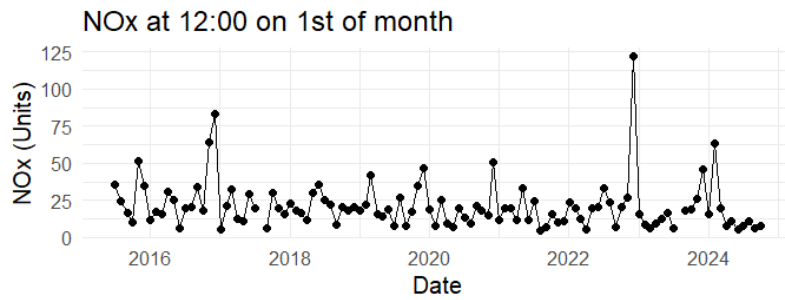


Fig1: Nitrogen Oxides as Nitrogen Dioxide(NOx) between July 2015 and November 2024

I chose line graphs because they show the change in pollutant levels over time, making temporal patterns visible. I also created a comparative plot of many pollutants-O3, NO, NO2, NOx-using ggplot() with color coding. This was helpful in identifying the relations and fluctuations among pollutants within the same timeframe.

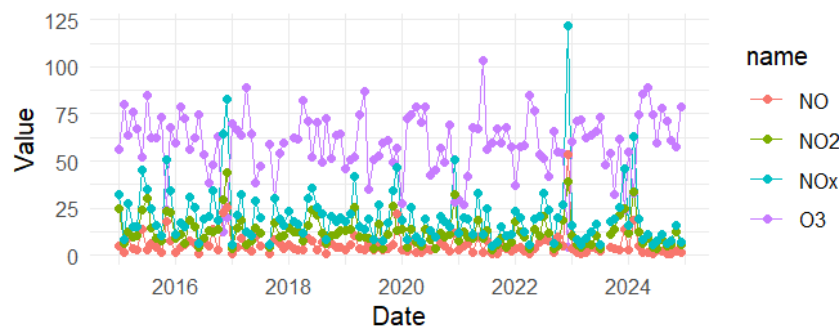


Fig2: Comparing vector of 4 pollutants

For correlation analysis between ozone and nitrogen dioxide, I created a scatter plot with a regression line using geom_point() and adding geom_smooth(method = "lm"). This visualization made it easier to interpret the strength and direction of the correlation.

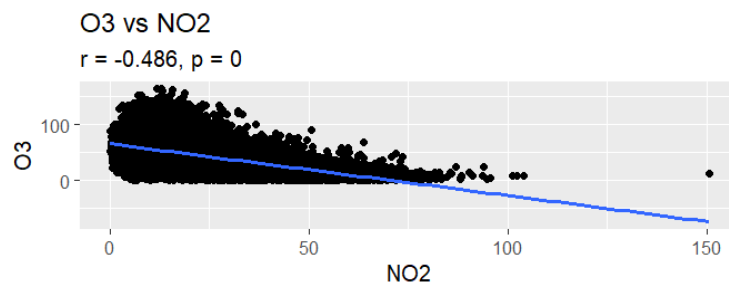


Fig3: Correlation between ozone and nitrogen dioxide

Challenges and Things Learned

The most difficult part of this assignment was the statistical testing. While comparing the levels of ozone (O3) upon the wind speed, I should have normally checked for normality and equal variances by shapiro.test() and leveneTest(), respectively. At first, I made an error by

conducting a t-test without checking any of these two assumptions. Having received the wrong answers, later I learned how to check on data distribution first. After finding that data were not normally distributed, I correctly switched to its non-parametric alternative-the Wilcoxon rank-sum test.

Checking whether the variance is equal in two groups.

```
leveneRes <- leveneTest(O3 ~ windGrp, data = df) print(leveneRes)
```

```
safeShapiro <- function(r) { n <- length(r) if (n >= 3 && n <= 5000) { return(shapiro.test(r)$p.value) } else { return(NA) } } lowO3 <- df$O3[df$windGrp == "low"] highO3 <- df$O3[df$windGrp == "high"] pLow <- safeShapiro(lowO3) pHigh <- safeShapiro(highO3)
```

```
cat("Low group Shapiro p =", pLow, "\n") cat("High group Shapiro p =", pHigh, "\n")
```

If normality is correct in both groups ($p > 0.05$) then it will be TRUE.

```
nrmlOk <- (!is.na(pLow) && pLow > 0.05) && (!is.na(pHigh) && pHigh > 0.05) if (nrmlOk) { tRes <- t.test(O3 ~ windGrp, data = df, var.equal = FALSE) # Welch print(tRes)
```

effect size

```
d <- cohen.d(O3 ~ windGrp, data = df) print(d) } else { wRes <- wilcox.test(O3 ~ windGrp, data = df) print(wRes) }
```

Another tricky part was the ANOVA analysis for NO₂ concentrations across 2016, 2018, and 2020. Residuals were not normal, and variances were unequal. Thus, I could not use classical ANOVA. I learned how to apply a Kruskal–Wallis test as a non-parametric alternative and did a Dunn's post-hoc test for pairwise comparison. This deepened my understanding of selecting appropriate tests based on data characteristics. Improvements for Future Work If I were to run the analysis again, I would make the results more dynamic and user-friendly by including interactive visualizations using Plotly or Shiny. I would automate missing data imputation through statistical and machine learning techniques in order to avoid manual filtering. Time-series decomposition would also be done to observe seasonal variations in pollutants, as this was not within the scope of the present work.

Anova Testing

```
if (shapiroResid$p.value > 0.05 && leveneRes2$Pr(>F)[1] > 0.05) {
```

using classical Anova

```
summary(anv0)
```

effect size

```
anvRes <- anovaTest(data = dfAnova, dv = NO2, between = yearFac) print(anvRes) eta <- dfAnova %>%  
anovaTest(NO2 ~ yearFac) %>% getAnvTbl() %>% .[["ges"]]
```

post-hoc Tukey

```
tukeyRes <- TukeyHSD(anv0) print(tukeyRes) } else {
```

Kruskal-Wallis (non-parametric)

```
krsklW <- kruskal.test(NO2 ~ yearFac, data = dfAnova) print(krsklW) if (krsklW$p.value < 0.05)  
{ dunnRes <- dunn_test(dfAnova, NO2 ~ yearFac, p.adjust.method = "bonferroni") print(dunnRes) } }
```

Use of Generative AI I used a generative AI system, ChatGPT and Claude, to help me refine to the improve clarity in phrasing; however, all code was written by me, executed by me, and all outputs interpreted by me. The analysis, visualizations, and statistical reasoning were wholly done by me in my own understanding and effort.

Conclusion

The project generally helped me improve my R programming and data analysis skills immensely. I learned how to manage data cleaning, create clear visualizations, and choose proper statistical tests based on assumptions. Having done all the steps on my own, I developed deeper confidence in conducting real-world environmental data analysis in R.