Lakes Water Quality Data Preprocessing

Hayley Dinh

Setup

```
# Load the necessary packages required to reproduce the report. For example:

library(kableExtra)
library(magrittr)
library(readxl)
library(dplyr)
library(tidyr)
library(lubridate)
library(Hmisc)
library(ggplot2)
library(patchwork)
library(MVN)
```

Executive Summary

This report presents the process and outcomes of wrangling water quality data from the Gippsland Lakes and Western Port regions, collected between 1990 and 2024 with R. The primary goal was to merge the data sets and clean it to support accurate and meaningful environmental analysis.

The process included converting data types, tidying data, manipulating variables, scanning and handling missing/ special values and univariate as well as multivariate outliers, and applying multiple transformations to reduce the skewness of data. The results revealed that the data sets are quite problematic, featuring a significant number of extreme values and exhibiting highly right-skewed distributions. Regarding outliers, the boxplot method was used to detect univariate outliers. Rather than removing these outliers which likely resulted from natural phenomena or sampling site differences, capping methods were applied to preserve data integrity. Similarly, the Mahalanobis method was used to detect and diagnose potential multivariate outliers. Regarding transformations, square root, reciprocal, and logarithmic were executed and carefully compared to determine the most suitable technique. The chosen methods were able to effectively address these issues to deliver clean and workable data ready for further in depth environmental analysis.

Data

The data sets were sourced from the following:

Gippsland Lakes Water Quality (https://discover.data.vic.gov.au/dataset/gippsland-lakes-water-quality-data-1990-2024/resource/d67dd6e7-e655-4745-8b49-b4c1dd101f7f)

Western Port Water Quality (https://discover.data.vic.gov.au/dataset/epa-western-port-water-quality-data-1990-2024)

The two selected data sets are both recording water quality data from 1990 to 2024. They have the same variables and data structure, making them relational. In this case, using a join function with cause two observations from two sites to be stored in one row, making it untidy. For that reason, they can be subsetted and seamlessly stacked on top of each other to form a large merged data set. The original data contains 25 variables explained in the following:

- Site_id: Unique id number of sampling site
- Site_name_short: Name of sampling site
- Water_body: Name of the larger body of water that the site belongs to
- Date: Date of measurements
- Type: Signifies whether the sample was taken from the surface or bottom of the site
- CHL_A: Measure of Chlorophyll a in ug/L
- Secchi_depth_m: Measure of Secchi Disk depth in meters

- DO_mg: Measure of dissolved oxygen in mg/L
- FL: Measure of Fluorescence in mg/m-3
- PAR: Measure of Photosynthetically Active Radiation - Water in UEINSTEINS/SQ.M/SEC
- Sal: Measure of salinity in PSU
- Temp: Measure of water temperature in degrees C with SEACAT method
- Turb: Measure of Turbidity in NTU
- DO_sat: Percentage of saturated dissolved oxygen
- Temperature: Measure of water temperature in degrees C with thermometer
- pH: Measure of pH in units
- TSS: Measure of Suspended solids in mg/L
- N_NH3: Measure of Ammonia in ug/L
- N_NO2: Measure of Nitrogen Dioxide in ug/L
- N_NO3: Measure of Nitrate in ug/L
- N_NOX: Measure of oxidised nitrogen in ug/L
- N_TOTAL: Measure of total nitrogen in ug/L
- P_PO4: Measure of Orthosphosphate in ug/L
- P_TOTAL: Measure of total phosphorus in ug/L
- SI: Measure of Silica in ug/L

Because this is a highly complex data set with a very large number of variables, most of them will be removed to deliver a more detailed analysis into relevant variables. A total of 13 variables will be removed, creating a new smaller data with 12 variables, being Site_id, Site_name_short, Water_body, Date, Type, N_NH3, N_NO2, N_NO3, N_NOX, N_TOTAL, P_PO4, P_TOTAL.

```
#setting work directory and importing data sets
setwd("~/Documents/Masters/Data Wrangling/data")
Gippsland = read_excel("Gippsland Lakes Water Quality Data.xlsx")
Western_port = read_excel("Western Port Water Quality Data.xlsx")

#setting seed
set.seed(3931982)

#subsetting chosen data sets
subset_g = Gippsland[sample(nrow(Gippsland), 2676), ]
subset_wp = Western_port[sample(nrow(Western_port), 864), ]

#merge two data sets on top of each other
df = bind_rows(subset_g,subset_wp)

#removing unnecessary variables
df = df %>% select(-CHL_A, -Secchi_depth_m,-DO_mg, -FL, -PAR, -Sal, -Temp, -DO_sat, -Temperatur
e, -pH, -TSS, -SI, -Turb)
```

Understanding Data Structure

```
#checking initial structure
str(df)
```

```
## tibble [3,540 × 12] (S3: tbl_df/tbl/data.frame)
##  $ site_id        : num [1:3540] 2311 2316 2322 2314 2311 ...
##  $ site_name_short: chr [1:3540] "Lake Victoria" "Lake King North" "Shaving Point" "Lake Kin
g South" ...
##  $ water_body     : chr [1:3540] "Gippsland Lakes" "Gippsland Lakes" "Gippsland Lakes" "Gipp
sland Lakes" ...
##  $ date           : POSIXct[1:3540], format: "2009-09-29" "2023-01-24" ...
##  $ Type           : chr [1:3540] "bottom" "surface" "bottom" "surface" ...
##  $ N_NH3          : chr [1:3540] "43.202300000000001" "7" "224.06700000000001" "4.32" ...
##  $ N_NO2          : chr [1:3540] "1.9914000000000001" "1" "5.3162000000000003" "0.06" ...
##  $ N_NO3          : chr [1:3540] "2.2422" "5" "46.054099999999998" "0.45" ...
##  $ N_NOX          : chr [1:3540] "4.2336" "6" "51.3703" "0.52" ...
##  $ N_TOTAL        : chr [1:3540] "451.44900000000001" "570" "658.03499999999997" "476.43"
...
##  $ P_PO4          : chr [1:3540] "2.4641000000000002" "8" "17.584" "11.26" ...
##  $ P_TOTAL        : chr [1:3540] "38.759" "50" "54.591299999999997" "48.41" ...
```

```
#converting data types
df$date = as.Date(df$date)
df$water_body = as.factor(df$water_body)
df$Type = factor(df$Type, levels = c("bottom","surface"), labels = c("Bottom Sample","Surface S
ample"))
df = df %>%
  mutate(across(c(N_NH3, N_NO2, N_NO3, N_NOX, N_TOTAL, P_PO4, P_TOTAL), as.numeric))

#checking transformed structure
str(df)
```

```
## tibble [3,540 × 12] (S3: tbl_df/tbl/data.frame)
##  $ site_id        : num [1:3540] 2311 2316 2322 2314 2311 ...
##  $ site_name_short: chr [1:3540] "Lake Victoria" "Lake King North" "Shaving Point" "Lake Kin
g South" ...
##  $ water_body     : Factor w/ 2 levels "Gippsland Lakes",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ date           : Date[1:3540], format: "2009-09-29" "2023-01-24" ...
##  $ Type           : Factor w/ 2 levels "Bottom Sample",..: 1 2 1 2 2 1 2 1 2 1 ...
##  $ N_NH3          : num [1:3540] 43.2 7 224.07 4.32 3.84 ...
##  $ N_NO2          : num [1:3540] 1.991 1 5.316 0.06 0.399 ...
##  $ N_NO3          : num [1:3540] 2.242 5 46.054 0.45 0.928 ...
##  $ N_NOX          : num [1:3540] 4.23 6 51.37 0.52 1.33 ...
##  $ N_TOTAL        : num [1:3540] 451 570 658 476 535 ...
##  $ P_PO4          : num [1:3540] 2.46 8 17.58 11.26 4.13 ...
##  $ P_TOTAL        : num [1:3540] 38.8 50 54.6 48.4 78.5 ...
```

As a result of reading in data from excel, most variables are stored as a character variable which is incorrect and will inhibit further analysis. After carefully reading variable descriptions and examining the actual data, each variable is converted into their proper type. Water_body was converted into a factor and type was converted into a labelled factor with values “Bottom sample” and “Surface sample” for better clarity. None of these factor variables represent a ranking, therefore, there is no need for ordering. The date variable was converted into the correct date format. Lastly, the rest of the variables were converted into numeric variables, as they all carry the measurement values. Lastly, site_name_short with values as the actual names of sampling locations can be left as a character variable. All these details are reflected in the transformed structure.

Tidy & Manipulate Data I

```
#removing duplicate data
df_unique = df %>%
  group_by(site_id, site_name_short, water_body, date, Type) %>%
  summarise(across(c(N_NH3, N_NO2, N_NO3, N_NOX, N_TOTAL, P_PO4, P_TOTAL), ~ mean(.x, na.rm = T
RUE)), .groups = "drop")

#separate date column into year, month and day
df_unique$year = df_unique$date %>% year()
df_unique$month = df_unique$date %>% month()
df_unique$day = df_unique$date %>% day()

#remove date column
df_unique = df_unique %>% select(-date)

#show new variables
colnames(df_unique)
```

```
##  [1] "site_id"         "site_name_short" "water_body"      "Type"
##  [5] "N_NH3"           "N_NO2"           "N_NO3"           "N_NOX"
##  [9] "N_TOTAL"         "P_PO4"           "P_TOTAL"         "year"
## [13] "month"           "day"
```

During the process of examining the data, it was apparent that some sites have more than one measurement of the same type on the same day. Which may cause complications in further tidying or grouping of data. For example, if an analyst were to transform the data into long format and later revert it back to wide format, multiple values will be grouped into one cell. To avoid that, duplicate measurements will be averaged so that on the same day, each site will only have one measurement of each type. Additionally, the date column will be separated into three columns of year, month and day. This allows for easier grouping and comparison of time periods.

Tidy & Manipulate Data II

```
#create new variable
df_unique$N_MEAN = (df_unique$N_NH3 + df_unique$N_NO2 + df_unique$N_NO3 + df_unique$N_NOX)/4
df_unique$NO_TOTAL = df_unique$N_NO2 + df_unique$N_NO3

#show new variables
colnames(df_unique)
```

```
##  [1] "site_id"         "site_name_short" "water_body"      "Type"
##  [5] "N_NH3"           "N_NO2"           "N_NO3"           "N_NOX"
##  [9] "N_TOTAL"         "P_PO4"           "P_TOTAL"         "year"
## [13] "month"           "day"             "N_MEAN"          "NO_TOTAL"
```

As all of the selected variables are all under the same unit, a new variable can be formed with the mean of the nitrogen measurements. This allows for comparison of these the concentration of just these four compounds across time in different areas. Similarly, a new variable can be created by combining the concentration of N_NO2 and N_NO3 to show the total concentration of NO based chemicals.

Scan For Missing Values

```
#removing negative values while not affecting NAs or NaNs
df_unique = df_unique %>%
  filter(if_all(where(is.numeric), ~ is.na(.) | . >= 0))

#recording empty strings as NA
df_unique[df_unique == "" ] = NA

#check for NA and special values
colSums(is.na(df_unique))
```

```
##         site_id site_name_short      water_body            Type           N_NH3
##               0               0               0               0             185
##           N_NO2           N_NO3           N_NOX         N_TOTAL           P_PO4
##             192             213             224             185             193
##         P_TOTAL            year           month             day          N_MEAN
##             186               0               0               0             231
##        NO_TOTAL
##             217
```

```
sapply(df_unique, function(x) sum(is.infinite(x)))
```

```
##         site_id site_name_short      water_body            Type           N_NH3
##               0               0               0               0               0
##           N_NO2           N_NO3           N_NOX         N_TOTAL           P_PO4
##               0               0               0               0               0
##         P_TOTAL            year           month             day          N_MEAN
##               0               0               0               0               0
##        NO_TOTAL
##               0
```

```
sapply(df_unique, function(x) sum(is.nan(x)))
```

```
##         site_id site_name_short      water_body            Type           N_NH3
##               0               0               0               0             185
##           N_NO2           N_NO3           N_NOX         N_TOTAL           P_PO4
##             192             213             224             185             193
##         P_TOTAL            year           month             day          N_MEAN
##             186               0               0               0             231
##        NO_TOTAL
##             217
```

```
#impute NA values with median of column
df_unique = df_unique %>%
  mutate(across(where(is.numeric), ~ ifelse(is.na(.x), median(.x, na.rm = TRUE), .x)))
```

In this specific case, measurements are representing chemical concentrations, this means that negative values are invalid and likely caused by faulty equipment or processes. For that reason, these observations will be removed beforehand.

Empty strings are then recoded as NAs so that they can be detected later. After that, the data is scanned for NAs, infinite values and NaN values. It is to be noted that scanning for NAs also includes NaN values. As a result, it was returned that the data set has no infinite values and interestingly, each column has the same number of NAs and NaN values. After looking into the actual data again, it can be determined that all NA values has been transformed into NaN values through the removal of duplicate observations.

Now that the missing values are clearly defined, they will be handled through the process of imputing with column means. The choice between mean and median are usually determined by the skewness of the data or the presence of a large number of outliers. Consequently, as outliers have not been handled, it is generally safer to use median to ensure the data will not be distorted with unsually high or low values.

Scan For Outliers

```
#boxplots and univariate outliers of N_NH3
NH3_boxplot = ggplot(df_unique, aes(x = "", y = N_NH3)) +
  geom_boxplot(outlier.colour = "red") +
  xlab("N_NH3") +
  ylab("ug/L")
length(boxplot.stats(df_unique$N_NH3)$out)
```

```
## [1] 381
```

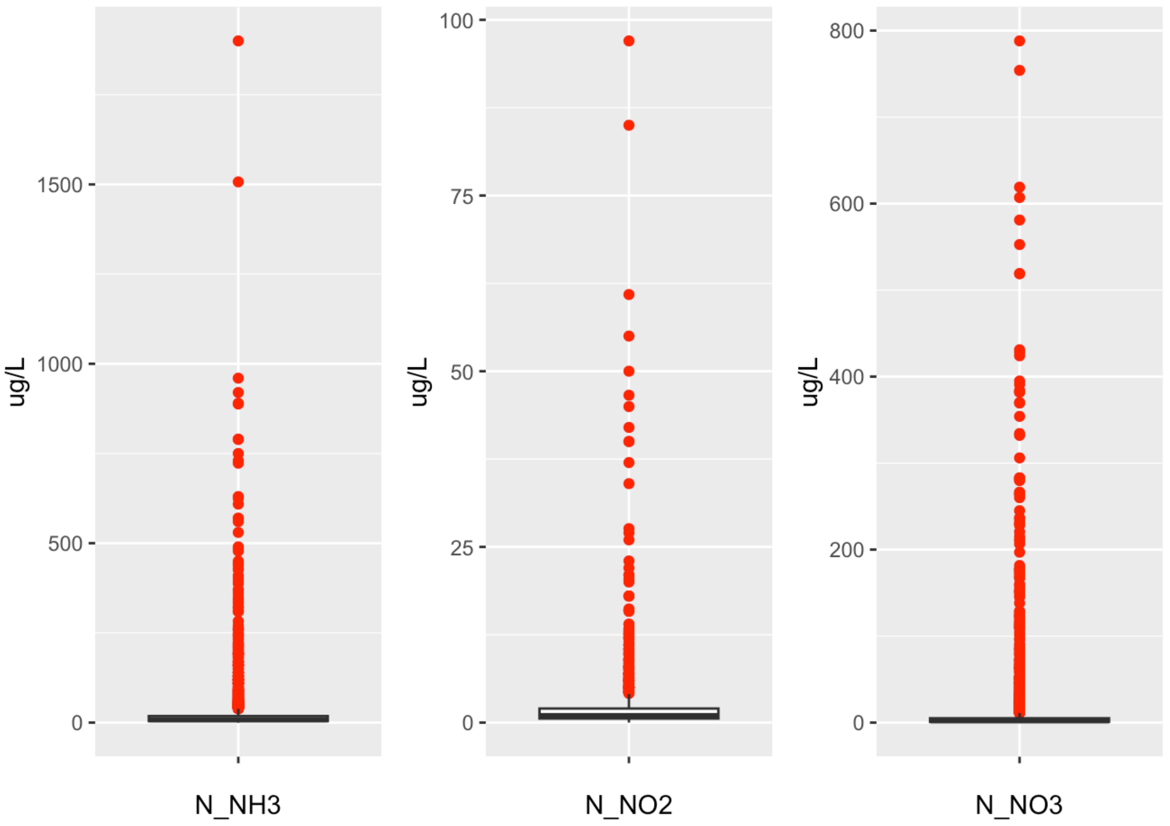
```
#boxplots and univariate outliers of N_NO2
NO2_boxplot = ggplot(df_unique, aes(x = "", y = N_NO2)) +
  geom_boxplot(outlier.colour = "red") +
  xlab("N_NO2") +
  ylab("ug/L")
length(boxplot.stats(df_unique$N_NO2)$out)
```

```
## [1] 186
```

```
#boxplots and univariate outliers of N_NO3
NO3_boxplot = ggplot(df_unique, aes(x = "", y = N_NO3)) +
  geom_boxplot(outlier.colour = "red") +
  xlab("N_NO3") +
  ylab("ug/L")
length(boxplot.stats(df_unique$N_NO3)$out)
```

```
## [1] 324
```

```
NH3_boxplot + NO2_boxplot + NO3_boxplot
```



```
#boxplots and univariate outliers of N_NOX
NOX_boxplot = ggplot(df_unique, aes(x = "", y = N_NOX)) +
  geom_boxplot(outlier.colour = "red") +
  xlab("N_NOX") +
  ylab("ug/L")
length(boxplot.stats(df_unique$N_NOX)$out)
```

```
## [1] 312
```

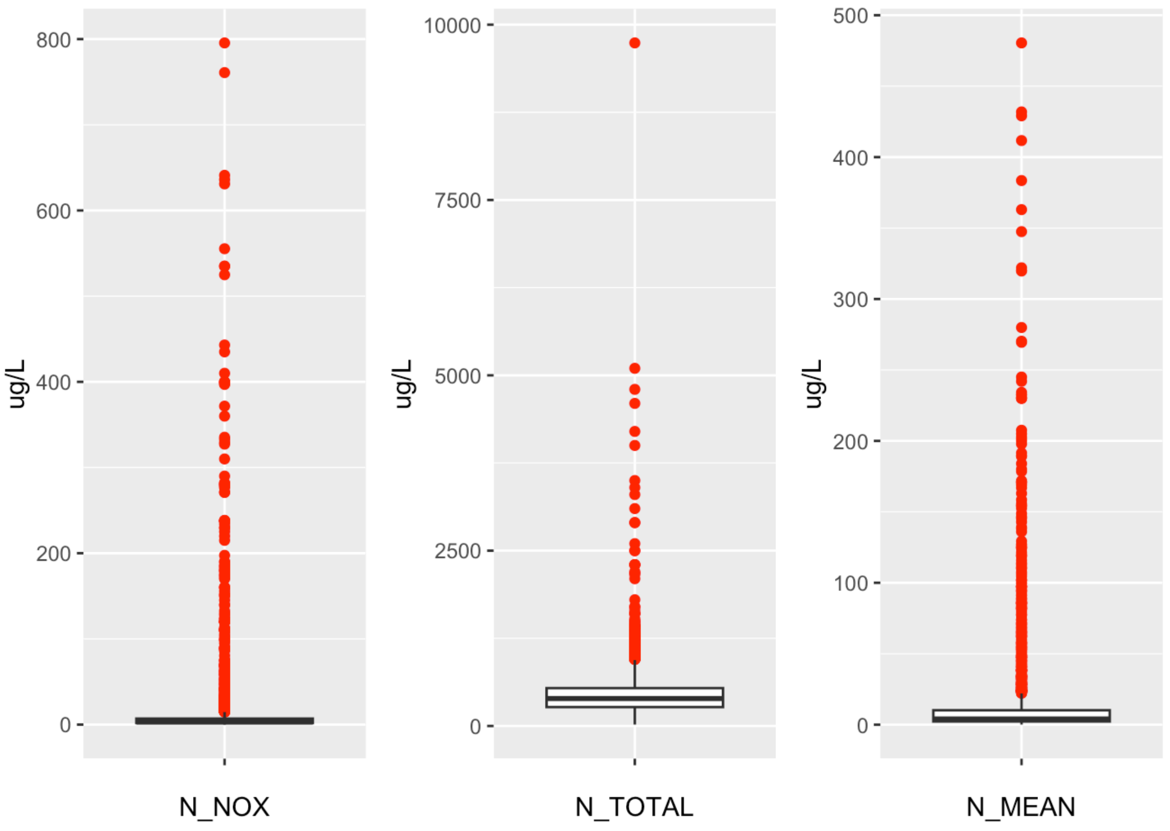
```
#boxplots and univariate outliers of N_TOTAL
NTotal_boxplot = ggplot(df_unique, aes(x = "", y = N_TOTAL)) +
  geom_boxplot(outlier.colour = "red") +
  xlab("N_TOTAL") +
  ylab("ug/L")
length(boxplot.stats(df_unique$N_TOTAL)$out)
```

```
## [1] 150
```

```
#boxplots and univariate outliers of N_MEAN
NMean_boxplot = ggplot(df_unique, aes(x = "", y = N_MEAN)) +
  geom_boxplot(outlier.colour = "red") +
  xlab("N_MEAN") +
  ylab("ug/L")
length(boxplot.stats(df_unique$N_MEAN)$out)
```

```
## [1] 349
```

```
NOX_boxplot + NTotal_boxplot + NMean_boxplot
```



```
#boxplots and univariate outliers of NO_TOTAL
NOTOTAL_boxplot = ggplot(df_unique, aes(x = "", y = NO_TOTAL)) +
  geom_boxplot(outlier.colour = "red") +
  xlab("NO_TOTAL") +
  ylab("ug/L")
length(boxplot.stats(df_unique$NO_TOTAL)$out)
```

```
## [1] 313
```

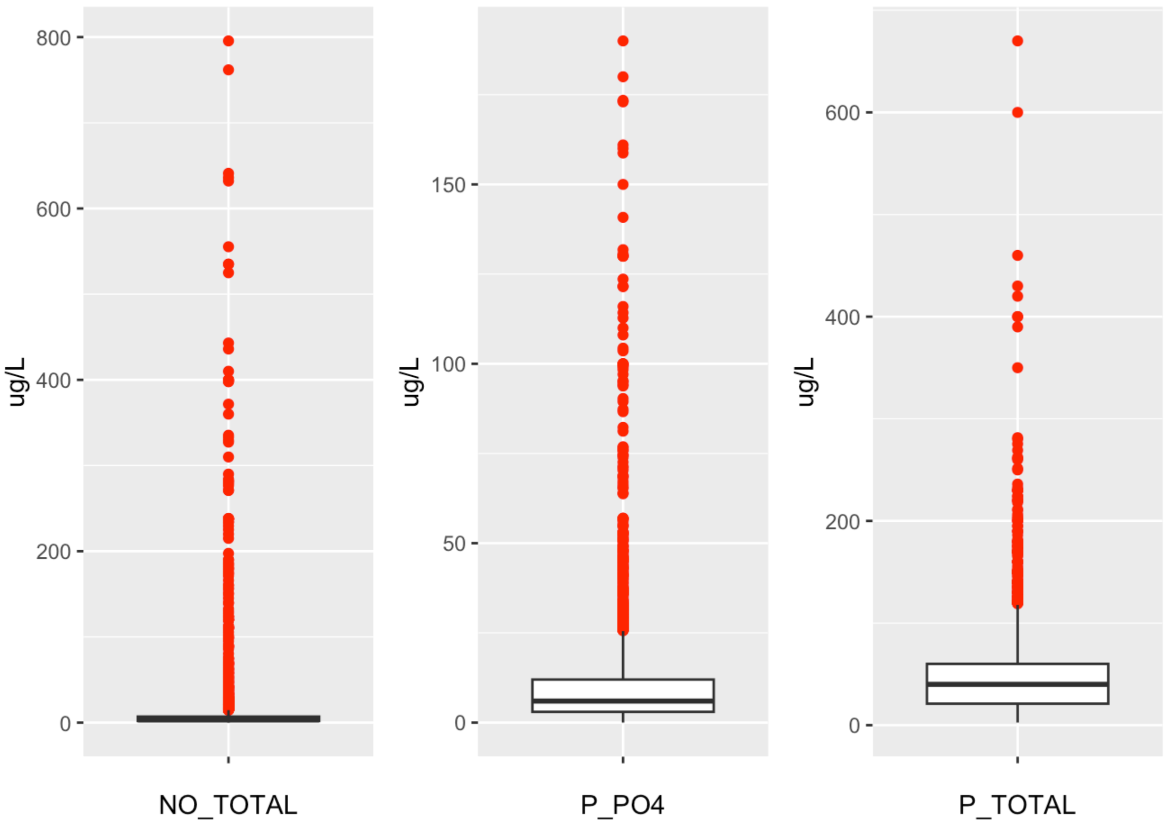
```
#boxplots and univariate outliers of P_PO4
PO4_boxplot = ggplot(df_unique, aes(x = "", y = P_PO4)) +
  geom_boxplot(outlier.colour = "red") +
  xlab("P_PO4") +
  ylab("ug/L")
length(boxplot.stats(df_unique$P_PO4)$out)
```

```
## [1] 224
```

```
#boxplots and univariate outliers of P_TOTAL
PTotal_boxplot = ggplot(df_unique, aes(x = "", y = P_TOTAL)) +
  geom_boxplot(outlier.colour = "red") +
  xlab("P_TOTAL") +
  ylab("ug/L")
length(boxplot.stats(df_unique$P_TOTAL)$out)
```

```
## [1] 149
```

```
NOTOTAL_boxplot + PO4_boxplot + PTotal_boxplot
```



```
#custom function to cap values
cap = function(x){
  quantiles <- quantile( x, c(0.25, 0.75) )
  x[ x < quantiles[1] - 1.5*IQR(x) ] = quantiles[1] - 1.5*IQR(x)
  x[ x > quantiles[2] + 1.5*IQR(x) ] = quantiles[2] + 1.5*IQR(x)
  return (x)
  }

#cap values
df_unique = df_unique %>% mutate(across(c(N_NH3, N_NO2, N_NO3, N_NOX, N_TOTAL, P_PO4, P_TOTAL),
cap))

#recalculate new variables
df_unique$N_MEAN = (df_unique$N_NH3 + df_unique$N_NO2 + df_unique$N_NO3 + df_unique$N_NOX)/4
df_unique$NO_TOTAL = df_unique$N_NO2 + df_unique$N_NO3

#check if univariate outlier removal was successful
length(boxplot.stats(df_unique$P_TOTAL)$out)
```

```
## [1] 0
```

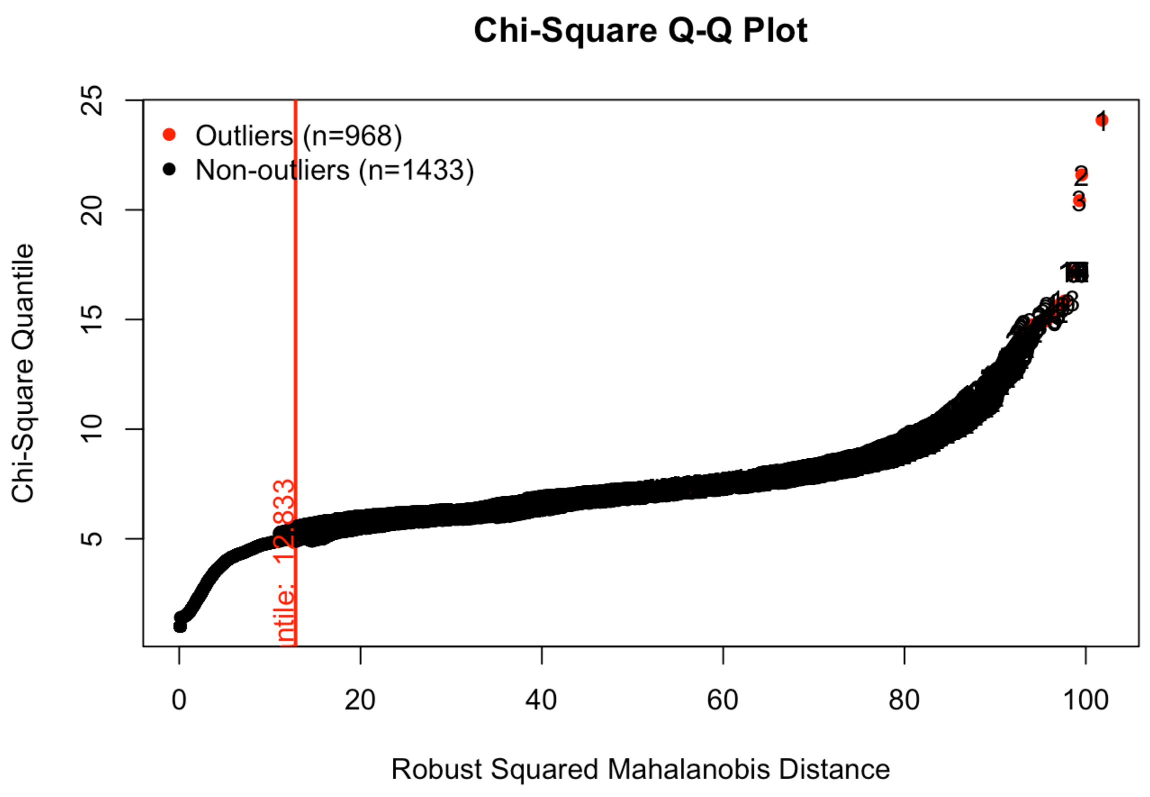
```
length(boxplot.stats(df_unique$N_MEAN)$out)
```

```
## [1] 0
```

```
#checking for multivariate outliers
df_numeric = df_unique[,c(5:11,15)]
cor(df_numeric)
```

```
##              N_NH3     N_NO2      N_NO3      N_NOX   N_TOTAL     P_PO4
## N_NH3   1.0000000 0.6083529 0.45056248 0.50646883 0.2795784 0.4908395
## N_NO2   0.6083529 1.0000000 0.63246109 0.74499064 0.2976462 0.2467141
## N_NO3   0.4505625 0.6324611 1.00000000 0.97690924 0.1314529 0.1419947
## N_NOX   0.5064688 0.7449906 0.97690924 1.00000000 0.1790448 0.1764657
## N_TOTAL 0.2795784 0.2976462 0.13145295 0.17904478 1.0000000 0.2781164
## P_PO4   0.4908395 0.2467141 0.14199465 0.17646566 0.2781164 1.0000000
## P_TOTAL 0.2641217 0.1583762 0.04021194 0.07712632 0.7801937 0.5497030
## N_MEAN  0.9132650 0.7684627 0.76731850 0.81268562 0.2732417 0.4124803
##            P_TOTAL    N_MEAN
## N_NH3   0.26412175 0.9132650
## N_NO2   0.15837624 0.7684627
## N_NO3   0.04021194 0.7673185
## N_NOX   0.07712632 0.8126856
## N_TOTAL 0.78019365 0.2732417
## P_PO4   0.54970296 0.4124803
## P_TOTAL 1.00000000 0.2122810
## N_MEAN  0.21228105 1.0000000
```

```
df_numeric = select(df_numeric,-N_MEAN, -N_NOX, -P_TOTAL)
outliers_result = mvn(data = df_numeric, multivariateOutlierMethod = "quan", showOutliers = TRU
E)
```



```
outliers_result$multivariateOutliers
```

	Observation <chr>	Mahalanobis Distance <dbl>	Outlier <chr>
1	1	101.783	TRUE
2	2	99.544	TRUE
3	3	99.299	TRUE

	Observation <chr>	Mahalanobis Distance <dbl>	Outlier <chr>
4	4	98.682	TRUE
5	5	98.682	TRUE
6	6	98.682	TRUE
7	7	98.682	TRUE
8	8	98.682	TRUE
9	9	98.682	TRUE
10	10	98.682	TRUE

1-10 of 968 rows Previous **1** 2 3 4 5 6 ... 97 Next

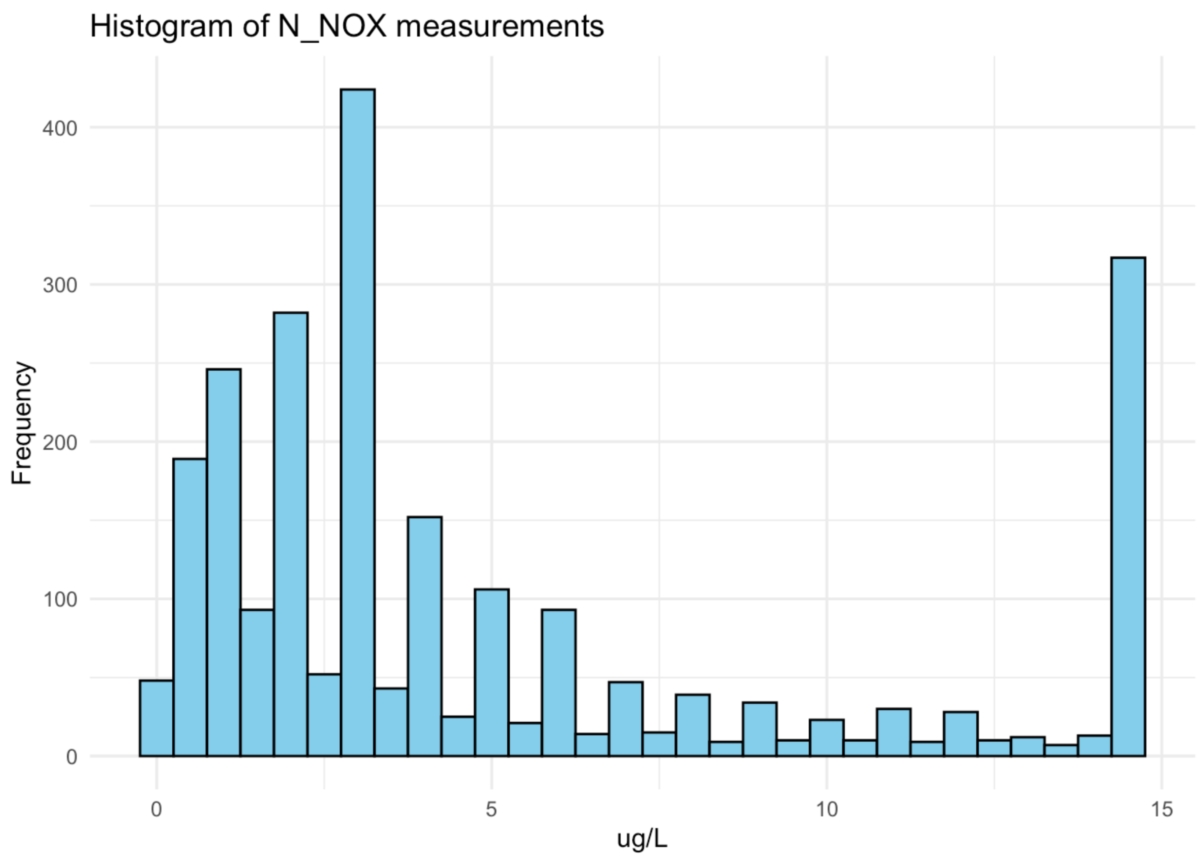
In order, univariate outliers will be tackled before multivariate outliers. To execute this, each variable was represented with a boxplot to visually represent outliers. Through the boxplots, it can be seen that all eight numeric variables are greatly right-skewed with a large number of very extreme higher outliers and no lower outliers with the exception of N_NO2, which has both types of outliers. The count of outliers was also indicated according to the boxplot method, outliers range from under 200 to under 400 values across variables which represents from around 8% to 16% of the observations. This is a significant amount, which means that deleting outliers is not an appropriate option.

Because this is an environmental data set, extreme values are not uncommon possibly due to changing seasons, intense weather conditions, human-caused impacts or natural disasters such as hurricanes, typhoons, hailstorms, droughts … Also, due to the large number of outliers, it is likely not caused by data entry or measurement errors. Hence, capping proves a viable method. As the boxplot method was used to detect outliers, the outlier definition in this case are values above the 75% percentile + 1.5 IQR and similarly values under the 25% - 1.5 IQR. Therefore, that is the capping boundary. This is executed through a custom function pulling all values within the fences of the boxplots. Finally, as the N_MEAN variable is a result of mathematical computation of 4 other variables, it is simply recalculated. The count of outliers of two variables are checked once again to confirm the results.

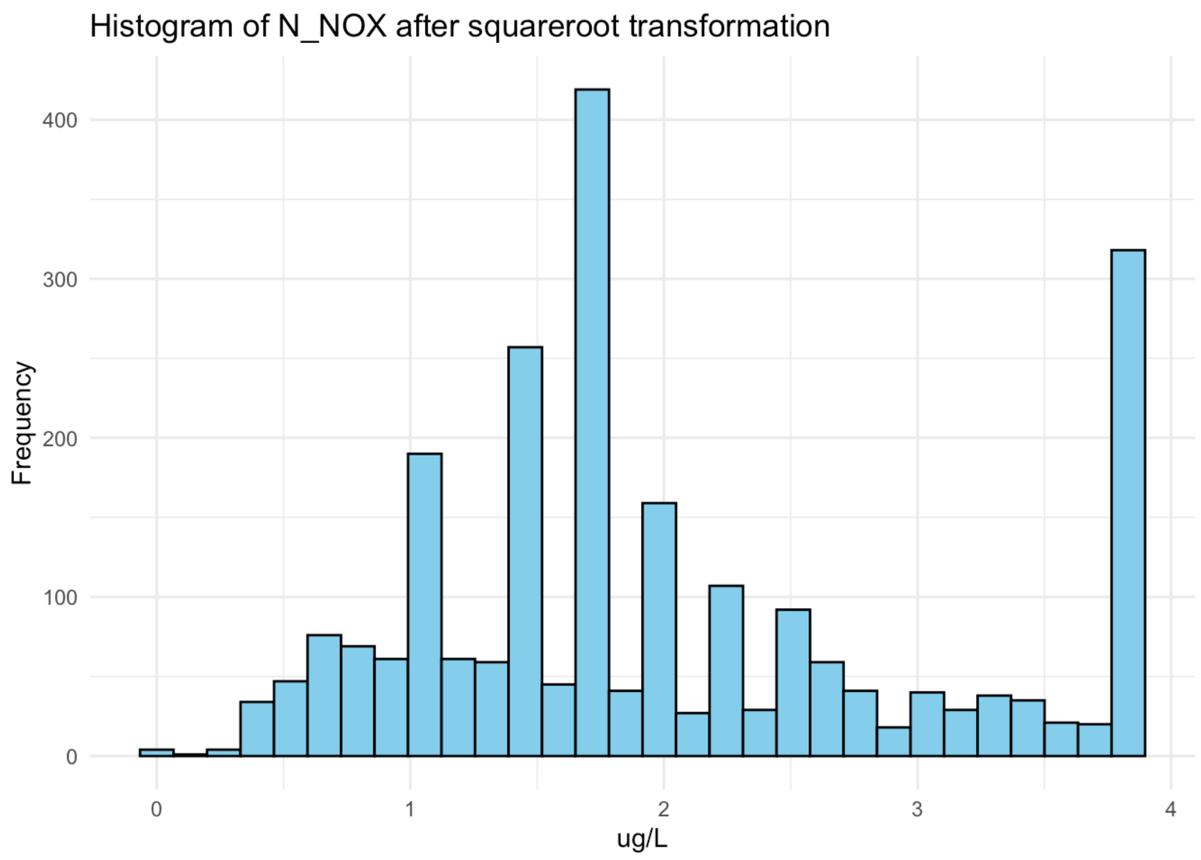
Regarding multivariate outlier detection, the numeric variables were separated and highly correlated variables were removed to enable later steps. Using the Mahalanobis method, the Chi-square Q-Q plot suggests the existence of 968 potential outliers, which makes up 40.3% of the data, almost half of the data set. This is very unusual and proposes that these are not real outliers, rather, it represents different groups of samples with substantially different means. This is also shown through the fact that the original data was sorted based on the sampling site which consequently caused the first 968 observations to be flagged. It is not unrealistic for natural chemical concentrations to differ considerably based on location. However, this can not be confirmed without much more in depth analysis methods, for instance, clustering the data. Nonetheless, such a large amount of outliers should not be automatically removed or imputed as it risks severly biasing the data.

Transform

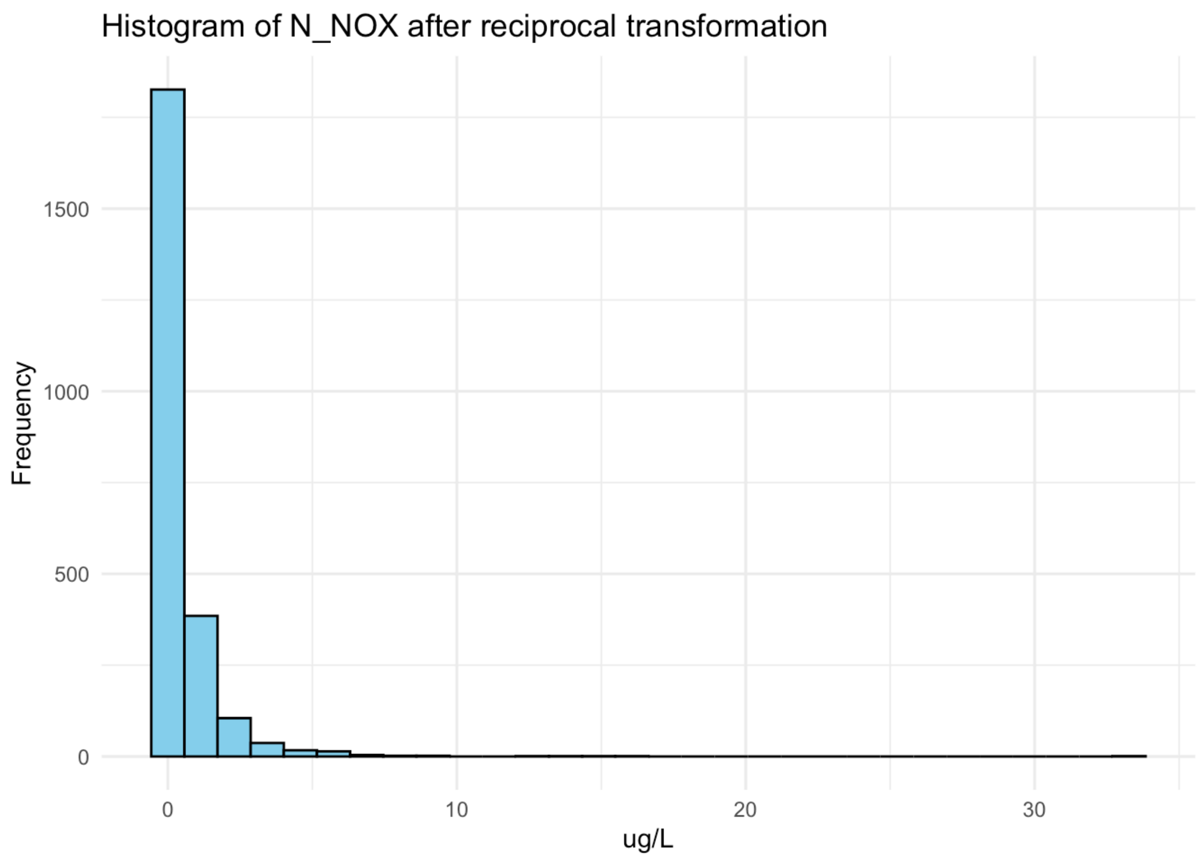
```r
#Histogram of N_NOX before transformation
ggplot(df_unique, aes(x = N_NOX)) +
  geom_histogram(binwidth = 0.5, fill = "skyblue", color = "black") +
  labs(title = "Histogram of N_NOX measurements",
       x = "ug/L",
       y = "Frequency") +
  theme_minimal()
```



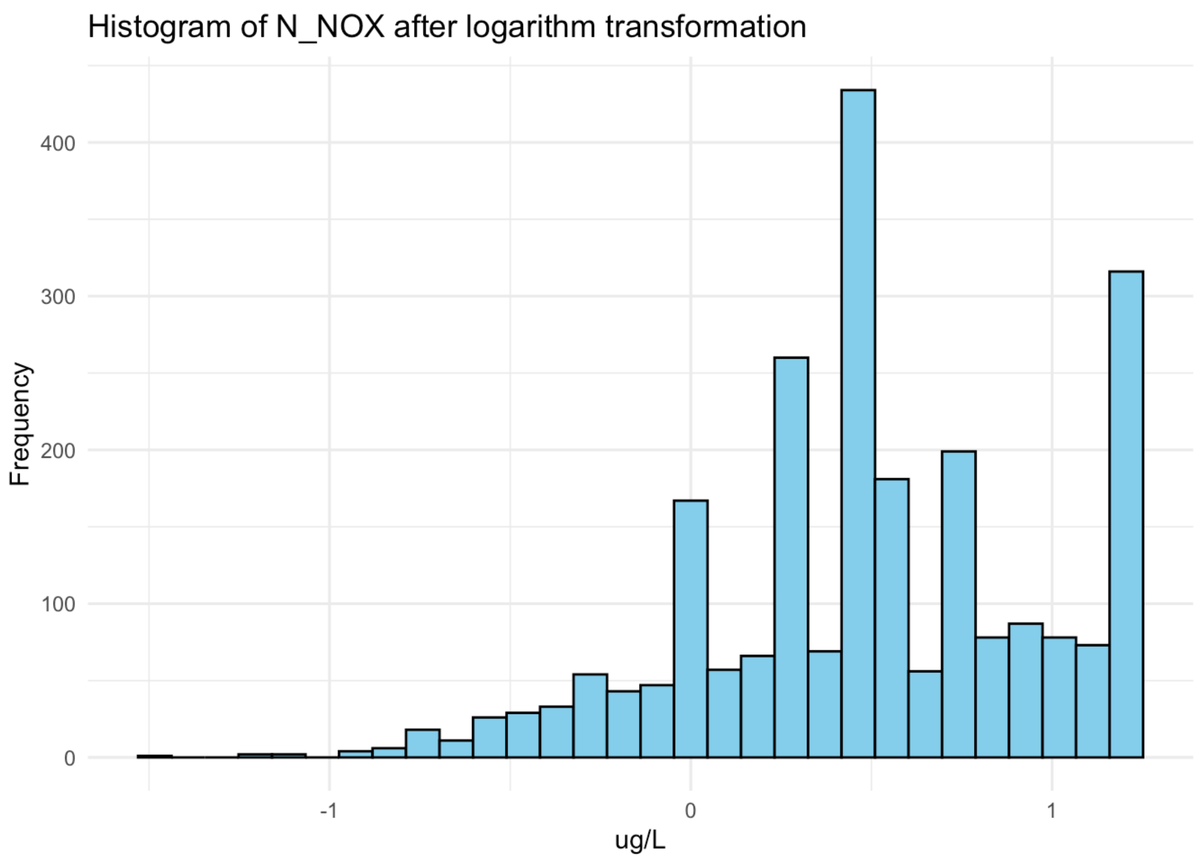
```
#Applying transformations to N_NOX and storing them as new variables
df_unique$Sqrt_NOX = sqrt(df_unique$N_NOX)
df_unique$Rec_NOX = 1/(df_unique$N_NOX)
df_unique$Log_NOX = log10(df_unique$N_NOX)

#Histogram of N_NOX after squareroot transformation
ggplot(df_unique, aes(x = Sqrt_NOX)) +
  geom_histogram(fill = "skyblue", color = "black") +
  labs(title = "Histogram of N_NOX after squareroot transformation",
       x = "ug/L",
       y = "Frequency") +
  theme_minimal()
```



```
#Histogram of N_NOX after reciprocal transformation
ggplot(df_unique, aes(x = Rec_NOX)) +
  geom_histogram(fill = "skyblue", color = "black") +
  labs(title = "Histogram of N_NOX after reciprocal transformation",
       x = "ug/L",
       y = "Frequency") +
  theme_minimal()
```



```
#Histogram of N_NOX after logarithm transformation
ggplot(df_unique, aes(x = Log_NOX)) +
  geom_histogram(fill = "skyblue", color = "black") +
  labs(title = "Histogram of N_NOX after logarithm transformation",
       x = "ug/L",
       y = "Frequency") +
  theme_minimal()
```



As seen in previous boxplots, all variables in this data set is largely right-skewed. One variable was chosen at random, N_NOX, to undergo transformation to reduce skewness. Squareroot, reciprocals and logarithms are all appropriate methods to reduce right-skewness. Therefore, all three were executed to determine the method with the best results. The histogram of N_NOX before transformation was provided for easy comparison. Squareroot transformation reduced the skewness quite a bit and but there is still quite a heavy right tail. Using reciprocal reduced the tail but the data is still severely right-skewed. Finally, using logarithm seems to achieve the highest level of symmetry and compression of extreme values. As a result, the logarithm transformation is most suitable for this variable.