University of San Diego
SHILEY-MARCOS SCHOOL OF ENGINEERING

ADS-506 – Final Team Project Start Form

Fill out this form and submit it by the end of Module 2 in Blackboard.

Team Number: 1

Team Leader/Representative: Aaron Carr

Full Names of Team Members:

1. Brianne Bell
2. Connie Chow
3. Aaron Carr

Working Title of Your Time Series Final Project:

Forecasting the Ocean Quality by San Diego

Motivation for choosing this project:

Climate change and warming oceans impacting marine life and atmosphere conditions.

Problem Statement: Short Description of Your Time Series Project and Objective(s):
Utilize ocean water measurements (salinity, temperature, density, chlorophyll, dissolved oxygen, and pH) to see impact over time. Ideally, predict specific parameters (not all) in a set time period in the future, based on the measurements from 2020-2021 or 1990-2021.

Name of Your Selected Dataset:
Water Quality-Ocean Monitoring Program; Parameter Results

Description of your selected dataset:
Measurements of different parameters for samples of ocean water taken at different locations for San Diego, CA each day.

Data source, number of variables, size of dataset, etc:

Water Quality - Ocean Monitoring Program - City of San Diego Open Data Portal

We are going to utilize the datasets from 1990-2021 to have an abundant source of time points to develop forecasts with.
The parameters are listed in a single column but include fluorometry, density, dissolved oxygen, entero (bacterial), fecal, OG, pH, salinity, SUSO, temperature, and percent light. We will focus on only some of these parameters.
Additionally, the dataset includes sample station location, retrieval date and time, depth (in meters), measurement value, units of measurement, and the specific project (outfall location).
When we combine the datasets, there are 1.2 million entries, however, this value will decrease when the arrangement of the parameters is altered to a more data-friendly arrangement.

Notable findings from your initial EDA:

There are a significant number of null values that will need to be addressed via either imputation or elimination (if appropriate). Some values appear to be out of valid range, generating Inf(inite) results in the describe() tables. Most likely the data will need to be aggregated using multiple features, such as parameter or station to create multiple series to analyze/forecast separately and/or using econometric methods. There are 12 total parameters that represent measures of ocean water characteristics (e.g., salinity, fecal matter amount, temperature, etc.). Out of the 1,231,466 samples, 0.6% would be considered an outlier based on the number of standard deviations from the mean (>+/-3 s.d.'s).
Figure 1 displays the initial time series plot for the global entero (salmonella) bacteria levels as they vary between 1990 and 2021. Note that the graph only goes through 2020 because of an issue with missing values that are affecting the scale of the axis; this will

be an additional consideration as part of data preprocessing. There is some indication of cycles and seasonality that will need to be factored into any forecasting that will be performed.

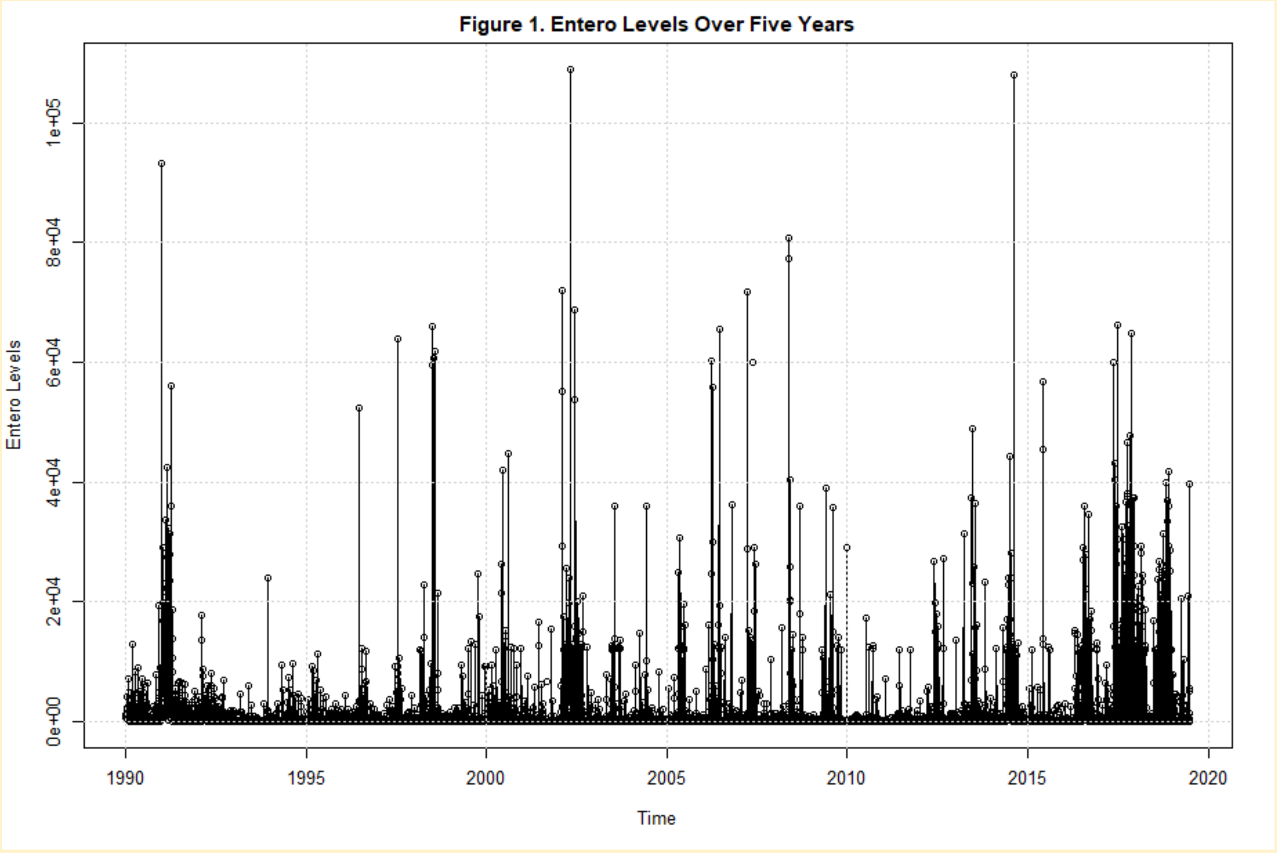
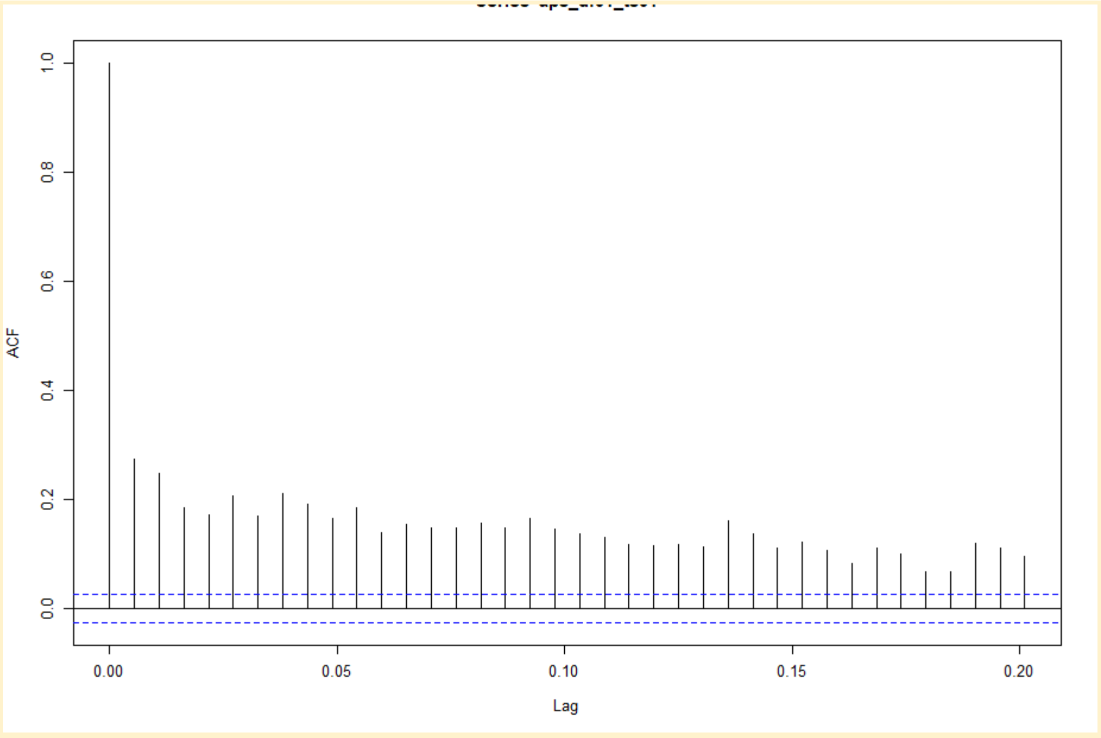


Figure 2 is the autocorrelation plot for the same entero level data and shows a trend of decreasing correlation from *t* to *t - n* timepoints.

Figure 2

Autocorrelation Plot



Github link:
https://github.com/USD-502-FinalProject/506-OceanWater-Team1

Appendix A - ADS506-01-FA22 - Final Project

Team 1

11/05/2022

RMarkdown global setup

```r
knitr::opts_chunk$set(fig.align = 'center')
```

```r
library(AppliedPredictiveModeling)
library(BioStatR)
library(car)
library(caret)
library(class)
library(corrplot)
library(datasets)
library(e1071)
library(Hmisc)
library(mlbench)
library(gridExtra)
library(psych)
library(randomForest)
library(RANN)
library(rpart)
library(rpart.plot)
library(scales)
library(tidyverse)
library(tseries)

set.seed(1699)
```

Create function to generate boxplots for continuous variables

```r
# Define function to produce formatted boxplots
box_comp <- function(xcol = c(), df = NA, rtn_met = TRUE) {
  sig <- 3
  metrics_df01 <- data.frame(metric = c("",
                                        "Total N:",
                                        "Count",
                                        "NA Count",
                                        "Mean",
                                        "Median",
                                        "Standard Deviation",
                                        "Variance",
                                        "Range",
                                        "Min",
                                      "Max",
                                      "25th Percentile",
                                      "75th Percentile",
                                      "Subset w/o Outliers:",
                                      "Count",
                                      "%",
                                      "Outlier %",
                                      "NA Count",
                                      "Mean",
                                      "Median",
                                      "Standard Deviation",
                                      "Variance",
                                      "Range",
                                      "Min",
                                      "Max"
                                      ))
for (var in xcol) {
  df_s1 <- df[, var]
  df_s1s1 <-data.frame(df_s1)
  df_s1_fit <- preProcess(df_s1s1,
                          method = c("center", "scale"))
  df_s1_trans <- predict(df_s1_fit, df_s1s1)


  # Calculate quartiles
  var_iqr_lim <- IQR(df_s1) * 1.5
  var_q1 <- quantile(df_s1, probs = c(.25))
  var_otlow <- var_q1 - var_iqr_lim
  var_q3 <- quantile(df_s1, probs = c(.75))
  var_othigh <- var_q3 + var_iqr_lim

  # Subset non-outlier data
  var_non_otlr_df01 <- subset(df, (abs(df_s1_trans) <= 3))
  #var_non_otlr_df01 <- subset(df, (df_s1 > var_otlow & df_s1 < var_othigh))
  df_s2 <- var_non_otlr_df01[, var]

  # Begin calculating measures of centrality & dispersion
  var_mean <- mean(df_s1)
  var_non_otlr_df01_trunc_mean <- mean(df_s2)
  var_med <- median(df_s1)
  var_non_otlr_df01_trunc_med <- median(df_s2)
  var_mode <- mode(df_s1)
  var_non_otlr_df01_trunc_mode <- mode(df_s2)
  var_stde <- sd(df_s1)
  var_non_otlr_df01_trunc_stde <- sd(df_s2)
  var_vari <- var(df_s1)
  var_non_otlr_df01_trunc_vari <- var(df_s2)
  var01_min <- min(df[, var])
  var01_max <- max(df[, var])
  var01_range <- var01_max - var01_min
  var02_min <- min(var_non_otlr_df01[, var])
  var02_max <- max(var_non_otlr_df01[, var])
  var02_range <- var02_max - var02_min
```

```
    # Configure y-axis min & max to sync graphs
    plot_min <- min(var01_min, var02_min)
    plot_max <- max(var01_max, var02_max)
    nonoutlier_perc <- round((as.numeric(dim(var_non_otlr_df01)[1] /
↪   as.numeric(dim(df)[1]))) * 100, 1)
    measure_val01 <- c(paste0("Variable: ", var),
                       "",
                       as.character(dim(df)[1]),
                       sum(is.na(df_s1)),
                       round(var_mean, sig),
                       round(var_med, sig),
                       round(var_stde, sig),
                       round(var_vari, sig),
                       round(var01_range, sig),
                       round(var01_min, sig),
                       round(var01_max, sig),
                       round(var_q1, sig),
                       round(var_q3, sig),
                       "",
                       as.character(dim(var_non_otlr_df01)[1]),
                       paste0(nonoutlier_perc, "%"),
                       paste0(round(100 - nonoutlier_perc, 1), "%"),
                       sum(is.na(df_s2)),
                       round(var_non_otlr_df01_trunc_mean, sig),
                       round(var_non_otlr_df01_trunc_med, sig),
                       round(var_non_otlr_df01_trunc_stde, sig),
                       round(var_non_otlr_df01_trunc_vari, sig),
                       round(var02_range, sig),
                       round(var02_min, sig),
                       round(var02_max, sig)
                       )

    var_name <- paste0("Variable: ", var)
    metrics_df01[, ncol(metrics_df01) + 1] <- measure_val01
}
  boxplot(df)
  if(rtn_met == TRUE) {
    return(metrics_df01)
  }
}
```

Importing Train/Test Datasets

```
#train_x01_df01 <- read.csv("../data/Drinking
↪  Water/analyte_tests_drinking_water_datasd.csv", header = TRUE, sep = ",")
#train_x02_df01 <- read.csv("../data/Campaign
↪  Funds/financial_support_2021_datasd_v1.csv", header = TRUE, sep = ",")
train_x03_df01a <- read.csv("../data/Ocean Water/water_quality_1990_1999_datasd.csv",
↪  header = TRUE, sep = ",")
train_x03_df01b <- read.csv("../data/Ocean Water/water_quality_2000_2010_datasd.csv",
↪  header = TRUE, sep = ",")
```

```
train_x03_df01c <- read.csv("../data/Ocean Water/water_quality_2011_2019_datasd.csv",
↪  header = TRUE, sep = ",")
train_x03_df01d <- read.csv("../data/Ocean Water/water_quality_2020_2021_datasd.csv",
↪  header = TRUE, sep = ",")

train_x03_df01 <- rbind(train_x03_df01a, train_x03_df01b, train_x03_df01c,
↪  train_x03_df01d)
#train_x01_df01 <- read.csv("../data/FD Incidents/fd_incidents_2022_datasd_v1.csv",
↪  header = TRUE, sep = ",")
#test_x01_df01 <- read.csv("../data/outlier-included/biodeg_test.csv", header = TRUE, sep
↪  = ",")

#train_y01_df01 <- read.csv("../data/outlier-included/response_train.csv", header = TRUE,
↪  sep = ",")
#test_y01_df01 <- read.csv("../data/outlier-included/response_test.csv", header = TRUE,
↪  sep = ",")

#train_y01_vc01 <- train_y01_df01[["x"]]
#test_y01_vc01 <- test_y01_df01[["x"]]

#print(head(train_x01_df01))
#describe(train_x01_df01)

#print(head(train_x02_df01))
#describe(train_x02_df01)

print(head(train_x03_df01))
```

```
##       sample station depth_m date_sample time project   parameter qualifier
## 1 9011158743      C5       9  1990-11-15         PLOO CHLOROPHYLL
## 2 9011158743      C5       9  1990-11-15         PLOO     DENSITY
## 3 9011158743      C5       9  1990-11-15         PLOO          DO
## 4 9011158743      C5       9  1990-11-15         PLOO          PH
## 5 9011158743      C5       9  1990-11-15         PLOO    SALINITY
## 6 9011158743      C5       9  1990-11-15         PLOO        TEMP
##    value   units
## 1  0.870    ug/L
## 2 23.855 sigma-t
## 3  6.550    mg/L
## 4  8.080      pH
## 5 33.617     ppt
## 6 19.430       C
```

```
describe(train_x03_df01)
```

```
## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning Inf
## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning Inf
## Warning in FUN(newX[, i], ...): no non-missing arguments to max; returning -Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to max; returning -Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to max; returning -Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to max; returning -Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to max; returning -Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to max; returning -Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to max; returning -Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to max; returning -Inf
```

```
##             vars       n   mean      sd min     max   range   se
## sample         1 1236769    NaN      NA Inf    -Inf    -Inf   NA
## station        2 1236769    NaN      NA Inf    -Inf    -Inf   NA
## depth_m        3 1152608  19.38   25.07   1     116     115 0.02
## date_sample    4 1236769    NaN      NA Inf    -Inf    -Inf   NA
## time           5 1236769    NaN      NA Inf    -Inf    -Inf   NA
## project        6 1236769    NaN      NA Inf    -Inf    -Inf   NA
## parameter      7 1236769    NaN      NA Inf    -Inf    -Inf   NA
## qualifier      8 1236769    NaN      NA Inf    -Inf    -Inf   NA
## value          9 1231466 124.24 1785.21 -37 1100000 1100037 1.61
## units         10 1236769    NaN      NA Inf    -Inf    -Inf   NA
```

```
train_x03_df01_ss <- train_x03_df01 %>%
  group_by(station, date_sample) %>%
  summarise(Count = n())
```

```
## `summarise()` has grouped output by 'station'. You can override using the
## `.groups` argument.
```

```
train_x03_df01_ay <- train_x03_df01 %>%
  group_by(parameter) %>%
  summarise(Count = n())

train_x03_df01_date <- train_x03_df01 %>%
  group_by(date_sample) %>%
  summarise(Count = n())

train_x03_df01_full <- train_x03_df01 %>%
  group_by(date_sample, parameter) %>%
  summarise(Total = sum(value))
```

```
## `summarise()` has grouped output by 'date_sample'. You can override using the
## `.groups` argument.
```

```
print(head(train_x03_df01_ss))
```

```
## # A tibble: 6 x 3
## # Groups:   station [1]
##   station date_sample Count
##   <chr>   <chr>       <int>
## 1 A1      1991-01-02     25
## 2 A1      1991-01-03     25
## 3 A1      1991-01-07     24
## 4 A1      1991-01-09     70
## 5 A1      1991-01-14     25
## 6 A1      1991-01-16     25
```

```
print(train_x03_df01_ay)
```

```
## # A tibble: 12 x 2
##    parameter    Count
##    <chr>        <int>
##  1 CHLOROPHYLL  88471
##  2 DENSITY      88317
##  3 DO          109542
##  4 ENTERO      144341
##  5 FECAL       137649
##  6 OG            7944
##  7 PH          107818
##  8 SALINITY    109492
##  9 SUSO         27543
## 10 TEMP        139066
## 11 TOTAL       137584
## 12 XMS         139002
```

```
print(head(train_x03_df01_date))
```

```
## # A tibble: 6 x 2
##   date_sample Count
##   <chr>       <int>
## 1 1990-11-15     14
## 2 1991-01-02    195
## 3 1991-01-03    195
## 4 1991-01-07    190
## 5 1991-01-08    181
## 6 1991-01-09    577
```

```
print(head(train_x03_df01_full))
```

```
## # A tibble: 6 x 3
## # Groups:   date_sample [1]
##   date_sample parameter   Total
##   <chr>       <chr>       <dbl>
## 1 1990-11-15  CHLOROPHYLL  2.14
## 2 1990-11-15  DENSITY     47.7
## 3 1990-11-15  DO          14.0
## 4 1990-11-15  PH          16.3
## 5 1990-11-15  SALINITY    67.2
## 6 1990-11-15  TEMP        38.7
```

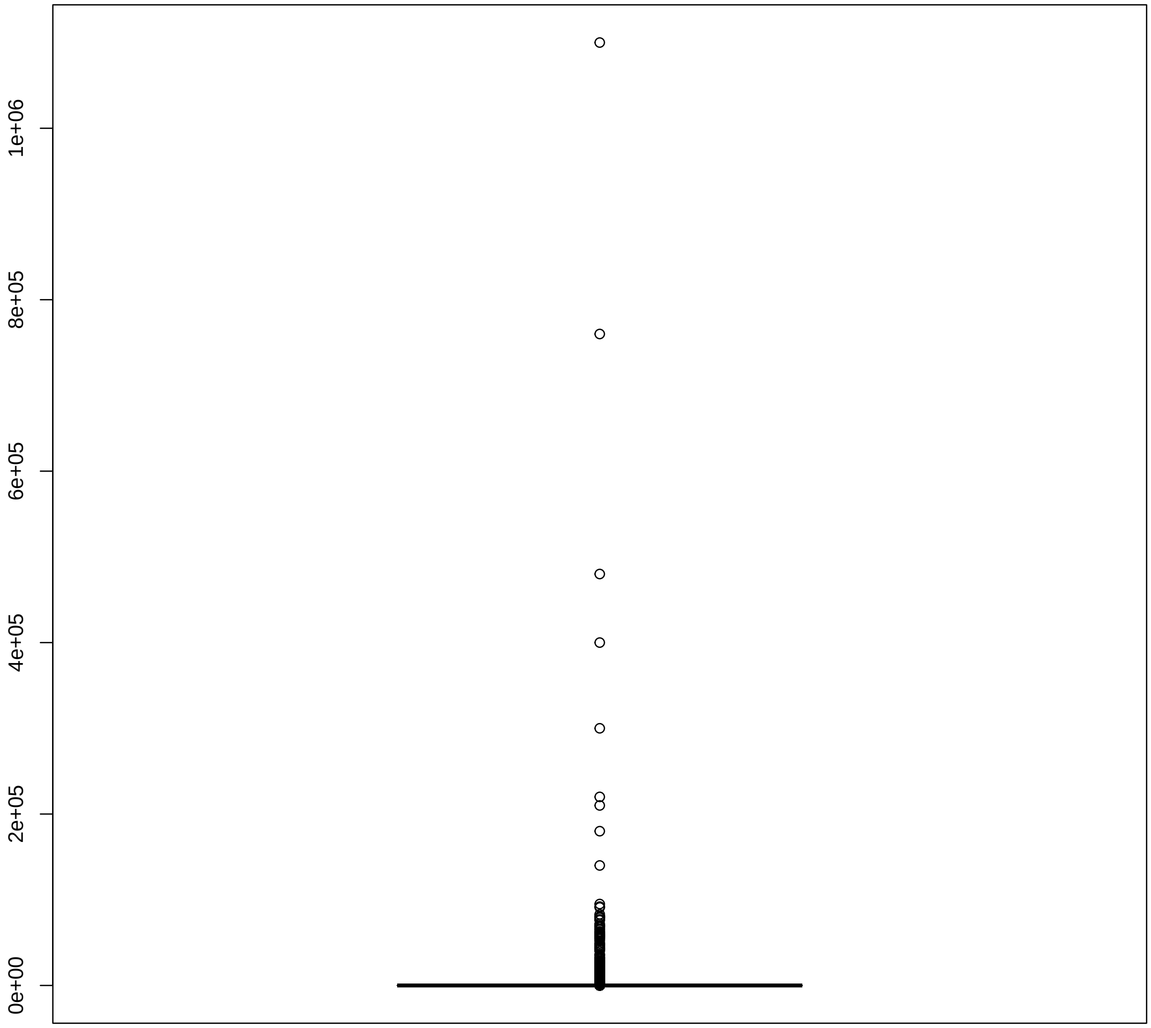
Run function to create comparative boxplots

```r
x01_lst01 <- c()

x01_lst02 <- c("analyte_value")
x02_lst02 <- c("contribution_amount",
               "contribution_annual")
x03_lst02 <- c("value")

x01_lst03 <- c()

x01_lst04 <- c()

x01_lst05 <- c()

x01_lst06 <- c()

x01_lst07 <- c()

x01_lst08 <- c()

x01_lst09 <- c()

x01_lst10 <- c()

x01_lst11 <- c()

x01_lst12 <- c()

x01_lst13 <- c()

x01_lst14 <- c()

x01_lst15 <- c()

x01_lst16 <- c()

#train_x01_df01_cols01 <- colnames(train_x01_df01)
#print(train_x01_df01_cols01)
#train_x01_df01_metrics <- box_comp(xcol = train_x01_df01_cols01, df = train_x01_df01)
#train_x01_df01_metrics
#write.csv(train_x01_df01_metrics, "../outputs/demos.csv", row.names = FALSE)

#train_x01_df03 <- subset(x = train_x01_df01, select = x01_lst02)
#train_x01_df03 <- na.omit(train_x01_df03)
#print(head(train_x01_df03))
#box_comp(xcol = x01_lst02, df = subset(x = train_x01_df03, select = x01_lst02), rtn_met
↪  = TRUE)

#train_x02_df03 <- subset(x = train_x02_df01, select = x02_lst02)
#train_x02_df03 <- na.omit(train_x02_df03)
#print(head(train_x02_df03))
#box_comp(xcol = x02_lst02, df = subset(x = train_x02_df03, select = x02_lst02), rtn_met
↪  = TRUE)
```

```
train_x03_df03 <- subset(x = train_x03_df01, select = x03_lst02)
train_x03_df03 <- na.omit(train_x03_df03)
print(head(train_x03_df03))
```

```
##    value
## 1  0.870
## 2 23.855
## 3  6.550
## 4  8.080
## 5 33.617
## 6 19.430
```

```
box_comp(xcol = x03_lst02, df = subset(x = train_x03_df03, select = x03_lst02), rtn_met =
↪  TRUE)
```



```
##                   metric               V2
## 1                         Variable: value
## 2               Total N:                 
## 3                  Count          1231466
## 4               NA Count                0
## 5                   Mean           124.24
## 6                 Median            8.343
## 7     Standard Deviation         1785.206
## 8               Variance      3186959.838
## 9                  Range          1100037
## 10                   Min              -37
## 11                   Max          1100000
## 12       25th Percentile                2
## 13       75th Percentile           33.214
## 14 Subset w/o Outliers:
## 15                Count         1223995
## 16                    %           99.4%
## 17            Outlier %            0.6%
## 18             NA Count               0
## 19                 Mean          43.899
## 20               Median           8.298
## 21   Standard Deviation         236.208
## 22             Variance       55794.454
## 23                Range            5437
## 24                  Min             -37
## 25                  Max            5400
```

```
print(head(train_x03_df01_full))
```

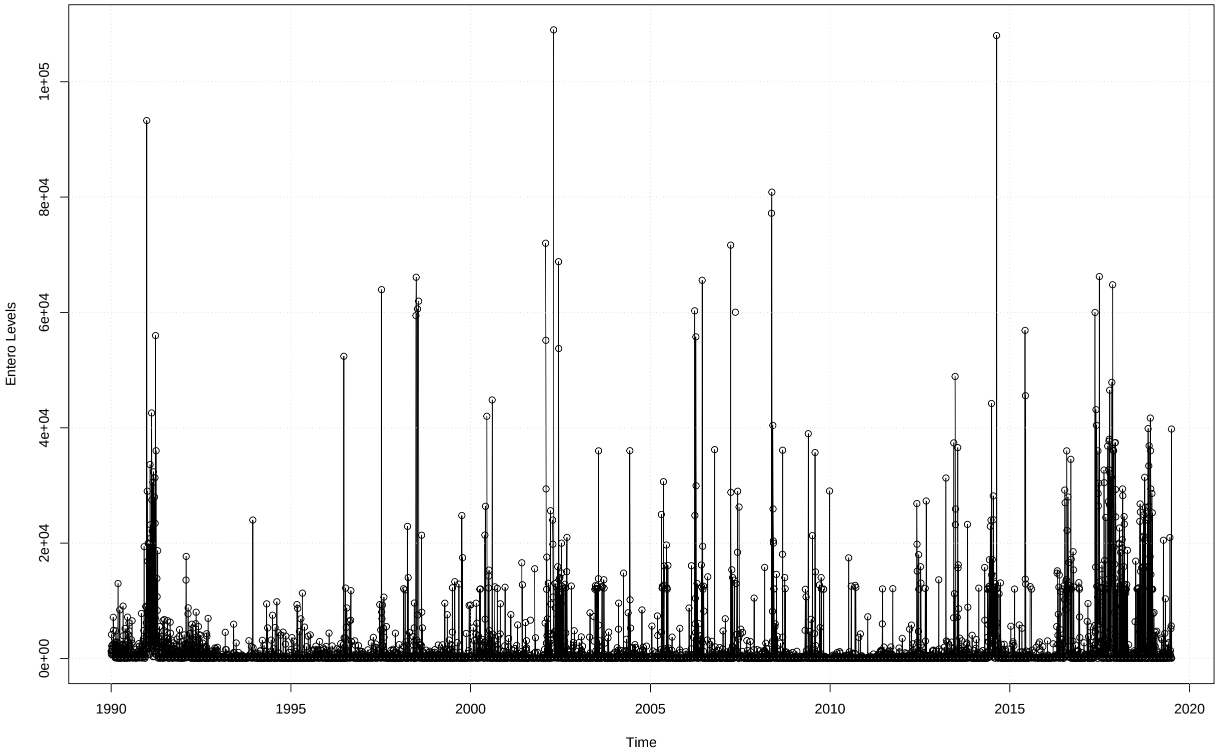
```
## # A tibble: 6 x 3
## # Groups:   date_sample [1]
##   date_sample parameter   Total
##   <chr>       <chr>       <dbl>
## 1 1990-11-15  CHLOROPHYLL  2.14
## 2 1990-11-15  DENSITY     47.7
## 3 1990-11-15  DO          14.0
## 4 1990-11-15  PH          16.3
## 5 1990-11-15  SALINITY    67.2
## 6 1990-11-15  TEMP        38.7
```

```
print(tail(train_x03_df01_full))
```

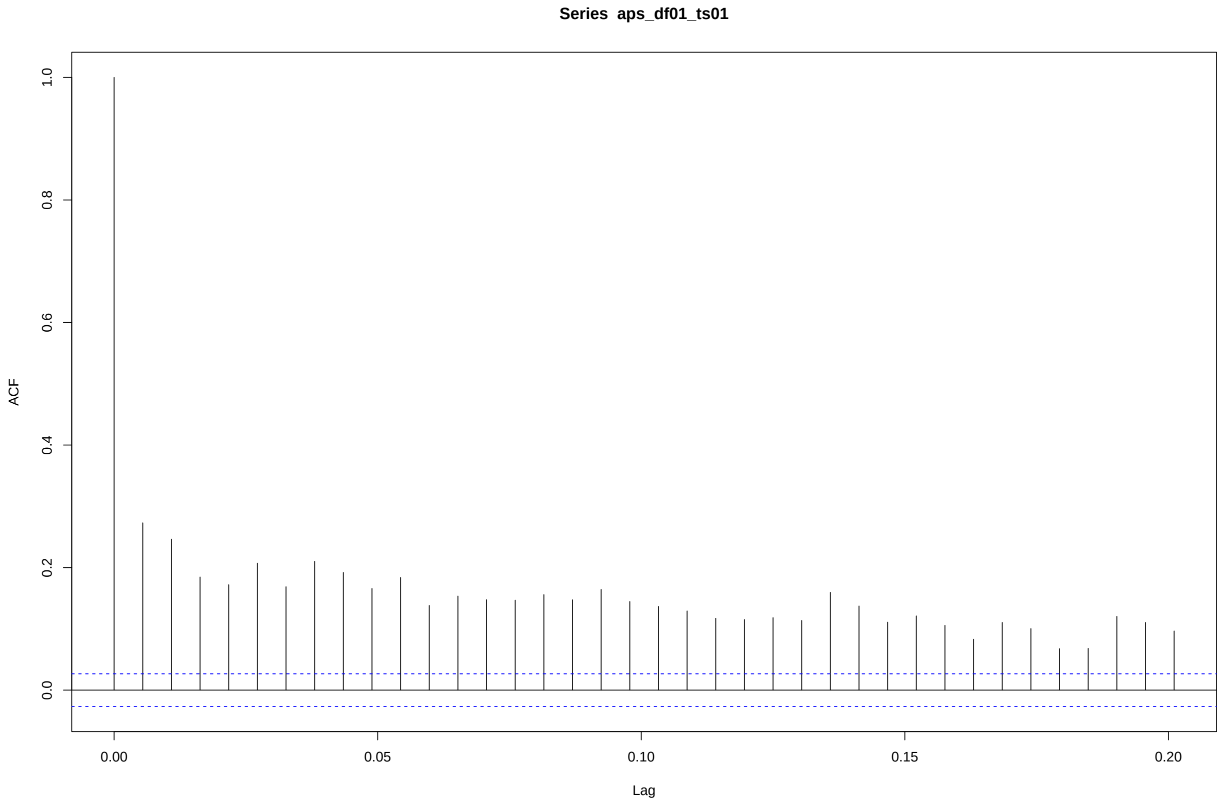
```
## # A tibble: 6 x 3
## # Groups:   date_sample [2]
##   date_sample parameter Total
##   <chr>       <chr>     <dbl>
## 1 2021-12-28  ENTERO    39780
## 2 2021-12-28  FECAL     49256
## 3 2021-12-28  TOTAL     83200
## 4 2021-12-29  ENTERO       36
## 5 2021-12-29  FECAL        40
## 6 2021-12-29  TOTAL       562
```

```
train_x03_df01_full02 <- train_x03_df01_full[train_x03_df01_full$parameter == "ENTERO", ]
# & train_x03_df01_full$station == "A1"
aps_df01_ts01 <- ts(train_x03_df01_full02$Total, start = c(1990, 1), freq = 184)
#, start = c(2020, 1), freq = 52
#print(aps_df01_ts01)

#ship_fore_avg <- tslm(aps_df01_ts01 ~ trend)
#ship_fore_trnd <- tslm(aps_df01_ts01 ~ trend + I(trend^2))

plot(aps_df01_ts01,
     xlab = "Time",
     ylab = "Entero Levels",
     type = "o",
     main = "Figure 1. Entero Levels Over Five Years")
grid()
```

Figure 1. Entero Levels Over Five Years



```
print(acf(aps_df01_ts01, pl=TRUE, na.action = na.pass))
```



```
##
## Autocorrelations of series 'aps_df01_ts01', by lag
##
## 0.00000 0.00543 0.01087 0.01630 0.02174 0.02717 0.03261 0.03804 0.04348 0.04891
##   1.000   0.273   0.246   0.185   0.172   0.207   0.169   0.210   0.192   0.166
## 0.05435 0.05978 0.06522 0.07065 0.07609 0.08152 0.08696 0.09239 0.09783 0.10326
##   0.184   0.138   0.154   0.148   0.147   0.156   0.147   0.164   0.145   0.137
## 0.10870 0.11413 0.11957 0.12500 0.13043 0.13587 0.14130 0.14674 0.15217 0.15761
##   0.129   0.117   0.115   0.118   0.114   0.160   0.137   0.111   0.121   0.106
## 0.16304 0.16848 0.17391 0.17935 0.18478 0.19022 0.19565 0.20109
##   0.083   0.110   0.100   0.068   0.068   0.120   0.110   0.096
```