Forecasting San Diego Power Consumption

Hunter Blum, Mackenzie Carter, Saba Alemayehu

2022-11-06

Github: https://github.com/mc3435/ADS_506_Group2 (https://github.com/mc3435/ADS_506_Group2)

Libraries

```
library(readxl)
library(rstudioapi)
library(parsedate)
library(lubridate)
library(psych)
library(corrplot)
library(outliers)
library(ggpmisc)
library(gridExtra)
library(zoo)
library(forecast)

# Note - Tidyverse is a collection of packages, see the Attaching packages section below. Usuall
y best to load last so its functions will mask over other packages.
library(tidyverse)
```

Set up cores for models - You may need to install/update Java.

```
library(parallel)
library(doParallel)

cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)
```

Section 1 - EDA/Cleaning

Read in data (So you can skip first part)

```
setwd(dirname(getActiveDocumentContext()$path))
SD <- read.csv("SD.csv")
SD <- SD[,-1]
SD <- SD %>% select(Date, SDGE, starts_with("Hourly"))
head(SD)
```

```
##                  Date     SDGE HourlyAltimeterSetting HourlyDewPointTemperature
## 1 2019-01-01 01:00:00 2088.454                 29.98                         40
## 2 2019-01-01 02:00:00 1998.915                 29.98                         35
## 3 2019-01-01 03:00:00 1934.057                 29.99                         30
## 4 2019-01-01 04:00:00 1891.809                 30.01                         29
## 5 2019-01-01 05:00:00 1898.795                 30.04                         24
## 6 2019-01-01 06:00:00 1969.965                 30.08                         26
##   HourlyDryBulbTemperature HourlyPrecipitation HourlyPresentWeatherType
## 1                       47                0.00
## 2                       47                0.00
## 3                       47                0.00
## 4                       46                0.00
## 5                       48                0.00
## 6                       45                0.00
##   HourlyPressureChange HourlyPressureTendency HourlyRelativeHumidity
## 1                   NA                     NA                     77
## 2                   NA                     NA                     63
## 3                -0.01                      1                     52
## 4                   NA                     NA                     51
## 5                   NA                     NA                     39
## 6                -0.09                      3                     48
##   HourlySeaLevelPressure HourlySkyConditions HourlyStationPressure
## 1                  29.98          FEW:02 100                 29.95
## 2                  29.98          FEW:02 100                 29.95
## 3                  29.98          FEW:02 100                 29.96
## 4                  30.00          FEW:02 100                 29.98
## 5                  30.03          FEW:02 100                 30.01
## 6                  30.08          FEW:02 100                 30.05
##   HourlyVisibility HourlyWetBulbTemperature HourlyWindDirection
## 1            10.00                       44                 320
## 2            10.00                       42                 030
## 3            10.00                       40                 020
## 4            10.00                       39                 040
## 5            10.00                       38                 050
## 6            10.00                       37                 020
##   HourlyWindGustSpeed HourlyWindSpeed
## 1                  NA               5
## 2                  NA               6
## 3                  NA               6
## 4                  NA               5
## 5                  NA               8
## 6                  NA               8
```

Structual and General EDA

First, lets take a broad look at our data, including variable types, descriptive statistics and NA counts.

```
str(SD)
```

```
## 'data.frame':    39487 obs. of  18 variables:
##  $ Date                     : chr  "2019-01-01 01:00:00" "2019-01-01 02:00:00" "2019-01-01 0
3:00:00" "2019-01-01 04:00:00" ...
##  $ SDGE                     : num  2088 1999 1934 1892 1899 ...
##  $ HourlyAltimeterSetting   : num  30 30 30 30 30 ...
##  $ HourlyDewPointTemperature: chr  "40" "35" "30" "29" ...
##  $ HourlyDryBulbTemperature : chr  "47" "47" "47" "46" ...
##  $ HourlyPrecipitation      : chr  "0.00" "0.00" "0.00" "0.00" ...
##  $ HourlyPresentWeatherType : chr  "" "" "" "" ...
##  $ HourlyPressureChange     : num  NA NA -0.01 NA NA -0.09 NA NA -0.07 NA ...
##  $ HourlyPressureTendency   : int  NA NA 1 NA NA 3 NA NA 1 NA ...
##  $ HourlyRelativeHumidity   : int  77 63 52 51 39 48 41 30 22 20 ...
##  $ HourlySeaLevelPressure   : num  30 30 30 30 30 ...
##  $ HourlySkyConditions      : chr  "FEW:02 100" "FEW:02 100" "FEW:02 100" "FEW:02 100" ...
##  $ HourlyStationPressure    : num  29.9 29.9 30 30 30 ...
##  $ HourlyVisibility         : chr  "10.00" "10.00" "10.00" "10.00" ...
##  $ HourlyWetBulbTemperature : int  44 42 40 39 38 37 38 40 40 41 ...
##  $ HourlyWindDirection      : chr  "320" "030" "020" "040" ...
##  $ HourlyWindGustSpeed      : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ HourlyWindSpeed          : int  5 6 6 5 8 8 7 8 8 7 ...
```

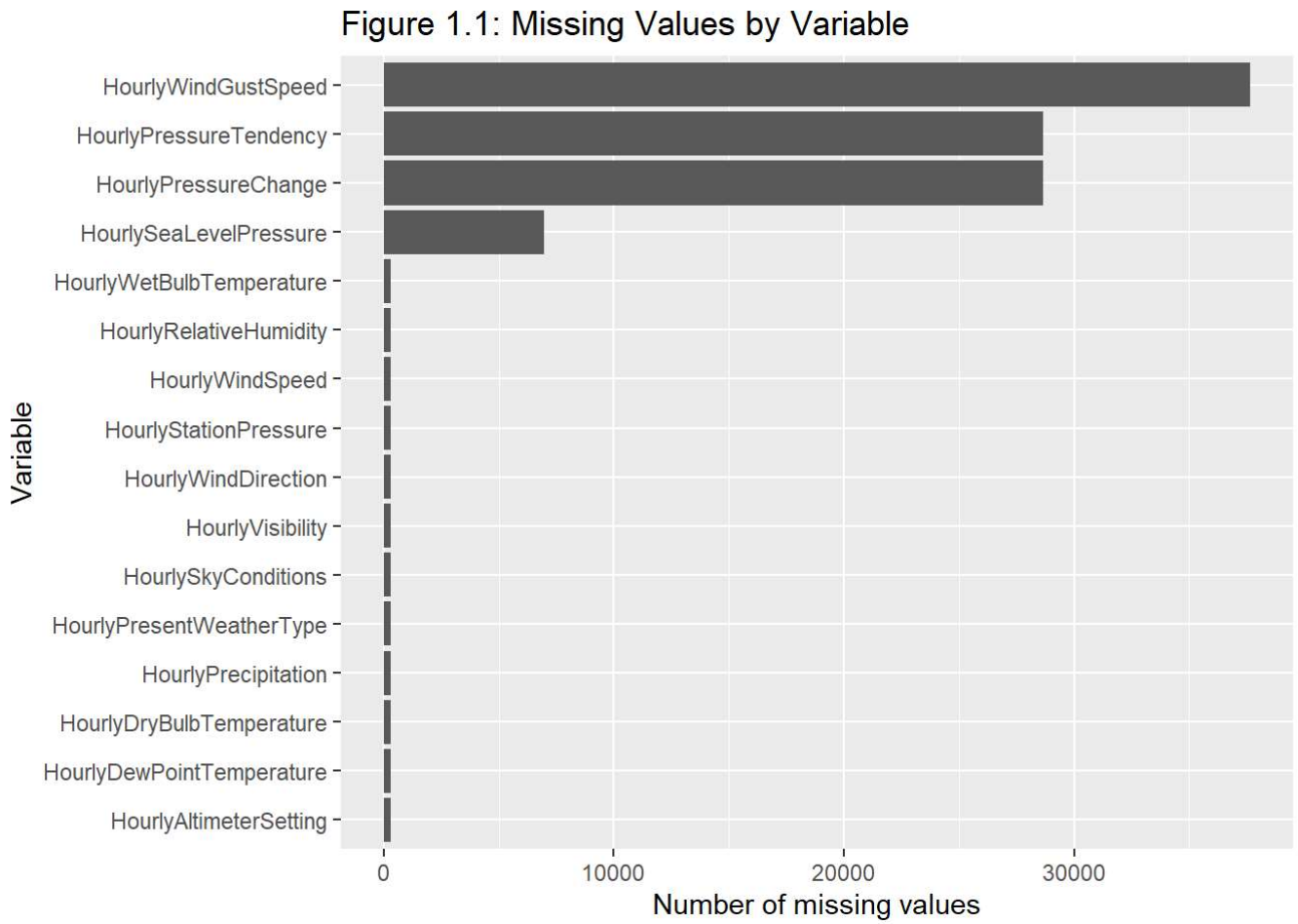
```
describe(SD)
```

```
##                           vars     n     mean      sd   median  trimmed
## Date*                        1 39487 16476.85 9477.72 16518.00 16494.31
## SDGE                         2 39487  2177.70  439.83  2109.82  2139.64
## HourlyAltimeterSetting       3 39184    29.99    0.11    29.98    29.99
## HourlyDewPointTemperature*   4 39184    61.16   14.48    63.00    62.37
## HourlyDryBulbTemperature*    5 39184    29.01    8.70    28.00    28.65
## HourlyPrecipitation*         6 39184     3.93   10.07     2.00     1.94
## HourlyPresentWeatherType*    7 39184     3.20    6.65     1.00     1.12
## HourlyPressureChange         8 10818     0.00    0.03     0.00     0.00
## HourlyPressureTendency       9 10818     4.29    2.77     3.00     4.35
## HourlyRelativeHumidity      10 39169    70.79   15.79    73.00    72.40
## HourlySeaLevelPressure      11 32531    29.99    0.11    29.98    29.99
## HourlySkyConditions*        12 39184  3664.45 2377.03  3710.50  3619.10
## HourlyStationPressure       13 39183    29.96    0.11    29.95    29.96
## HourlyVisibility*           14 39184    18.75    4.62    17.00    17.92
## HourlyWetBulbTemperature    15 39168    58.48    6.77    58.00    58.76
## HourlyWindDirection*        16 39184    21.99   13.22    25.00    22.29
## HourlyWindGustSpeed         17  1847    21.71    4.58    21.00    20.95
## HourlyWindSpeed             18 39182     5.37    3.98     5.00     5.14
##                              mad    min      max    range  skew kurtosis
## Date*                    12189.94   1.00 32852.00 32851.00 -0.01    -1.20
## SDGE                       395.69 840.18  4651.74  3811.57  1.01     1.87
## HourlyAltimeterSetting       0.10  29.50    30.39     0.89  0.23     0.35
## HourlyDewPointTemperature*  13.34   1.00    92.00    91.00 -0.73     0.35
## HourlyDryBulbTemperature*    7.41   1.00    72.00    71.00  0.50     0.78
## HourlyPrecipitation*         0.00   1.00    57.00    56.00  4.93    22.81
## HourlyPresentWeatherType*    0.00   1.00    47.00    46.00  3.01     7.91
## HourlyPressureChange         0.03  -0.11     0.13     0.24  0.27     0.21
## HourlyPressureTendency       2.97   0.00     8.00     8.00  0.00    -1.36
## HourlyRelativeHumidity      13.34   6.00   100.00    94.00 -1.09     1.59
## HourlySeaLevelPressure       0.10  29.50    30.39     0.89  0.27     0.23
## HourlySkyConditions*      3183.88   1.00  8143.00  8142.00  0.08    -1.36
## HourlyStationPressure        0.10  29.47    30.36     0.89  0.23     0.35
## HourlyVisibility*            0.00   1.00    30.00    29.00  1.08     1.92
## HourlyWetBulbTemperature     7.41  32.00    78.00    46.00 -0.39     0.01
## HourlyWindDirection*        11.86   1.00    40.00    39.00 -0.44    -1.24
## HourlyWindGustSpeed          2.97  16.00    46.00    30.00  1.91     4.64
## HourlyWindSpeed              4.45   0.00    33.00    33.00  0.55     0.98
##                              se
## Date*                     47.70
## SDGE                       2.21
## HourlyAltimeterSetting     0.00
## HourlyDewPointTemperature* 0.07
## HourlyDryBulbTemperature*  0.04
## HourlyPrecipitation*       0.05
## HourlyPresentWeatherType*  0.03
## HourlyPressureChange       0.00
## HourlyPressureTendency     0.03
## HourlyRelativeHumidity     0.08
## HourlySeaLevelPressure     0.00
## HourlySkyConditions*      12.01
## HourlyStationPressure      0.00
```

```
## HourlyVisibility*           0.02
## HourlyWetBulbTemperature    0.03
## HourlyWindDirection*        0.07
## HourlyWindGustSpeed         0.11
## HourlyWindSpeed             0.02
```

Evaluate missing values by column

```
missing.values <- SD %>%
    gather(key = "key", value = "val") %>%
    mutate(is.missing = is.na(val)) %>%
    group_by(key, is.missing) %>%
    summarise(num.missing = n()) %>%
    filter(is.missing==T) %>%
    select(-is.missing) %>%
    arrange(desc(num.missing))

missing.values %>%
  ggplot() +
    geom_bar(aes(x=reorder(key, num.missing), y=num.missing), stat = 'identity') +
    labs(x='Variable', y="Number of missing values", title='Figure 1.1: Missing Values by Variab
le') +
    coord_flip()
```



Evaluate NAs by observation - 304 observations are missing all weather data, we will either need to impute or stick with regression models.

```
NA_row <- rowSums(is.na(SD))
obs <- seq(1:nrow(SD))

NA_row <- data.frame(obs, NA_row)
NA_row %>% filter(NA_row == 16) %>% tally()
```

```
##     n
## 1 303
```

Properly format variables

```
# Numeric
SD$HourlyDryBulbTemperature <- as.numeric(as.character(SD$HourlyDryBulbTemperature))
SD$HourlyDewPointTemperature <- as.numeric(as.character(SD$HourlyDewPointTemperature))
SD$HourlyPrecipitation <- as.numeric(as.character(SD$HourlyPrecipitation))
SD$HourlyVisibility <- as.numeric(SD$HourlyVisibility)
SD$HourlyWindDirection <- as.numeric(SD$HourlyWindDirection)

# Factors
SD$HourlyPresentWeatherType <- as.factor(SD$HourlyPresentWeatherType)
SD$HourlySkyConditions <- as.factor(SD$HourlySkyConditions)

# Only run this if you loaded in SD data from EDA
SD$Date <- as_datetime(SD$Date)
```

Imputation

Since we're doing a time-based imputation, we won't separate training/testing data before imputing in case the first validation observation is missing values.

```
# NAs in wind gusts should be zero
SD <- SD %>% mutate(HourlyWindGustSpeed = ifelse(is.na(HourlyWindGustSpeed), 0, HourlyWindGustSp
eed))

# Fill the rest of NAs with the last non-NA value
SD <- na.locf(SD)
```

Numeric EDA

```
num <- SD %>% dplyr::select(where(is.numeric))
describe(num)
```

```
##                           vars     n    mean     sd  median trimmed    mad
## SDGE                         1 39485 2177.71 439.84 2109.87 2139.65 395.75
## HourlyAltimeterSetting       2 39485   29.99   0.11   29.98   29.99   0.10
## HourlyDewPointTemperature    3 39485   53.63   9.22   55.00   54.59   7.41
## HourlyDryBulbTemperature     4 39485   64.31   7.57   64.00   64.32   7.41
## HourlyPrecipitation          5 39485    0.00   0.02    0.00    0.00   0.00
## HourlyPressureChange         6 39485    0.00   0.03    0.00    0.00   0.03
## HourlyPressureTendency       7 39485    4.28   2.76    3.00    4.34   2.97
## HourlyRelativeHumidity       8 39485   70.72  15.86   73.00   72.35  13.34
## HourlySeaLevelPressure       9 39485   29.99   0.11   29.98   29.99   0.10
## HourlyStationPressure       10 39485   29.96   0.11   29.95   29.96   0.10
## HourlyVisibility            11 39485    9.11   2.13   10.00    9.72   0.00
## HourlyWetBulbTemperature    12 39485   58.45   6.78   58.00   58.74   7.41
## HourlyWindDirection         13 39485  185.02 124.86  220.00  189.20 118.61
## HourlyWindGustSpeed         14 39485    1.02   4.69    0.00    0.00   0.00
## HourlyWindSpeed             15 39485    5.37   3.99    5.00    5.13   4.45
##                             min     max   range  skew kurtosis   se
## SDGE                     840.18 4651.74 3811.57  1.01     1.87 2.21
## HourlyAltimeterSetting    29.50   30.39    0.89  0.25     0.37 0.00
## HourlyDewPointTemperature  6.00   74.00   68.00 -1.10     1.72 0.05
## HourlyDryBulbTemperature  38.00   97.00   59.00  0.03     0.21 0.04
## HourlyPrecipitation        0.00    0.51    0.51 13.73   252.55 0.00
## HourlyPressureChange      -0.11    0.13    0.24  0.27     0.25 0.00
## HourlyPressureTendency     0.00    8.00    8.00  0.01    -1.35 0.01
## HourlyRelativeHumidity     6.00  100.00   94.00 -1.09     1.60 0.08
## HourlySeaLevelPressure    29.50   30.39    0.89  0.25     0.39 0.00
## HourlyStationPressure     29.47   30.36    0.89  0.25     0.37 0.00
## HourlyVisibility           0.00   10.00   10.00 -2.62     6.07 0.01
## HourlyWetBulbTemperature  32.00   78.00   46.00 -0.39     0.01 0.03
## HourlyWindDirection        0.00  360.00  360.00 -0.46    -1.34 0.63
## HourlyWindGustSpeed        0.00   46.00   46.00  4.68    21.72 0.02
## HourlyWindSpeed            0.00   33.00   33.00  0.55     0.98 0.02
```

```
summary(num)
```

```
##       SDGE        HourlyAltimeterSetting HourlyDewPointTemperature
##  Min.   : 840.2   Min.   :29.50          Min.   : 6.00
##  1st Qu.:1866.5   1st Qu.:29.92          1st Qu.:49.00
##  Median :2109.9   Median :29.98          Median :55.00
##  Mean   :2177.7   Mean   :29.99          Mean   :53.63
##  3rd Qu.:2416.1   3rd Qu.:30.06          3rd Qu.:60.00
##  Max.   :4651.7   Max.   :30.39          Max.   :74.00
##  HourlyDryBulbTemperature HourlyPrecipitation HourlyPressureChange
##  Min.   :38.00            Min.   :0.000000    Min.   :-0.1100000
##  1st Qu.:59.00            1st Qu.:0.000000    1st Qu.:-0.0200000
##  Median :64.00            Median :0.000000    Median : 0.0000000
##  Mean   :64.31            Mean   :0.002048    Mean   :-0.0001889
##  3rd Qu.:69.00            3rd Qu.:0.000000    3rd Qu.: 0.0200000
##  Max.   :97.00            Max.   :0.510000    Max.   : 0.1300000
##  HourlyPressureTendency HourlyRelativeHumidity HourlySeaLevelPressure
##  Min.   :0.000          Min.   :  6.00         Min.   :29.50
##  1st Qu.:2.000          1st Qu.: 63.00         1st Qu.:29.92
##  Median :3.000          Median : 73.00         Median :29.98
##  Mean   :4.284          Mean   : 70.72         Mean   :29.99
##  3rd Qu.:7.000          3rd Qu.: 81.00         3rd Qu.:30.06
##  Max.   :8.000          Max.   :100.00         Max.   :30.39
##  HourlyStationPressure HourlyVisibility HourlyWetBulbTemperature
##  Min.   :29.47         Min.   : 0.000   Min.   :32.00
##  1st Qu.:29.89         1st Qu.:10.000   1st Qu.:54.00
##  Median :29.95         Median :10.000   Median :58.00
##  Mean   :29.96         Mean   : 9.105   Mean   :58.45
##  3rd Qu.:30.03         3rd Qu.:10.000   3rd Qu.:64.00
##  Max.   :30.36         Max.   :10.000   Max.   :78.00
##  HourlyWindDirection HourlyWindGustSpeed HourlyWindSpeed
##  Min.   :  0         Min.   : 0.000      Min.   : 0.000
##  1st Qu.: 20         1st Qu.: 0.000      1st Qu.: 3.000
##  Median :220         Median : 0.000      Median : 5.000
##  Mean   :185         Mean   : 1.015      Mean   : 5.367
##  3rd Qu.:300         3rd Qu.: 0.000      3rd Qu.: 8.000
##  Max.   :360         Max.   :46.000      Max.   :33.000
```

Outliers

```
tests = lapply(num, grubbs.test)
tests$HourlyWindSpeed
```

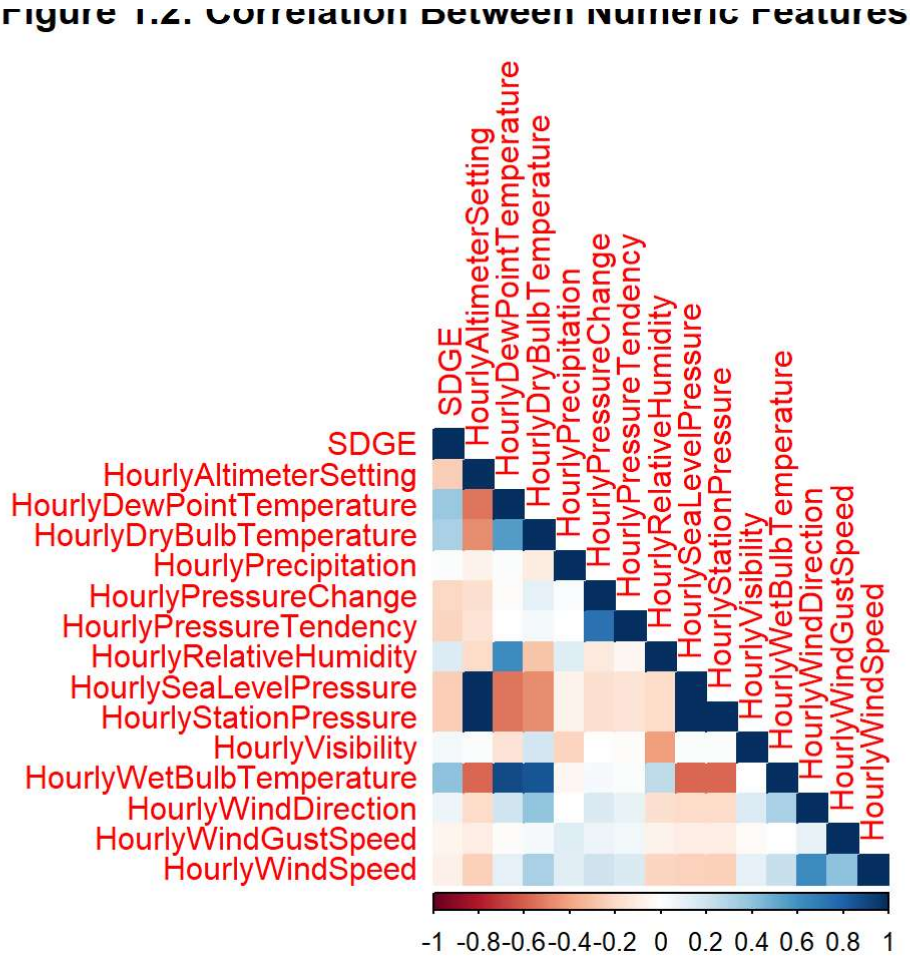
```
##
##  Grubbs test for one outlier
##
## data:  X[[i]]
## G = 6.92792, U = 0.99878, p-value = 8.309e-08
## alternative hypothesis: highest value 33 is an outlier
```

The grubbs test shows our max value for wind speed, 33, is an outlier. This makes sense because wind speed ranges in the single digits- it is likely that 33 was a typo.

Not sure 33 mph winds is a typo, could easily happen in a storm.

Correlation

```
M = cor(num)
corrplot(M,, main = "Figure 1.2: Correlation Between Numeric Features", type = "lower", method =
"color")
```

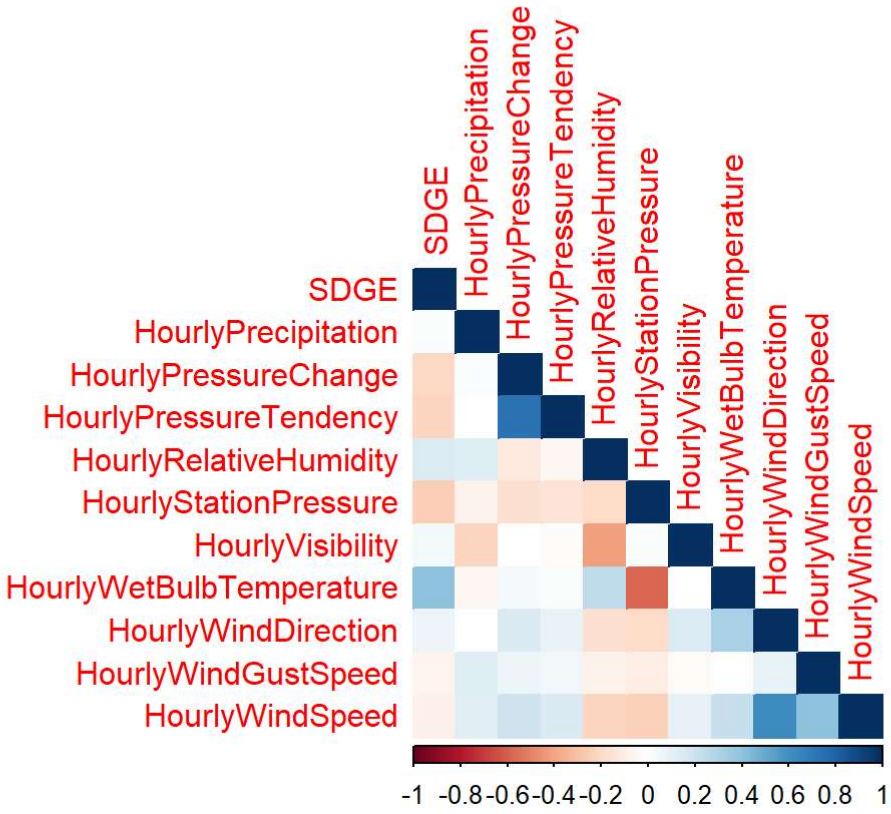


As we can see from our corrplot, station pressure, altimeter setting, & sea level pressure are 100% correlated. Hourly dry bulb, wet bulb, and dew point temperatures are closely correlated as well. We will be keeping wet bulb temperature as that is analogous to what humans feel. Altimeter setting is another measure of pressure, so we will be keeping hourly station pressure for ease of understanding.

Correlations below look much better

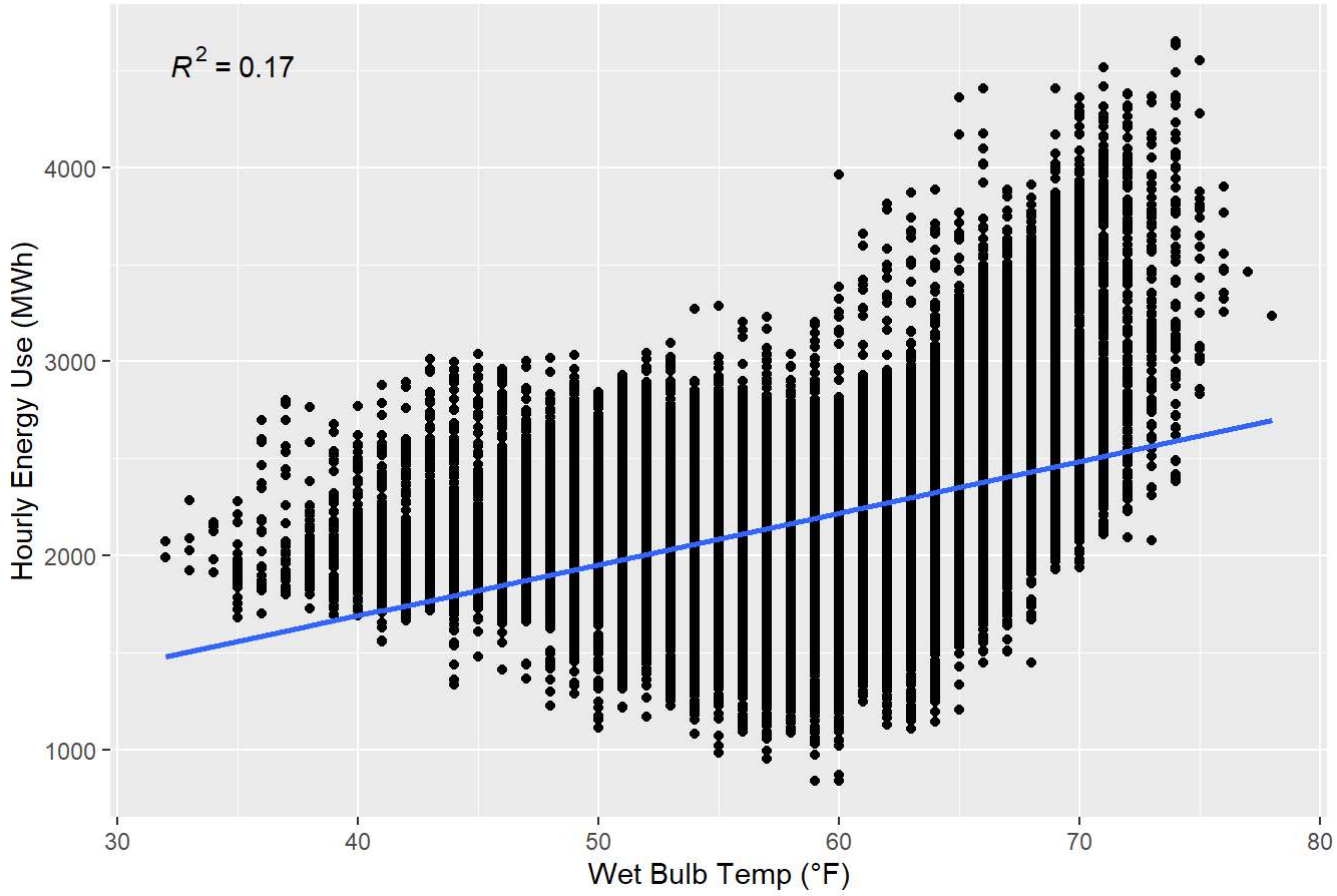
```
drop2 <- c("HourlyDryBulbTemperature","HourlyAltimeterSetting", "HourlyDewPointTemperature", "Ho
urlySeaLevelPressure")
num2 = num[,!(names(num) %in% drop2)]
M2 = cor(num2)
corrplot(M2,  type = "lower", main = "Figure 1.3: Correlations After Removal of Highly-Correlate
d Variables" , method = "color")
```

Figure 1.3: Correlations After Removal of Highly-Correlated Variables




```
ggplot(data = num2, aes(x= HourlyWetBulbTemperature, y=SDGE)) +
  geom_point() +
  stat_poly_line() + stat_poly_eq() +
  labs(title="Figure 1.4: SDGE Usage By Wet Bulb Temperature",
         x="Wet Bulb Temp (°F)", y = "Hourly Energy Use (MWh)")
```

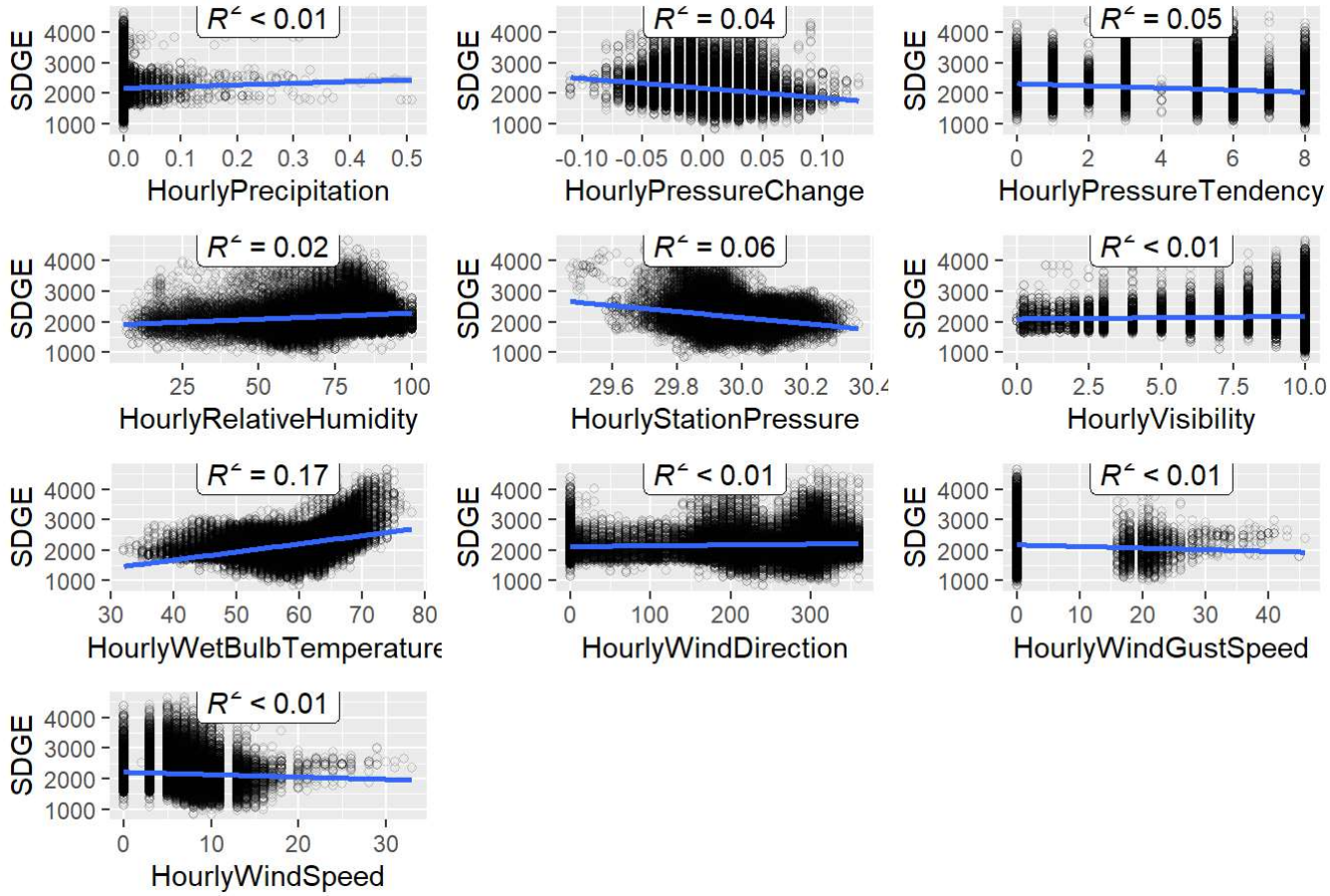
Figure 1.4: SDGE Usage By Wet Bulb Temperature



Scatter plots for numeric features compared to hourly energy use - may need to explore non-linear options.

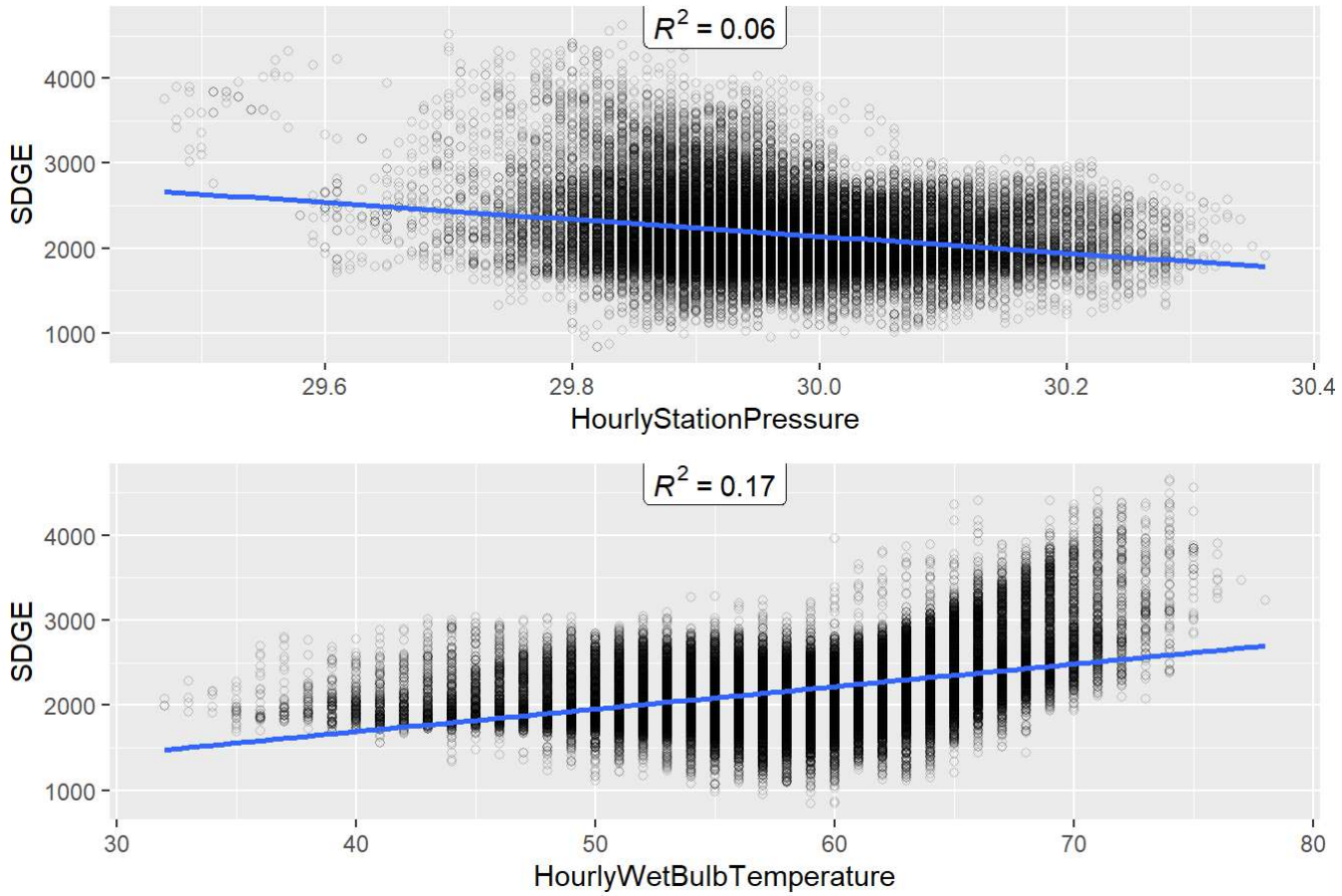
```
ScatPlotter.SD <- function(numvar){
  ggplot(num2, aes(x = num2[,numvar], y = SDGE)) +
    geom_point(shape = 1 , alpha = 0.2) +
    stat_poly_line() + stat_poly_eq(geom = "label", label.x = "middle") +
    xlab(colnames(num2[numvar]))
}

num_cols <- c(2:11)
Scatterplots <- lapply(num_cols, ScatPlotter.SD)
grid.arrange(grobs = Scatterplots, top = "Figure 1.5: Scatterplots for Each Numeric Predictor v
s. SDGE")
```

Figure 1.5: Scatterplots for Each Numeric Predictor vs. SDGE



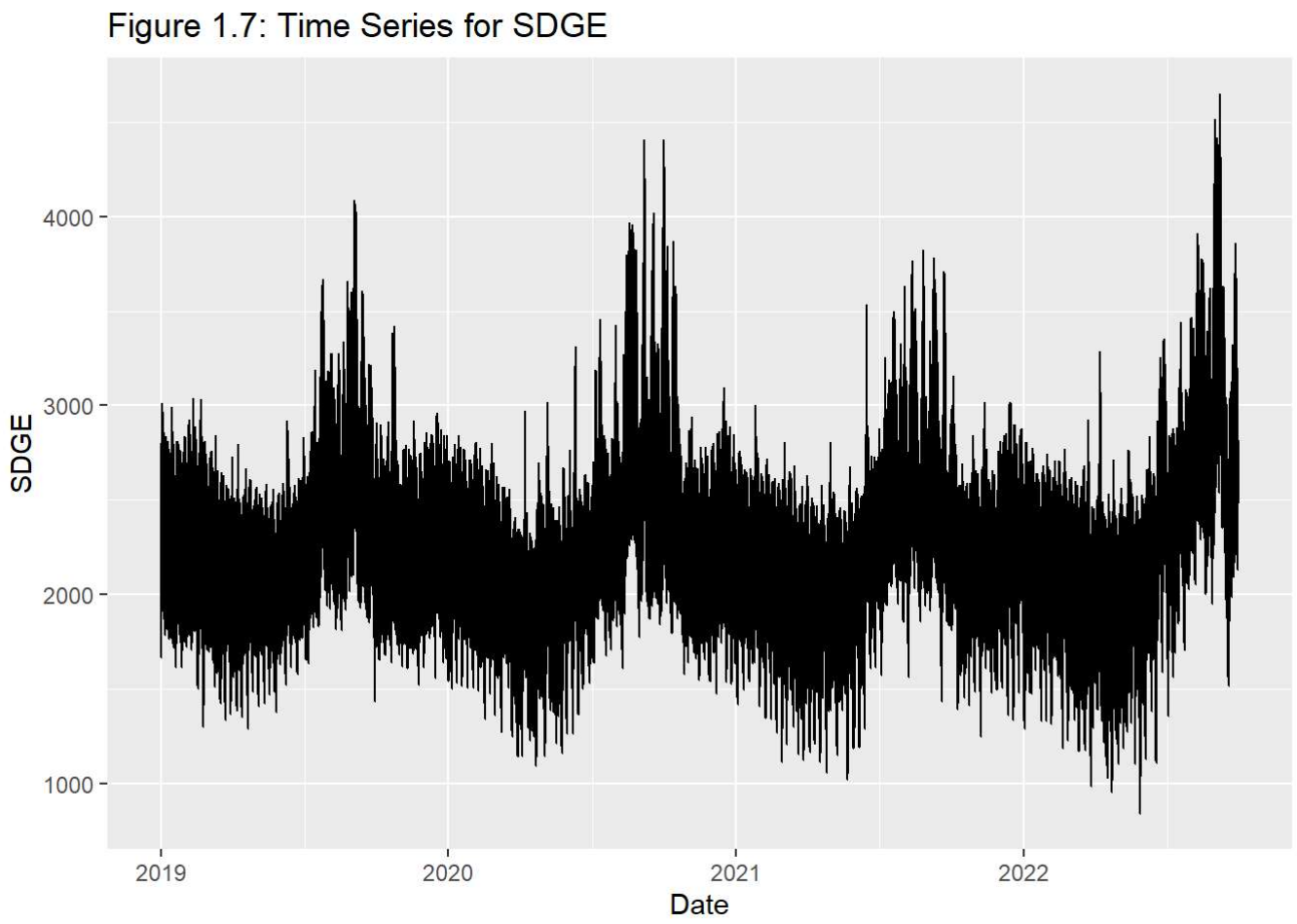

```
# For paper
paper_cols <- c(6,8)
Scatter_paper <- lapply(paper_cols, ScatPlotter.SD)
grid.arrange(grobs = Scatter_paper, top = "Figure 1.6: Scatterplots for Selected Predictors vs.
 SDGE")
```

Figure 1.6: Scatterplots for Selected Predictors vs. SDGE



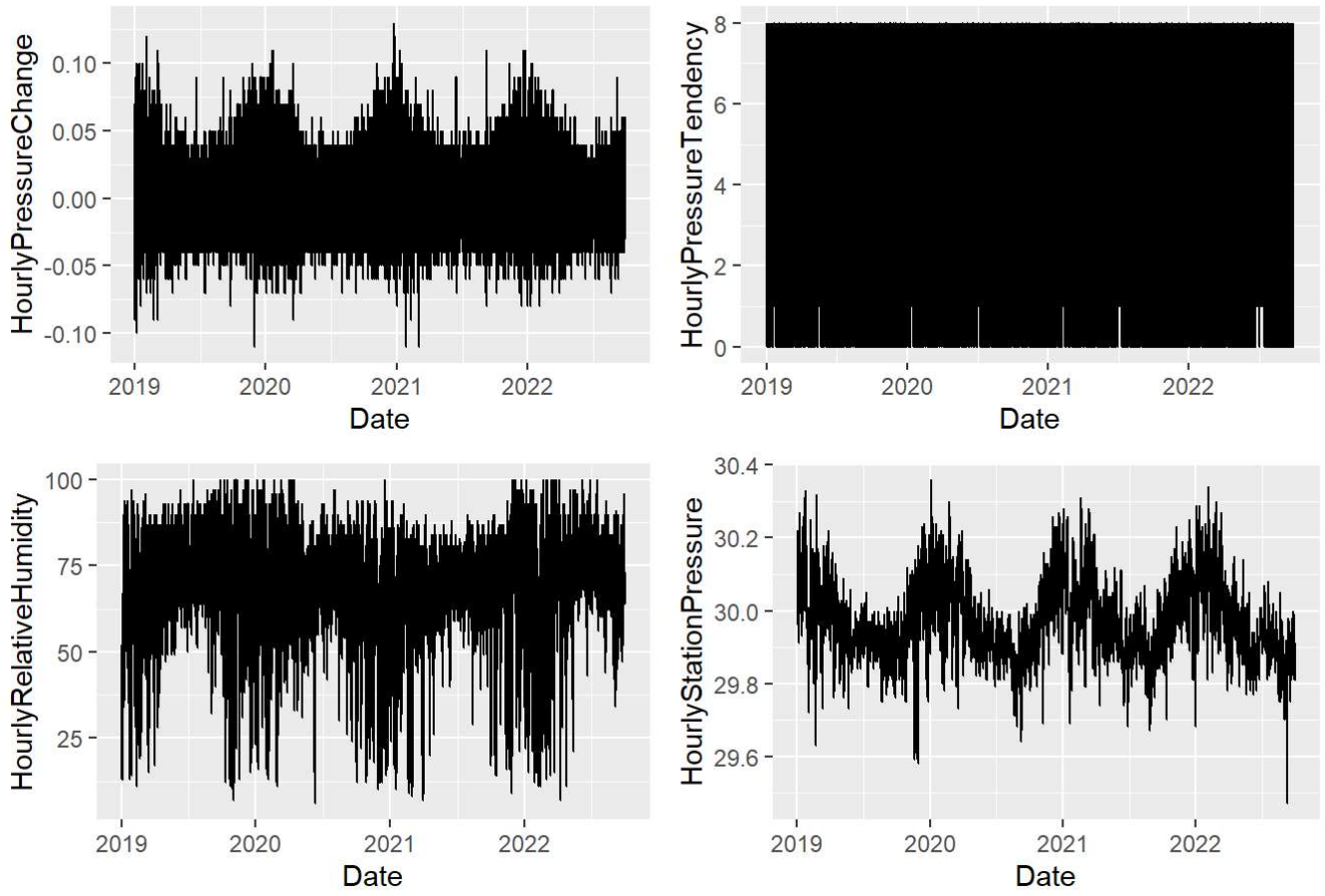
Time Series for Numeric Features Will make adjustments/sort out features to make graphs easier to read

```
num_wdate <- num2
num_wdate$Date <- SD$Date

TS_plotter <- function(x){
  ggplot(num_wdate, aes(x = Date)) +
  geom_line(aes(y = num_wdate[,x]))+
  ylab(colnames(num_wdate[x]))
}

PHP_cols <- c(3,4,5,6)
TWV_cols <- c(2,8,9,10,11)
TS_target <- TS_plotter(1) + ggtitle("Figure 1.7: Time Series for SDGE")
TS_PHP <- lapply(PHP_cols, TS_plotter)
TS_TWV <- lapply(TWV_cols, TS_plotter)
TS_paper <- lapply(paper_cols, TS_plotter)

plot(TS_target)
```

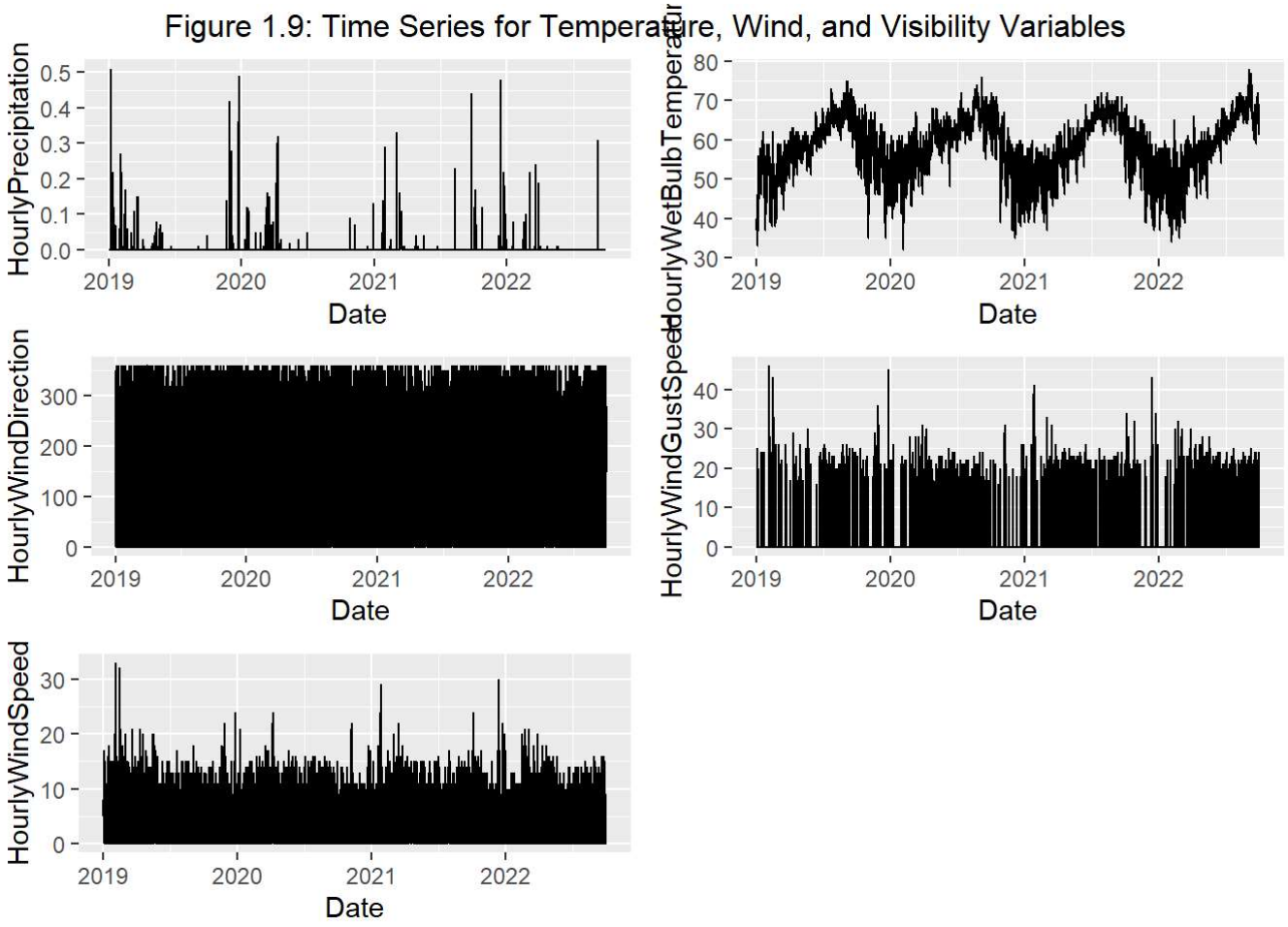



```
grid.arrange(grobs = TS_PHP, top = "Figure 1.8: Time Series for Pressure, Humidity, and Precipit
ation Variables")
```

Figure 1.8: Time Series for Pressure, Humidity, and Precipitation Variables

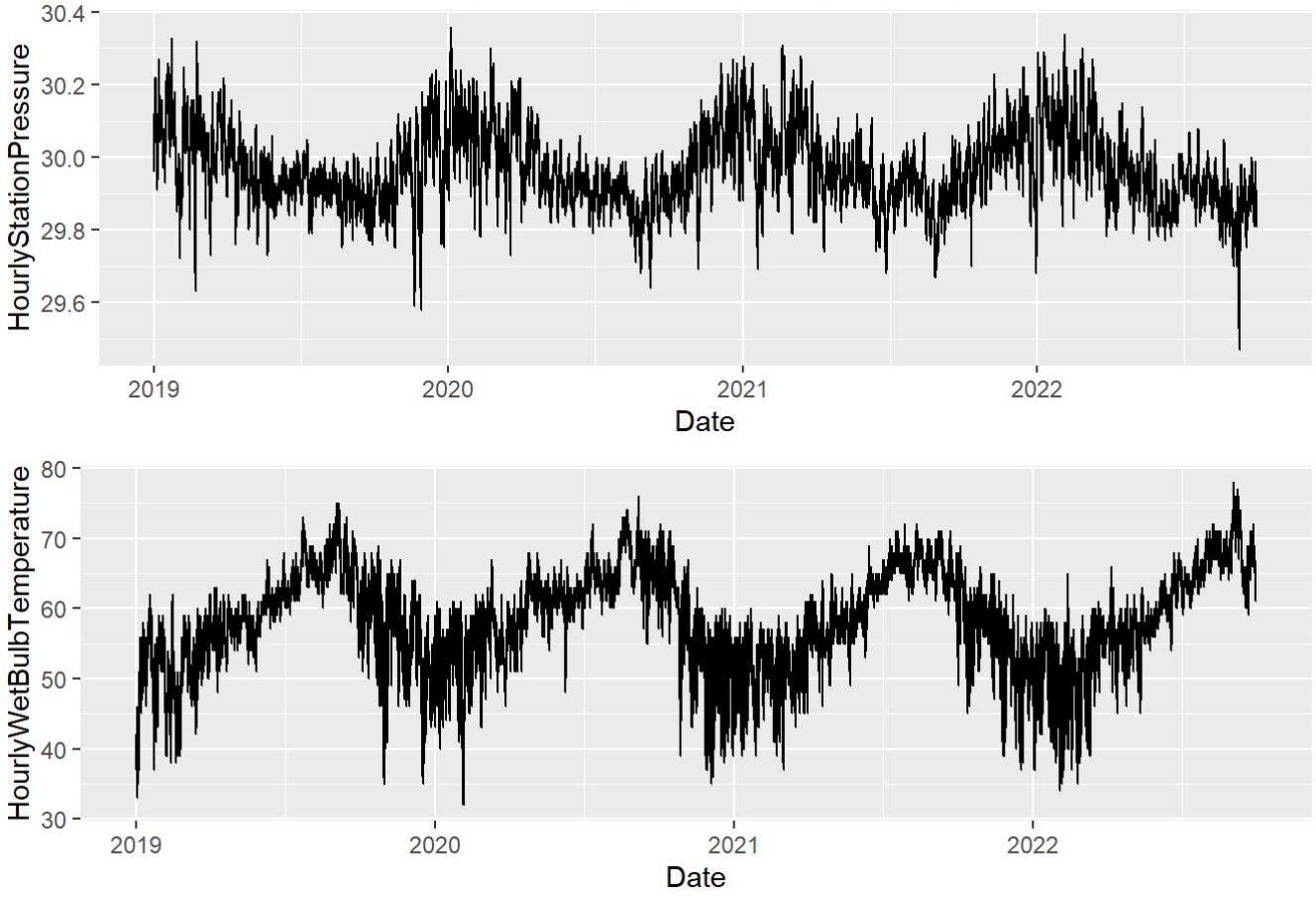



```
grid.arrange(grobs = TS_TWV, top = "Figure 1.9: Time Series for Temperature, Wind, and Visibilit
y Variables")
```



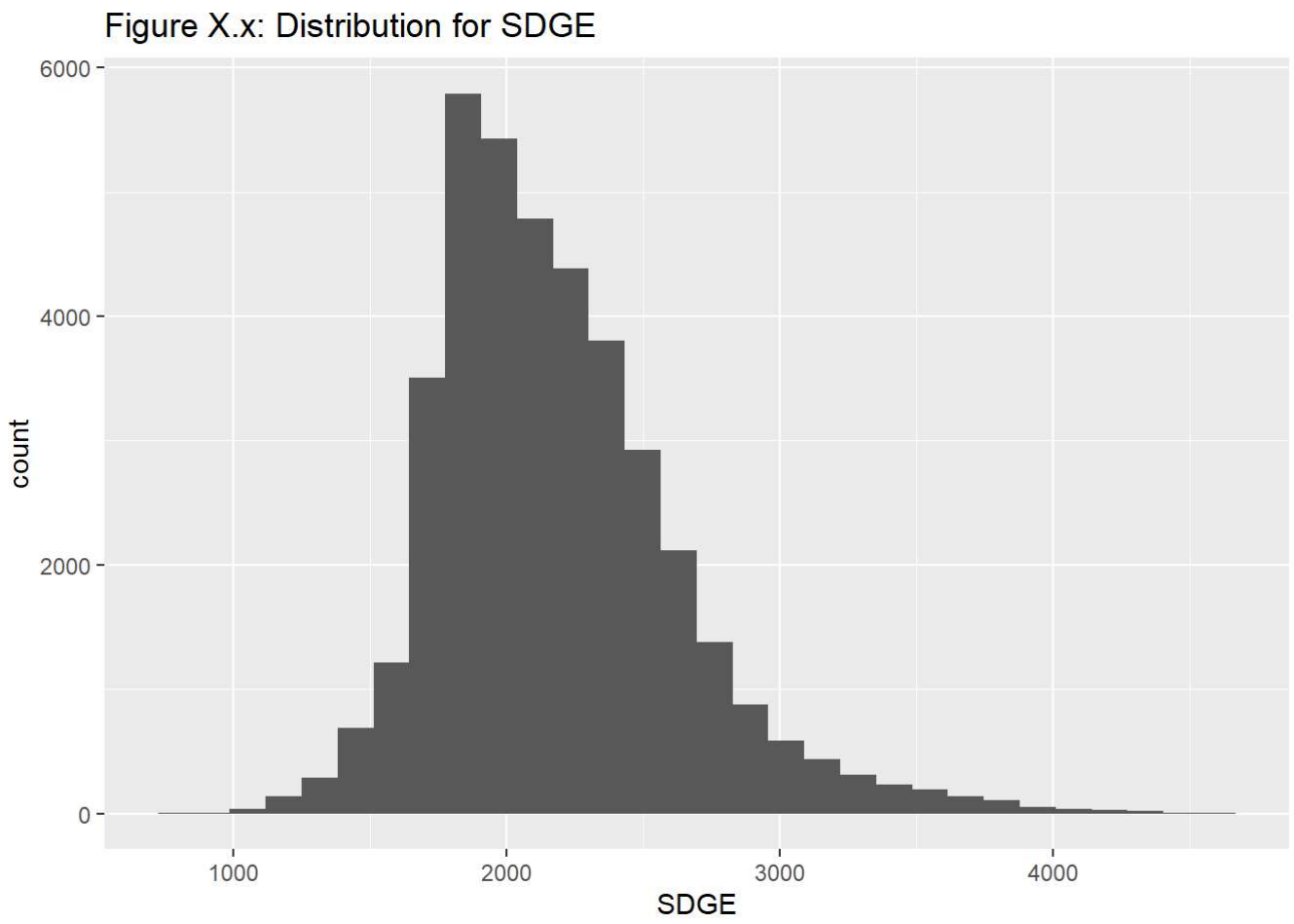

```
grid.arrange(grobs = TS_paper, top = "Figure 1.10 Time Series for Selected Predictors")
```

Figure 1.10 Time Series for Selected Predictors



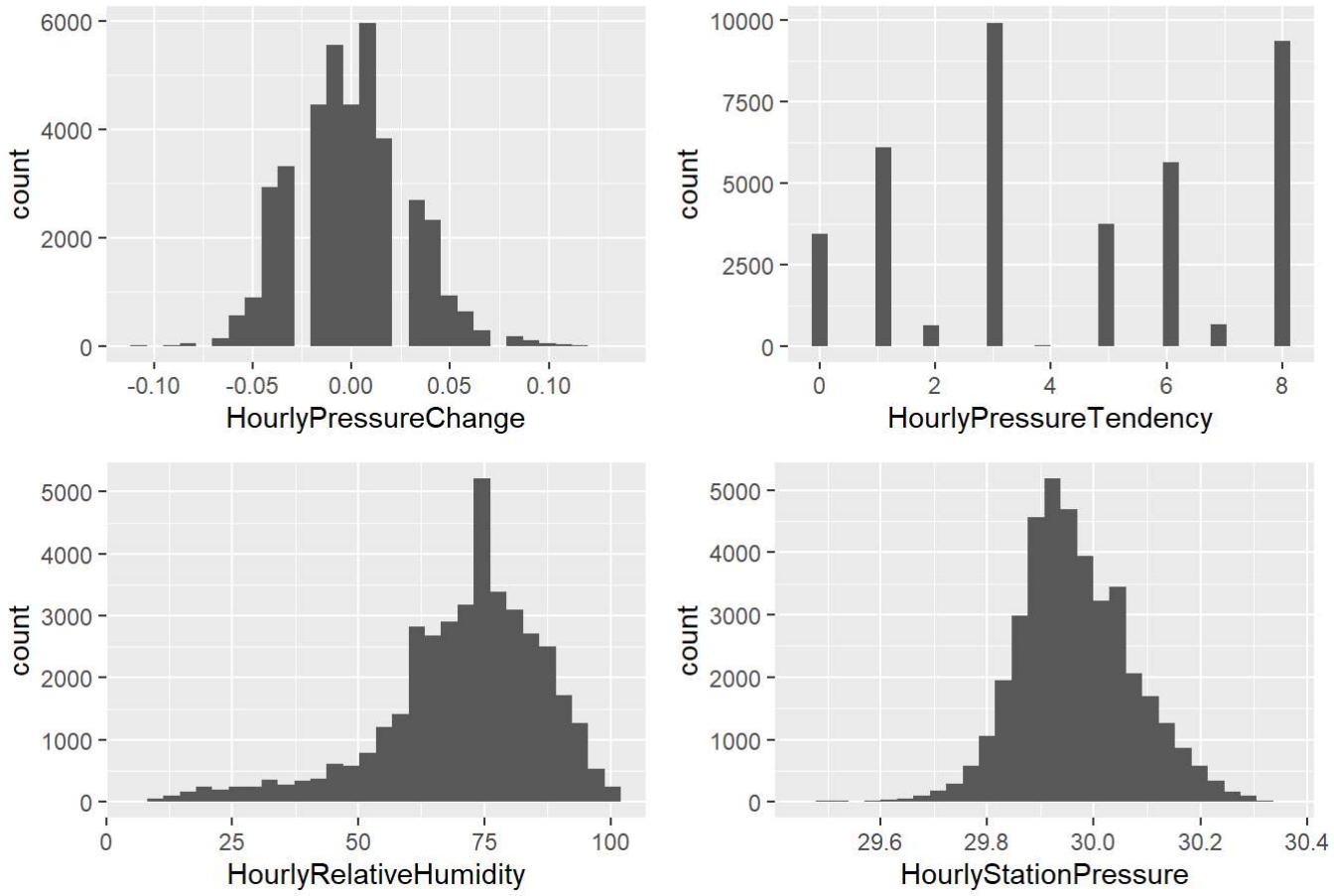
Distributions - also will make this look better tomorrow.

```
Hist_plotter <- function(x){
  ggplot(num_wdate, aes(x = num_wdate[,x])) +
    geom_histogram() +
    xlab(colnames(num_wdate[x]))
}

Hist_target <- Hist_plotter(1) + ggtitle("Figure X.x: Distribution for SDGE")
Hist_PHP <- lapply(PHP_cols, Hist_plotter)
Hist_TWV <- lapply(TWV_cols, Hist_plotter)
Hist_paper <- lapply(paper_cols, Hist_plotter)

plot(Hist_target)
```



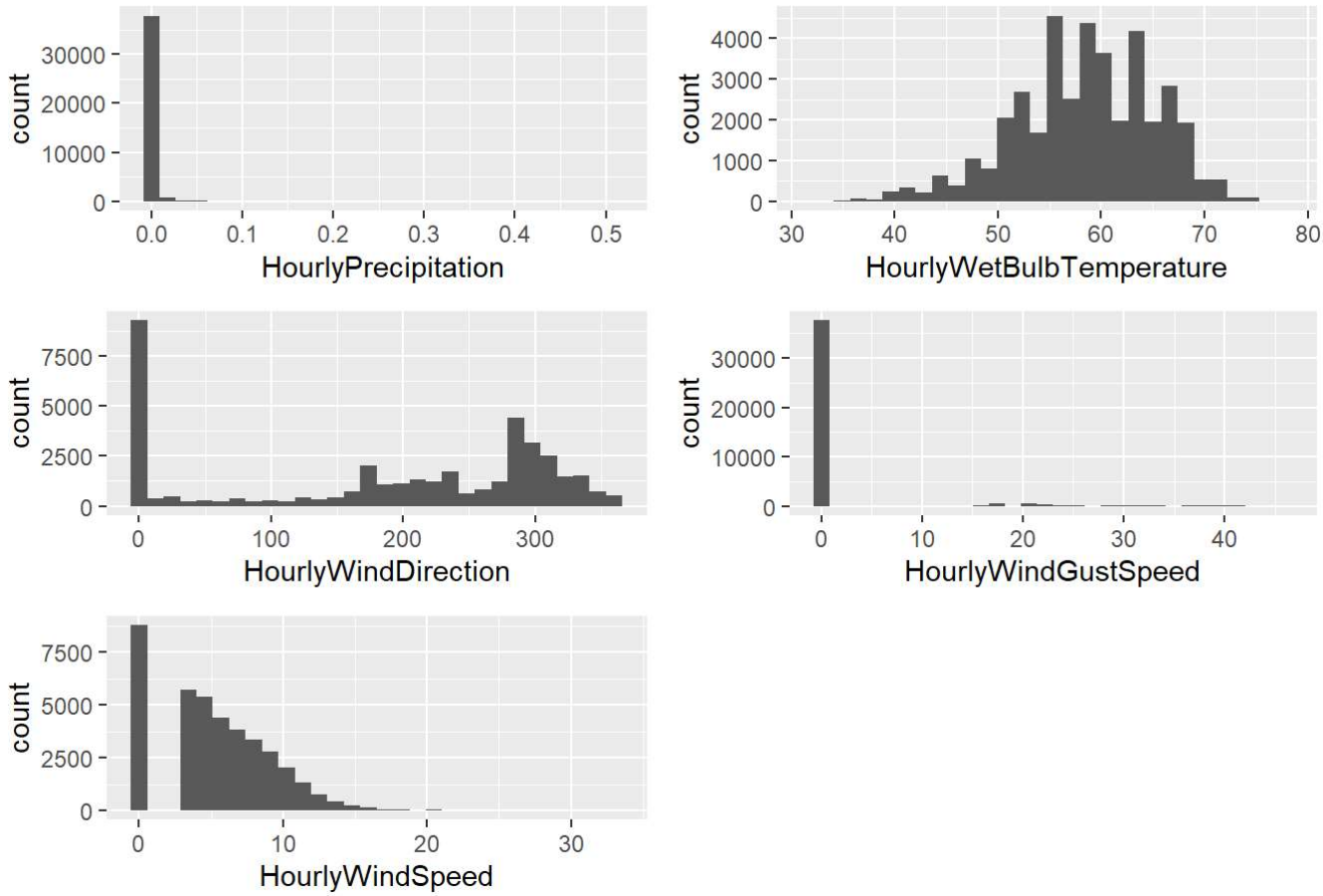

```
grid.arrange(grobs = Hist_PHP, top = "Figure X.x: Distribution for Pressure, Humidity, and Preci
pitation Variables")
```

Figure X.x: Distribution for Pressure, Humidity, and Precipitation Variables



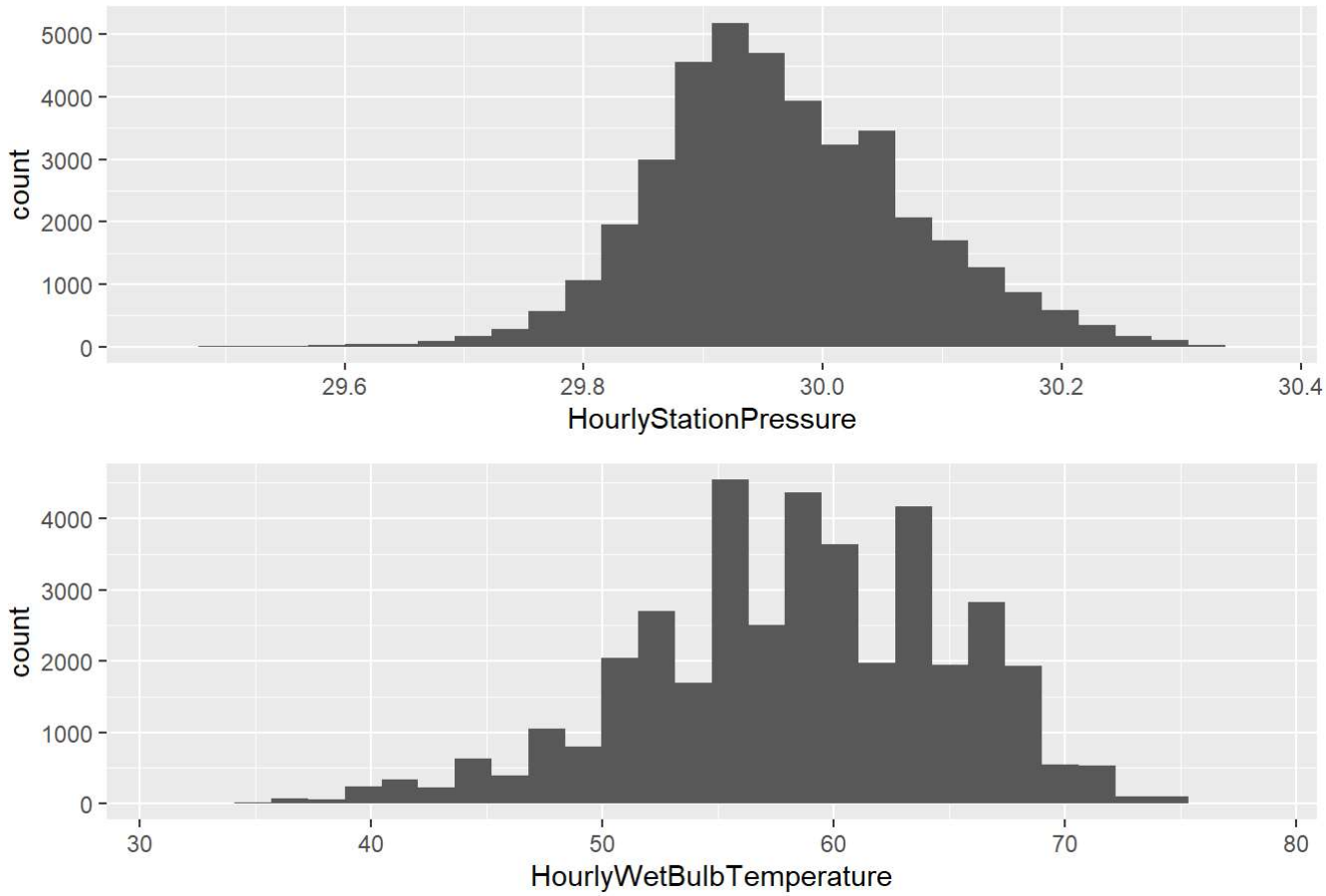

```
grid.arrange(grobs = Hist_TWV, top = "Figure X.x: Distribution for Temperature, Wind, and Visibi
lity Variables")
```

Figure X.x: Distribution for Temperature, Wind, and Visibility Variables




```
grid.arrange(grobs = Hist_paper, top = "Figure 1.13: Distributions for Selected Predictors")
```

Figure 1.13: Distributions for Selected Predictors



Categorical EDA

Category Count Tables - Bar plots were too messy since we have so many categories

```
SD %>% group_by(HourlyPresentWeatherType) %>% tally() %>% arrange(desc(n))
```

```
## # A tibble: 47 x 2
##    HourlyPresentWeatherType     n
##    <fct>                    <int>
##  1 ""                       34730
##  2 "BR:1 ||"                 2299
##  3 "-RA:02 |RA |RA"           769
##  4 "HZ:7 |FU |HZ"             492
##  5 "FG:2 |FG |"               395
##  6 "-RA:02 BR:1 |RA |RA"      343
##  7 "RA:02 |RA |RA"            104
##  8 "RA:02 BR:1 |RA |RA"        92
##  9 "-DZ:01 |DZ |DZ"            59
## 10 "+RA:02 BR:1 |RA |RA"       59
## # ... with 37 more rows
```

```
SD %>% group_by(HourlySkyConditions) %>% tally() %>% arrange(desc(n))
```

```
## # A tibble: 8,143 x 2
##    HourlySkyConditions     n
##    <fct>               <int>
##  1 CLR:00               3875
##  2 FEW:02 250           1513
##  3 SCT:04 250            624
##  4 FEW:02 10             519
##  5 BKN:07 250            487
##  6 OVC:08 15             465
##  7 OVC:08 14             446
##  8 FEW:02 15             378
##  9 OVC:08 10             377
## 10 OVC:08 13             375
## # ... with 8,133 more rows
```

ANOVAs to see if category affects SDGE - Wouldn't work with SkyConditions since there are 8,145 categories. I think that having so many groups is forcing a significant result for WeatherType, looking at our Tukey most categories aren't different, so we'll just stick with our numeric features.

```
WT_aov <- aov(SDGE ~ HourlyPresentWeatherType, data = SD)

summary(WT_aov)
```

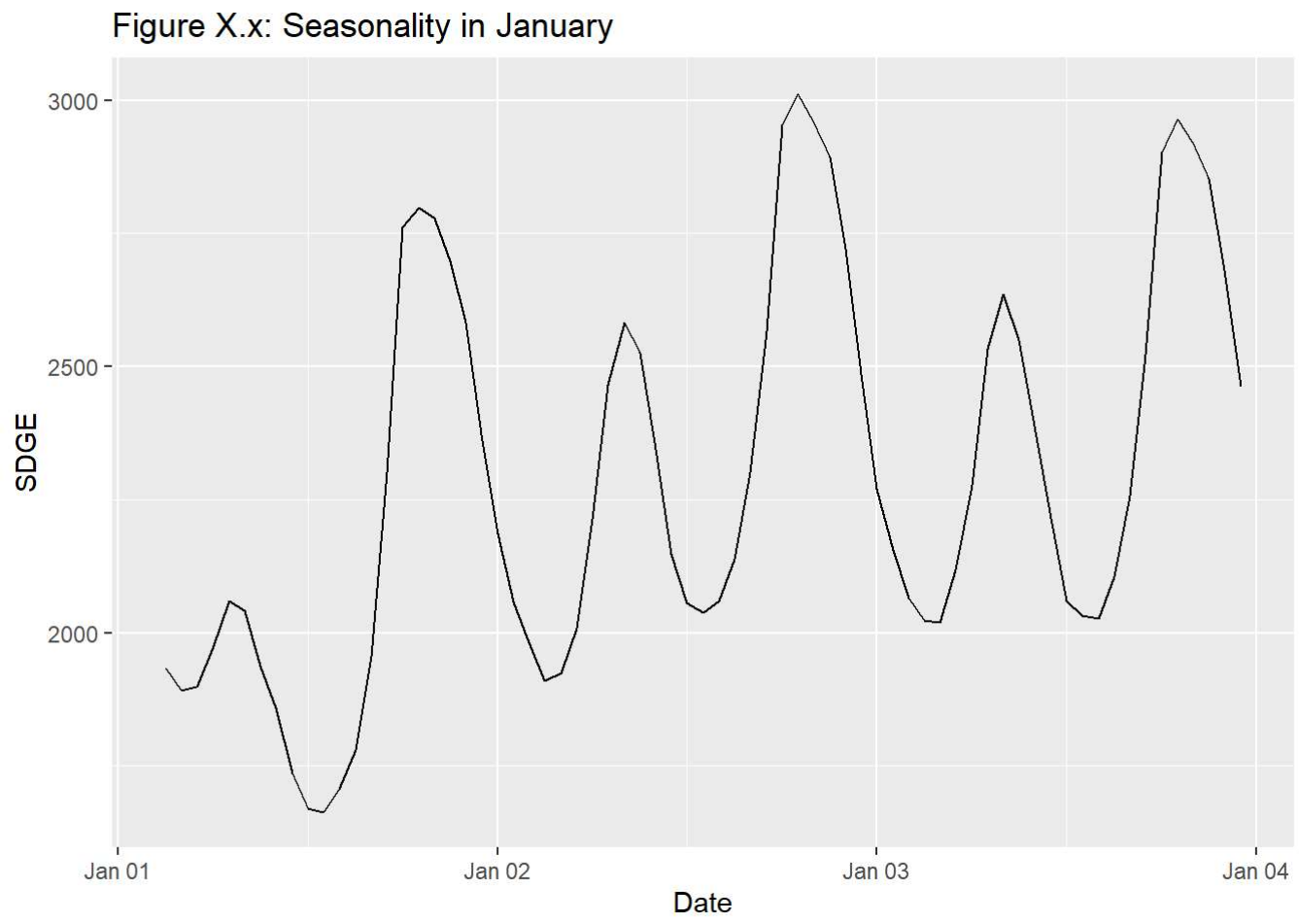
```
##                             Df    Sum Sq Mean Sq F value Pr(>F)
## HourlyPresentWeatherType    46 5.866e+07 1275246   6.635 <2e-16 ***
## Residuals                39438 7.580e+09  192199
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#Sky_aov <- aov(SDGE ~ HourlySkyConditions, data = SD) #This has too many categories and takes f
orever to run.

WT_tukey <- TukeyHSD(WT_aov)
```

Target EDA

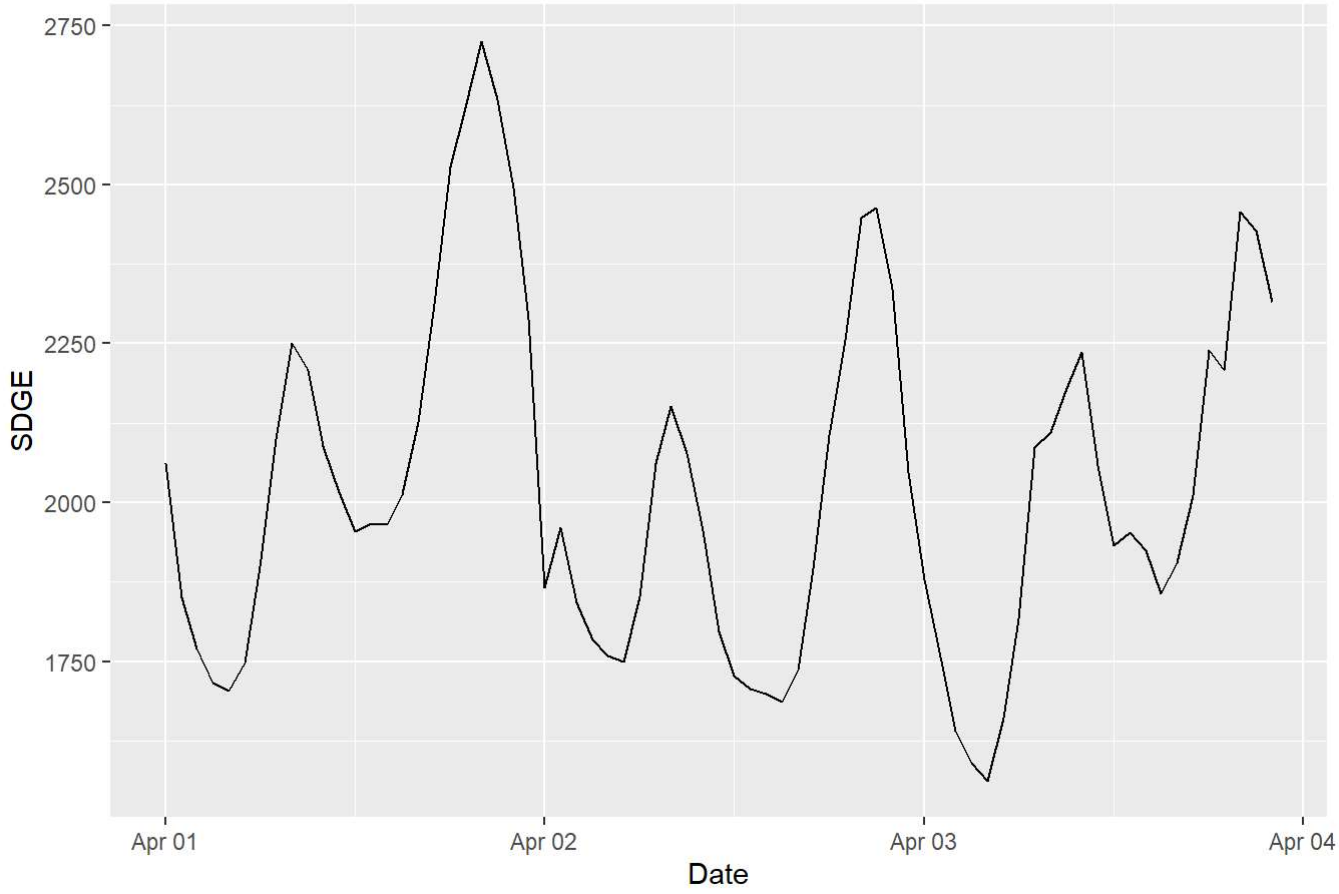
We've already seen yearly seasonality in SDGE from time series above, lets evaluate a few days to see if there is also a daily seasonality. We'll evaluate the first couple days in Jan, Apr, Jul, and Oct. We definitely see a daily-based seasonality. So, we'll have to account for seasonality across the year and the day when modelling.

```
SD %>% filter(Date < '2019-01-03 18:00:00') %>% ggplot(aes(x = Date, y = SDGE)) + geom_line() +
 ggtitle("Figure X.x: Seasonality in January")
```



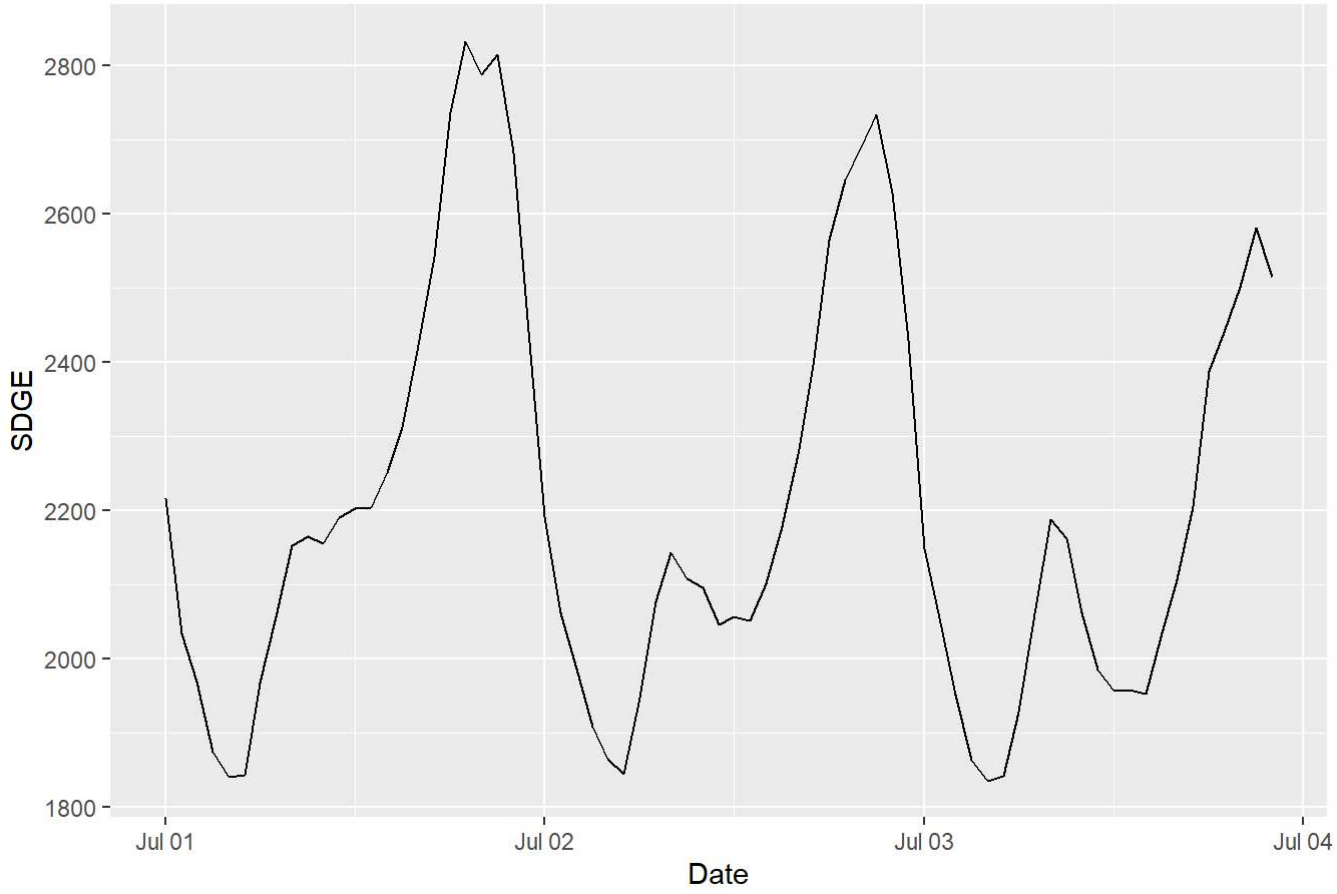

```
SD %>% filter(Date < '2019-04-03 18:00:00') %>% filter(Date > '2019-03-31 18:00:00') %>% ggplot
(aes(x = Date, y = SDGE)) + geom_line() + ggtitle("Figure 1.14: Seasonality in April")
```

Figure 1.14: Seasonality in April



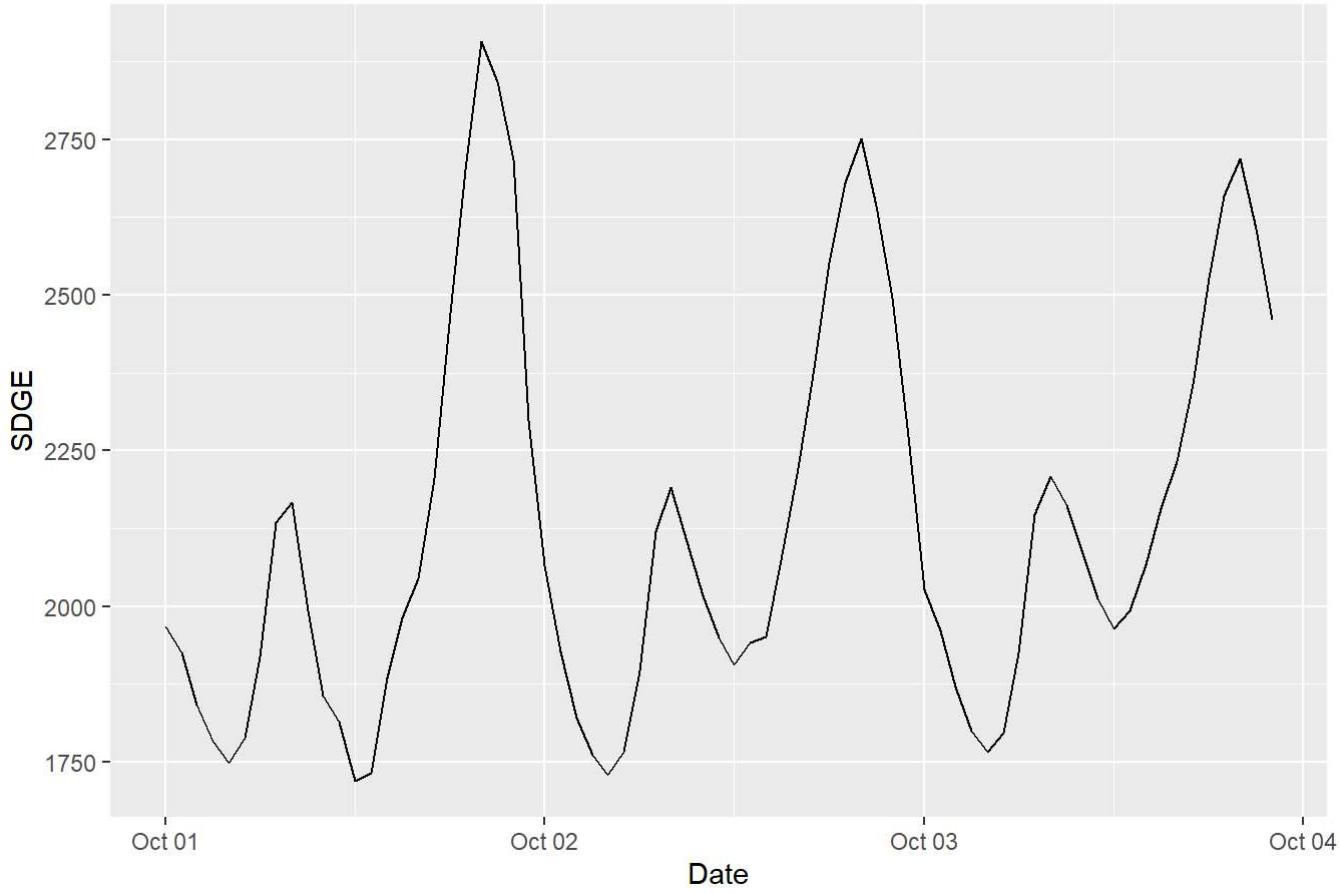

```
SD %>% filter(Date < '2019-07-03 18:00:00') %>% filter(Date > '2019-06-30 18:00:00') %>% ggplot
(aes(x = Date, y = SDGE)) + geom_line() + ggtitle("Figure 1.15: Seasonality in July")
```

Figure 1.15: Seasonality in July




```
SD %>% filter(Date < '2019-10-03 18:00:00') %>% filter(Date > '2019-09-30 18:00:00') %>% ggplot
(aes(x = Date, y = SDGE)) + geom_line() + ggtitle("Figure 1.16: Seasonality in October")
```

Figure 1.16: Seasonality in October



Section 2 - Modelling

When incorporating weather variables into the model, we will lag the predictors behind by 1-hour. This will make it so the SDGE has approximately 1 hour and 5 minutes to take any action on the with a forecast (since weather originally came in at minute 54 of each hour + 1 minute to run the model and return the prediction).

To simulate a real-life scenario, we will use a sliding train/test window for models that need specific training periods for weights (like exponential smoothing/regression). So, the validation set will begin in the last month (Sep 2022), then we will slide 1 hour forward and rerun the model, iterating until we reach the end of the month.

Any non-weighted method (Naive/moving average) will be calculated on the entire data set and then split for evaluation, since we can assume we'll always have the last n-observations we need for the forecast.

Baseline Model - Naive Forecast

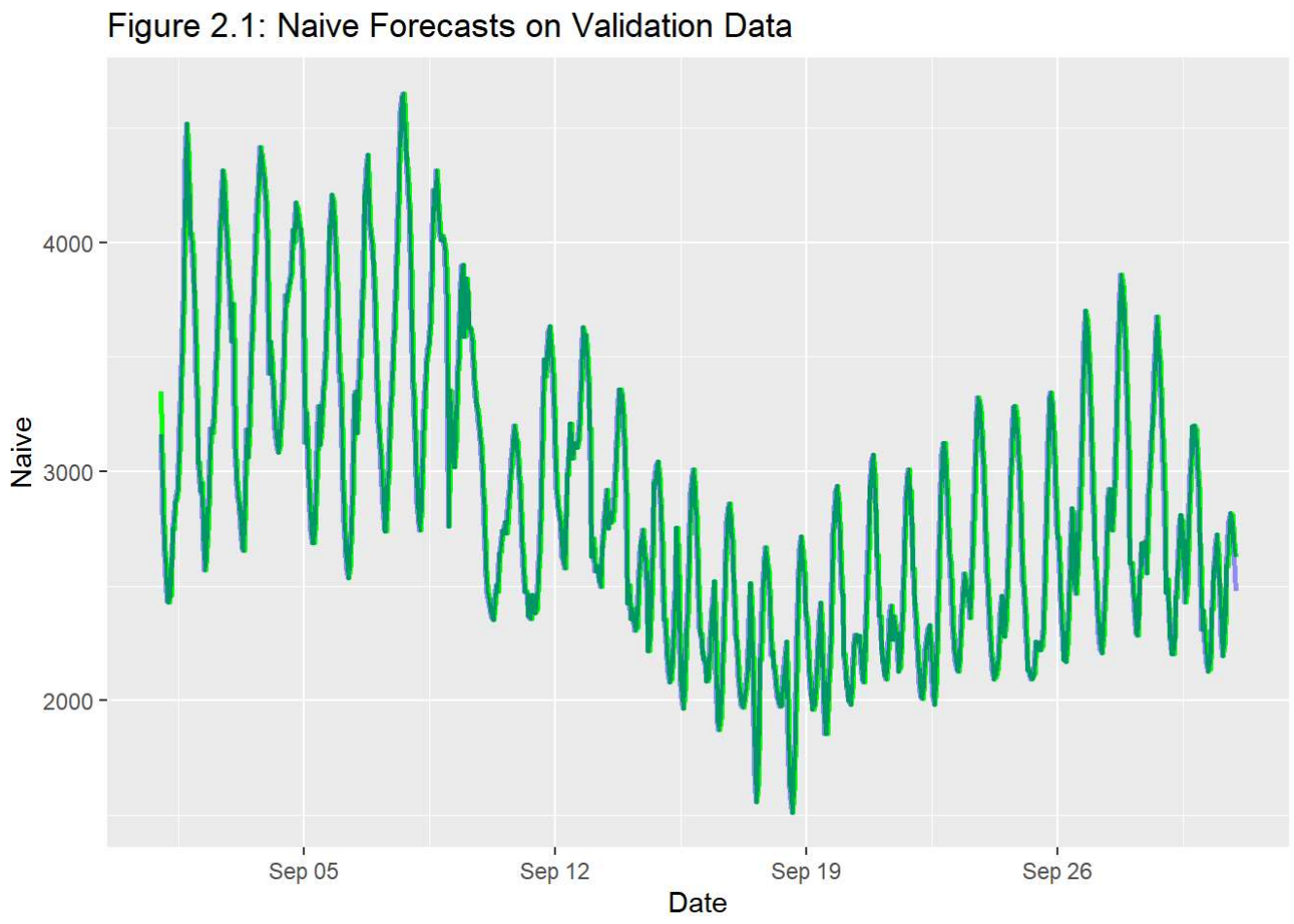
Create forecasts and calculate metrics

```
# SD got knocked out of order somewhere, rearrange to make sure lagging works
SD <- SD %>% arrange(Date)

# Create Naive Forecasts
SD$Naive <- lag(SD$SDGE)

# Calculate Error
SD$Naive_err <- SD$SDGE - SD$Naive

# Split for evaluation
SD_tr <- SD %>% filter(Date < '2022-08-31 19:00:00') %>% na.omit()
SD_val <- SD %>% filter(Date > '2022-08-31 18:00:00')

# Calculate Metrics
Naive_rmse_tr <- sqrt(mean(SD_tr$Naive_err^2))
Naive_rmse_val <- sqrt(mean(SD_val$Naive_err^2))

paste0("We had a RMSE of ", round(Naive_rmse_val,2), " with naive forecasts on the validation da
ta.")
```

```
## [1] "We had a RMSE of 148.92 with naive forecasts on the validation data."
```

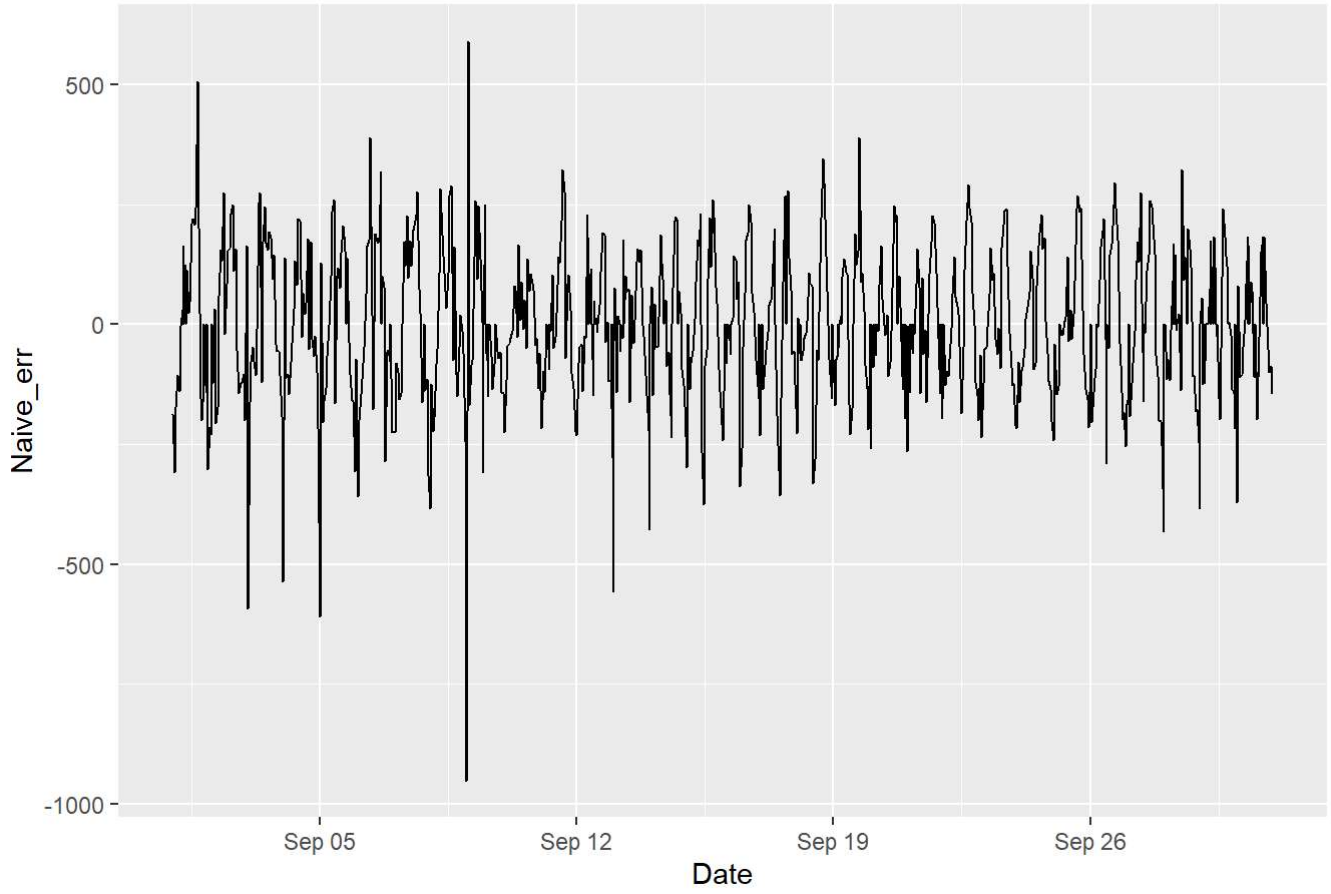
Plot on validation data

```
ggplot(SD_val, aes(x = Date)) +
  geom_line(aes(y = Naive), color = 'green', linewidth = 1) +
  geom_line(aes(y = SDGE), color = 'blue', alpha = 0.4, linewidth = 1) +
  ggtitle("Figure 2.1: Naive Forecasts on Validation Data")
```




```
ggplot(SD_val, aes(x = Date, y = Naive_err)) +
  geom_line() +
  ggtitle("Figure 2.2: Residuals for Naive Forecasts on Validation Data")
```

Figure 2.2: Residuals for Naive Forecasts on Validation Data



Moving Average Forecasts

We'll use a function to evaluate different window sizes

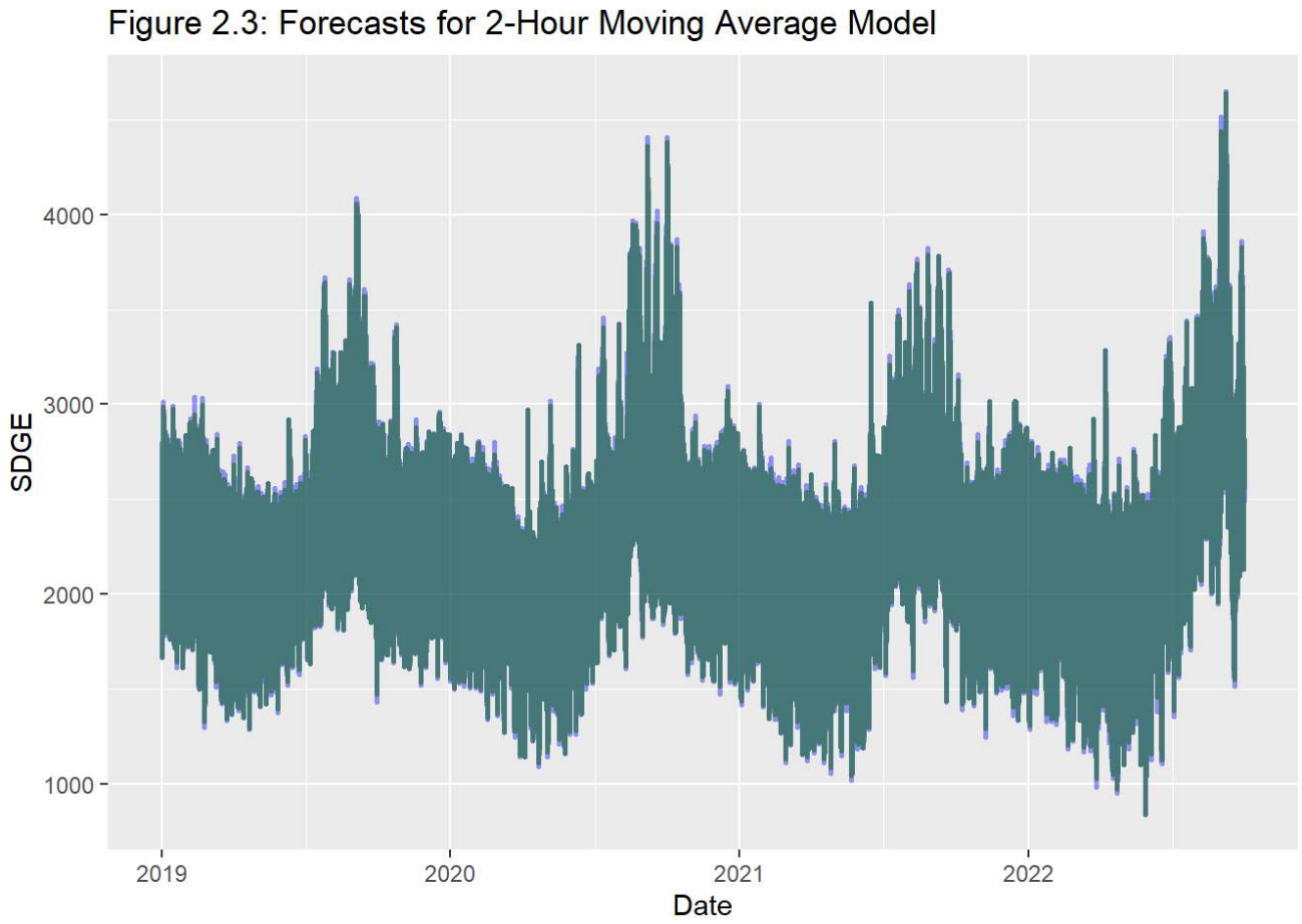
```
SD_madf <- SD
RMSE_tr <- c()
RMSE_val <- c()

SD.ma <- function(k){

  # Calculate forecasts
  DailyMA <- rollmean(SD_madf$SDGE, k = k)

  # Add NAs to forecast vector so we can bind it to our df
  DailyMA <- c(rep(NA, k-1), DailyMA)

  SD_madf$DailyMA <- DailyMA

  # Calculate Error
  SD_madf$DailyMA_err <- SD_madf$SDGE - SD_madf$DailyMA

  # Split for eval
  SD_tr <- SD_madf %>% filter(Date < '2022-08-31 19:00:00') %>% na.omit()
  SD_val <- SD_madf %>% filter(Date > '2022-08-31 18:00:00')

  # Calculate Metrics
  RMSE_tr <- sqrt(mean(SD_tr$DailyMA_err^2))
  RMSE_val <- sqrt(mean(SD_val$DailyMA_err^2))

  # List and return
  RMSE_list <- list(k, RMSE_tr, RMSE_val)

}

k = c(1:24, seq(48, 168, 24))

# Create results data frame
SD_ma_res <- lapply(k, SD.ma)
SD_ma_res <- as.data.frame(do.call(rbind, SD_ma_res))
colnames(SD_ma_res) <- c("k", "RMSE_tr", "RMSE_val")

SD_ma_res <- SD_ma_res %>% filter(RMSE_tr > 0)
best_window <- SD_ma_res[which.min(SD_ma_res$RMSE_tr), ]
best_window$k
```

```
## [[1]]
## [1] 2
```

```
MA_rmse_tr <- best_window$RMSE_tr[[1]]
MA_rmse_val <- best_window$RMSE_val[[1]]
```

Plot

```
DailyMA_2 <- rollmean(SD_madf$SDGE, k = 2)
DailyMA_2 <- c(rep(NA, 1), DailyMA_2)


SD_madf$DailyMA_2 <- DailyMA_2

ggplot(SD_madf, aes(x = Date)) +
  geom_line(aes(y = SDGE), color = "blue", linewidth = 1, alpha = 0.4) +
  geom_line(aes(y = DailyMA_2), color = "darkgreen", linewidth = 1, alpha = 0.5) +
  ggtitle("Figure 2.3: Forecasts for 2-Hour Moving Average Model")
```



Auto-ARIMA with predictors

Split predictors and target - WetBulbTemp and StationPressure were the only variables with slight relationships, so we'll use those.

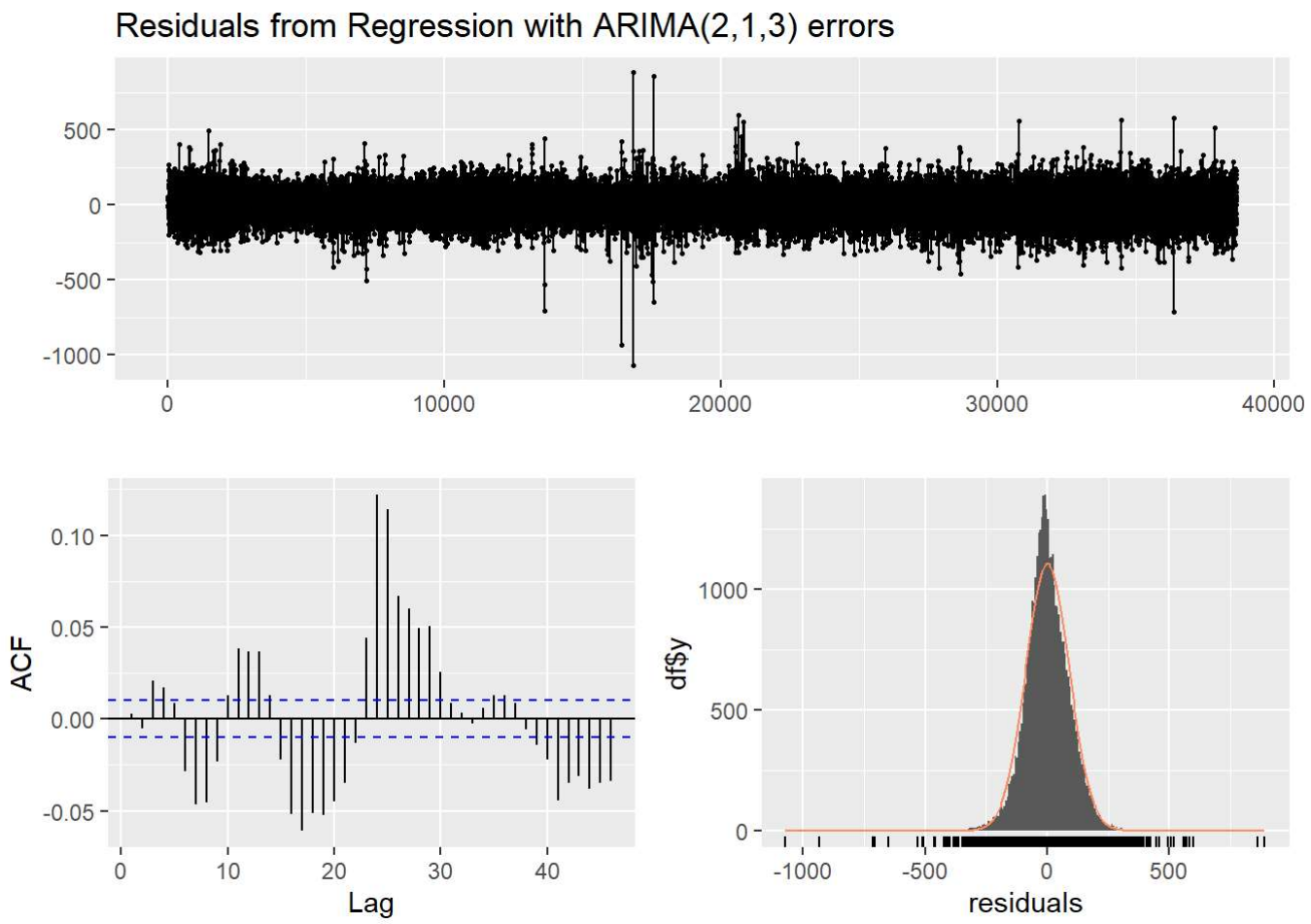
```
X_tr <- SD_tr %>% select(HourlyWetBulbTemperature, HourlyStationPressure) %>% as.matrix()
X_val <- SD_val %>% select(HourlyWetBulbTemperature, HourlyStationPressure) %>% as.matrix()
y_tr <- SD_tr$SDGE
y_test <- SD_val$SDGE
```

Create Model

```
set.seed(123)
auto_arima_model <- auto.arima(y_tr, xreg = X_tr, seasonal = TRUE, stepwise = FALSE)
summary(auto_arima_model)
```

```
## Series: y_tr
## Regression with ARIMA(2,1,3) errors
##
## Coefficients:
##          ar1      ar2      ma1     ma2      ma3  HourlyWetBulbTemperature
##       1.6628  -0.7662  -1.2538  0.3669  -0.0969                    4.3833
## s.e.  0.0055   0.0053   0.0073  0.0086   0.0059                    0.4049
##       HourlyStationPressure
##                   419.9968
## s.e.                38.4614
##
## sigma^2 = 7916:  log likelihood = -228200.4
## AIC=456416.7   AICc=456416.7   BIC=456485.2
##
## Training set error measures:
##                     ME     RMSE    MAE        MPE     MAPE      MASE
## Training set 0.1335366 88.96111 67.611 -0.1351454 3.152328 0.7576989
##                     ACF1
## Training set 0.002582904
```

```
auto_arima_rmse_tr <- sqrt(mean(auto_arima_model$residuals^2))

checkresiduals(auto_arima_model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(2,1,3) errors
## Q* = 254.15, df = 5, p-value < 2.2e-16
##
## Model df: 5.   Total lags used: 10
```

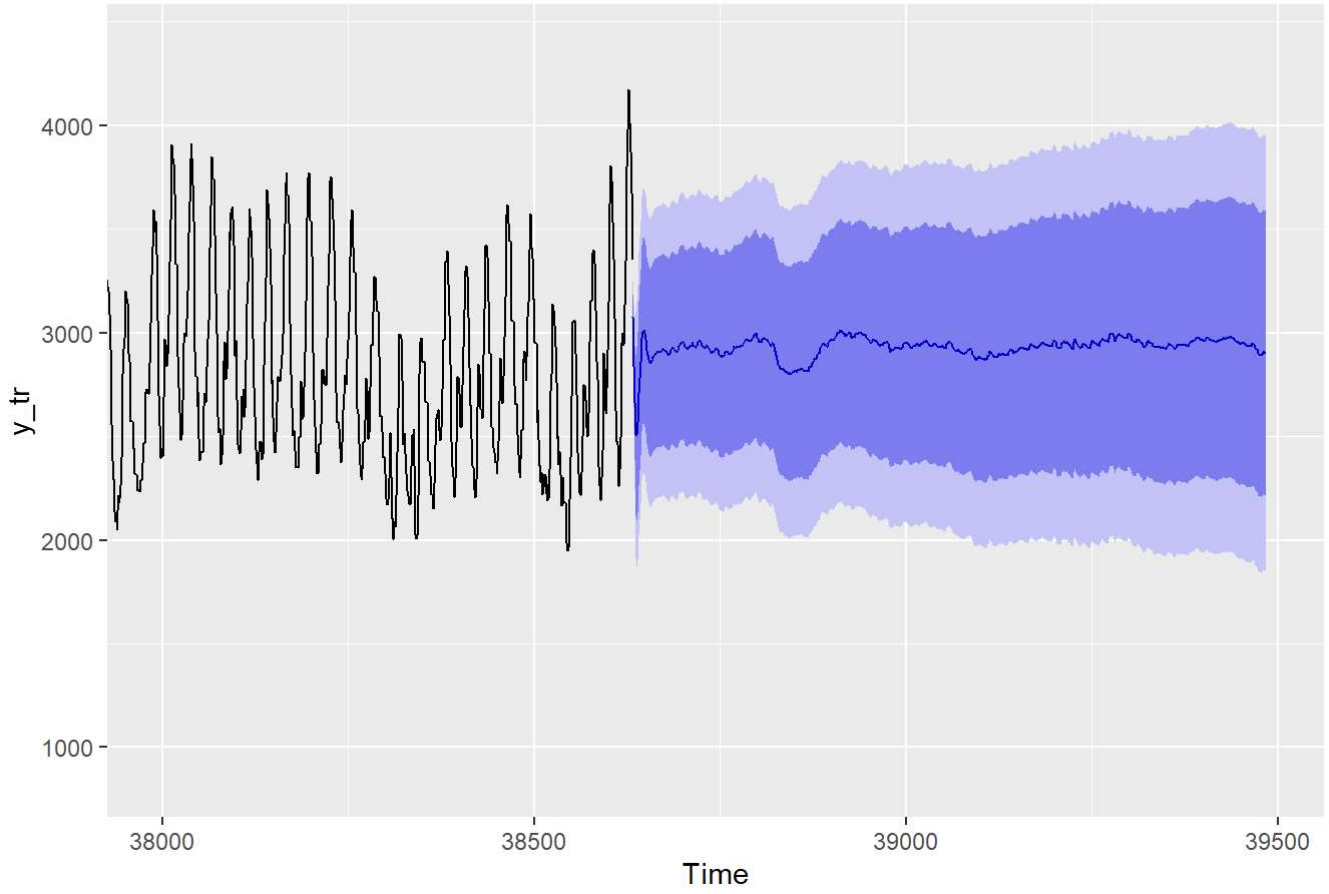
Forecast

```
val_forecast <- forecast(auto_arima_model, xreg = X_val)

# Calculate errors
val_forecast_err <- SD_val$SDGE - val_forecast$mean
auto_arima_rmse_val <- sqrt(mean(val_forecast_err^2))
```

Plot

```
autoplot(val_forecast, series = "Forecast") +
  coord_cartesian(xlim = c(38000, 39488)) +
  ggtitle("Figure 2.5: Forecasts from Regression with ARIMA(2,1,3) errors")
```

Figure 2.5: Forecasts from Regression with ARIMA(2,1,3) errors



ETS

Create time series - Will just do it for everything in case we need predictor ts later

```
X_tr_ts <- ts(X_tr, frequency = 24)
X_val_ts <- ts(X_val, frequency = 24)
y_val_ts <- ts(y_test, frequency = 24)
y_tr_ts <- ts(y_tr, frequency = 24)
```

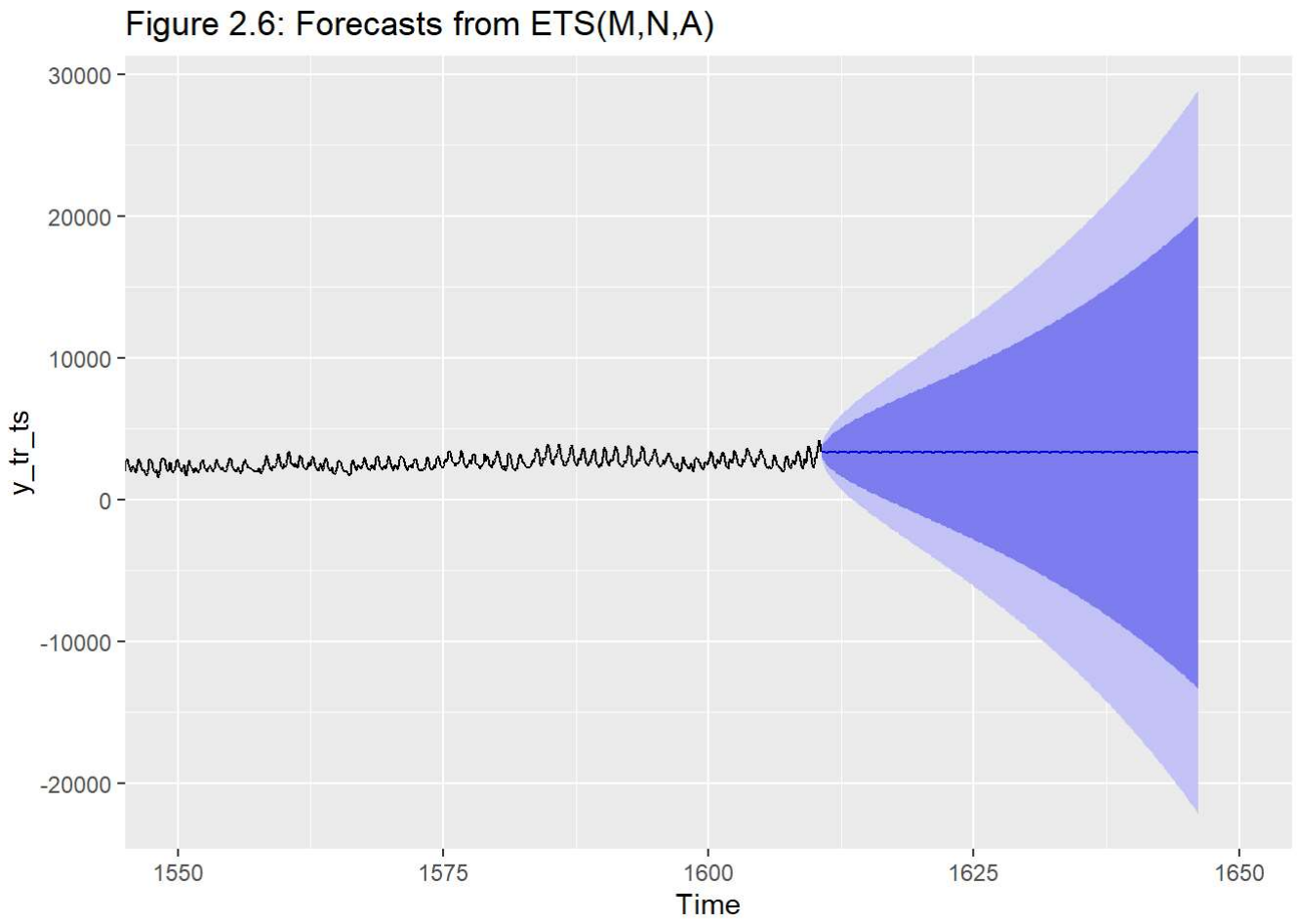
Fit ETS

```
SD_ets <- ets(y_tr_ts, model = "ZNA")
summary(SD_ets)
```

```
## ETS(M,N,A)
##
## Call:
##  ets(y = y_tr_ts, model = "ZNA")
##
##   Smoothing parameters:
##     alpha = 0.9972
##     gamma = 0.0028
##
##   Initial states:
##     l = 2238.6627
##     s = -4.2951 1.6768 2.219 8.0848 13.2796 20.7892
##            69.2731 40.3217 26.6011 3.5964 -1.3387 -3.4258 -3.7249 -6.2906 -6.1811 -11.8018 -1
3.4591 -12.6333 -7.5075 -4.7885 -3.1824 1.2944 2.7467 -111.254
##
##   sigma:  0.0573
##
##      AIC     AICc      BIC
## 779017.7 779017.7 779248.9
##
## Training set error measures:
##                       ME     RMSE      MAE        MPE     MAPE     MASE
## Training set 0.02877078 121.8588 91.10728 -0.1583515 4.237729 0.305293
##                    ACF1
## Training set 0.5911181
```

Forecast, Calculate RMSE, and Plot

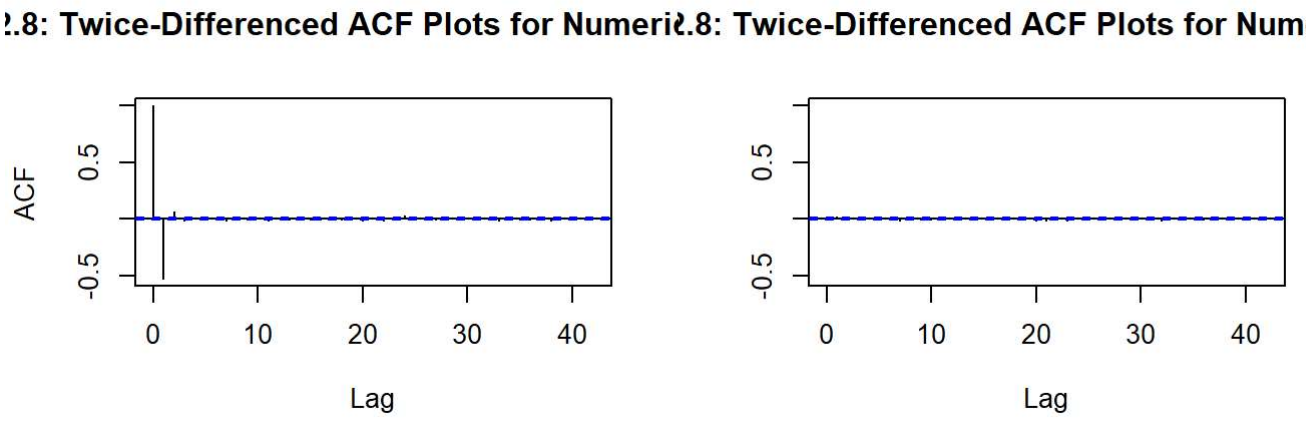
```
ets_forecast <- forecast(SD_ets, h=length(y_test))

#RMSE
val_forecast_err <- SD_val$SDGE - ets_forecast$mean
ets_rmse_val <- sqrt(mean(val_forecast_err^2))

# Not sure what happened, but training residuals didn't calculate right, so we have to calculate
them ourselves
tr_forecast_err <- SD_tr$SDGE - SD_ets$fitted
ets_rmse_tr <- sqrt(mean(tr_forecast_err^2))

#plot
autoplot(ets_forecast, series = "Forecast") +
  coord_cartesian(xlim = c(1550, 1650)) +
  ggtitle("Figure 2.6: Forecasts from ETS(M,N,A)")
```

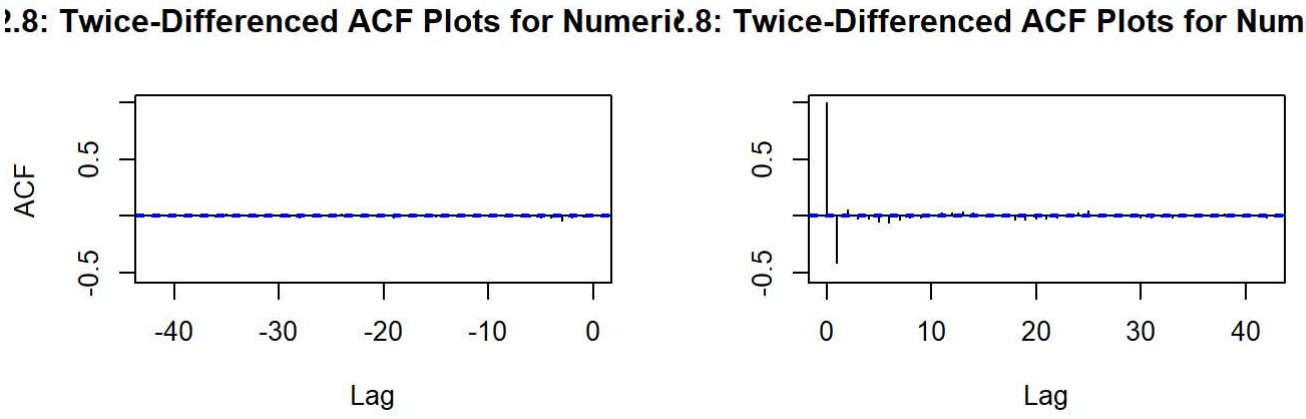


Manual ARIMA

Twice differenced ACF plot

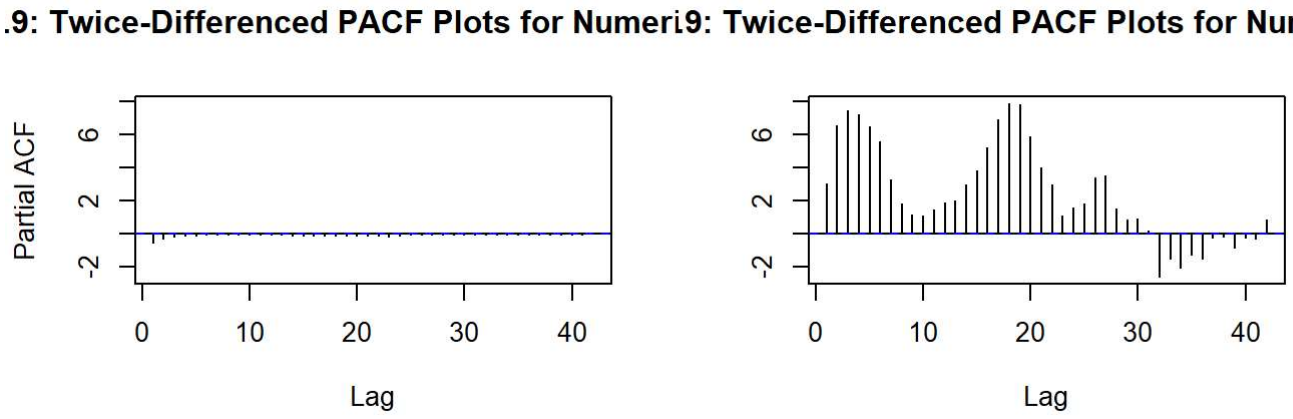
```
acf(diff(diff(X_tr)) , main = "Figure 2.8: Twice-Differenced ACF Plots for Numeric Predictors")
```

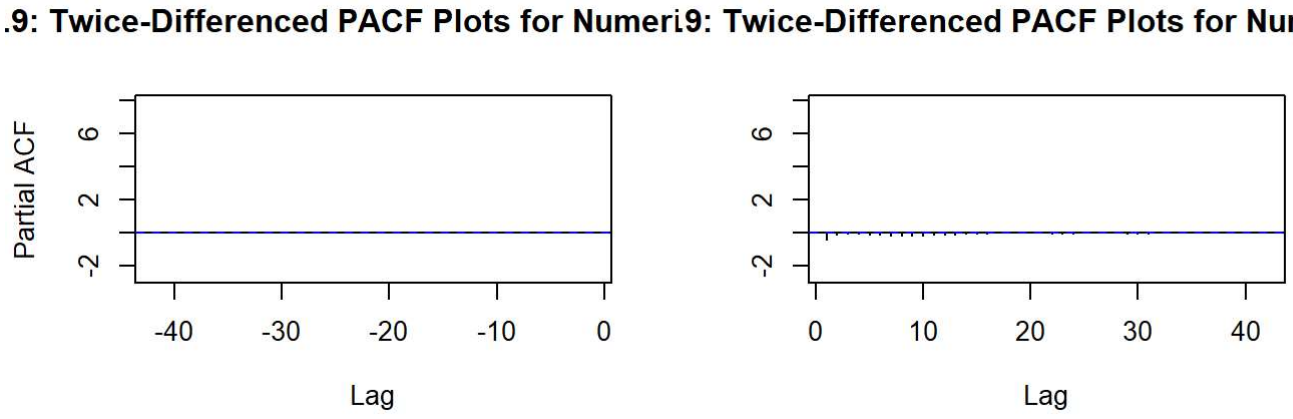




Twice differenced PACF plot

```
#take a look pacf plot with differenced
pacf(diff(diff(X_tr)), main = "Figure 2.9: Twice-Differenced PACF Plots for Numeric Predictors")
```





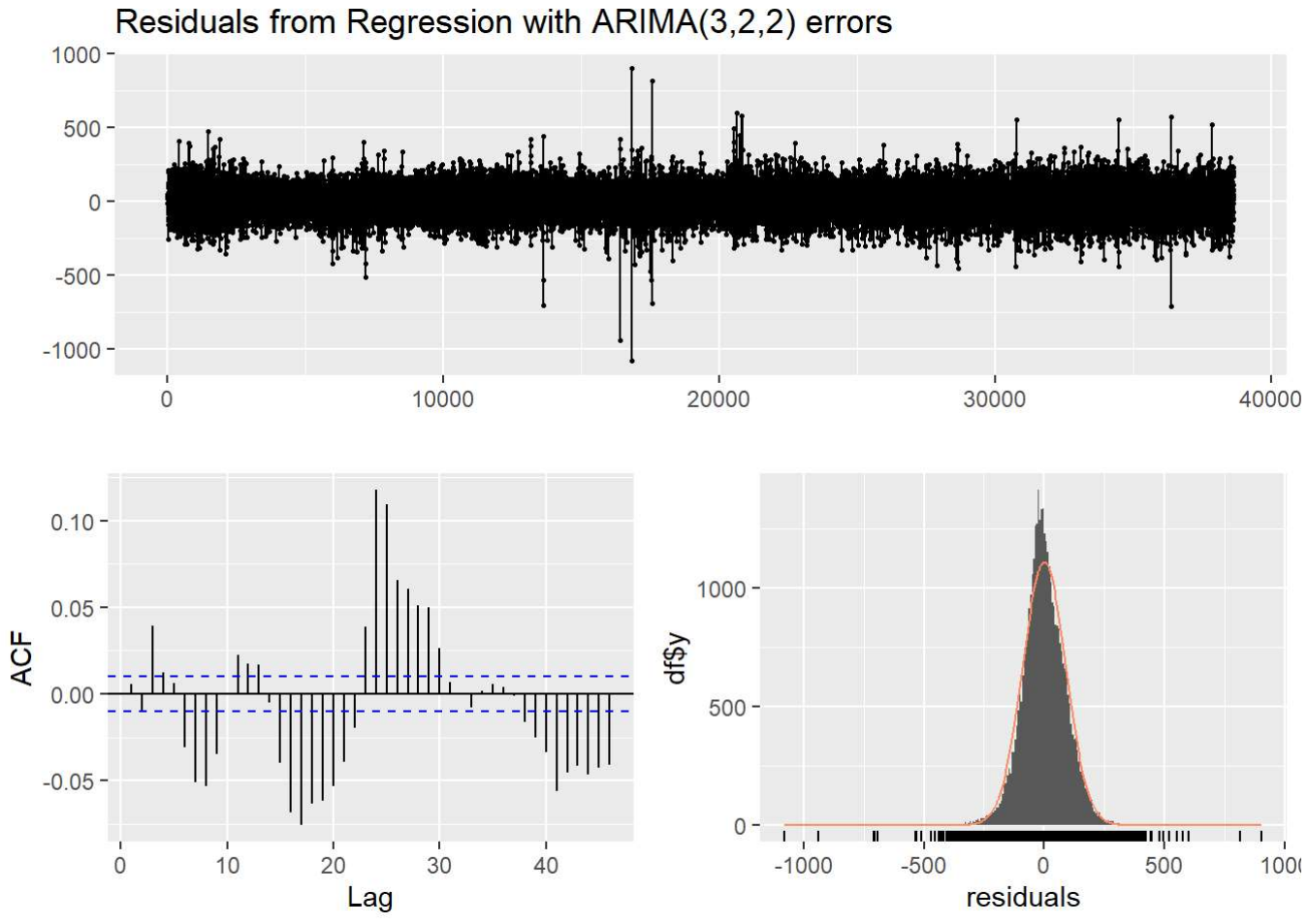
Model

```
#Arima model
manual_arima_model<- Arima(y_tr, order=c(3,2,2),xreg=X_tr)
summary(manual_arima_model)
```

```
## Series: y_tr
## Regression with ARIMA(3,2,2) errors
##
## Coefficients:
##          ar1      ar2      ar3      ma1     ma2  HourlyWetBulbTemperature
##       1.3593  -0.2603  -0.2348  -1.9505  0.9506                    3.8320
## s.e.  0.0053   0.0087   0.0050   0.0023  0.0023                    0.4098
##       HourlyStationPressure
##                   1116.7293
## s.e.                44.1988
##
## sigma^2 = 8029:  log likelihood = -228473.1
## AIC=456962.1   AICc=456962.1   BIC=457030.6
##
## Training set error measures:
##                       ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.006496278 89.59582 68.20864 -0.1170186 3.180756 0.7643966
##                     ACF1
## Training set 0.005892414
```

Residual Plot

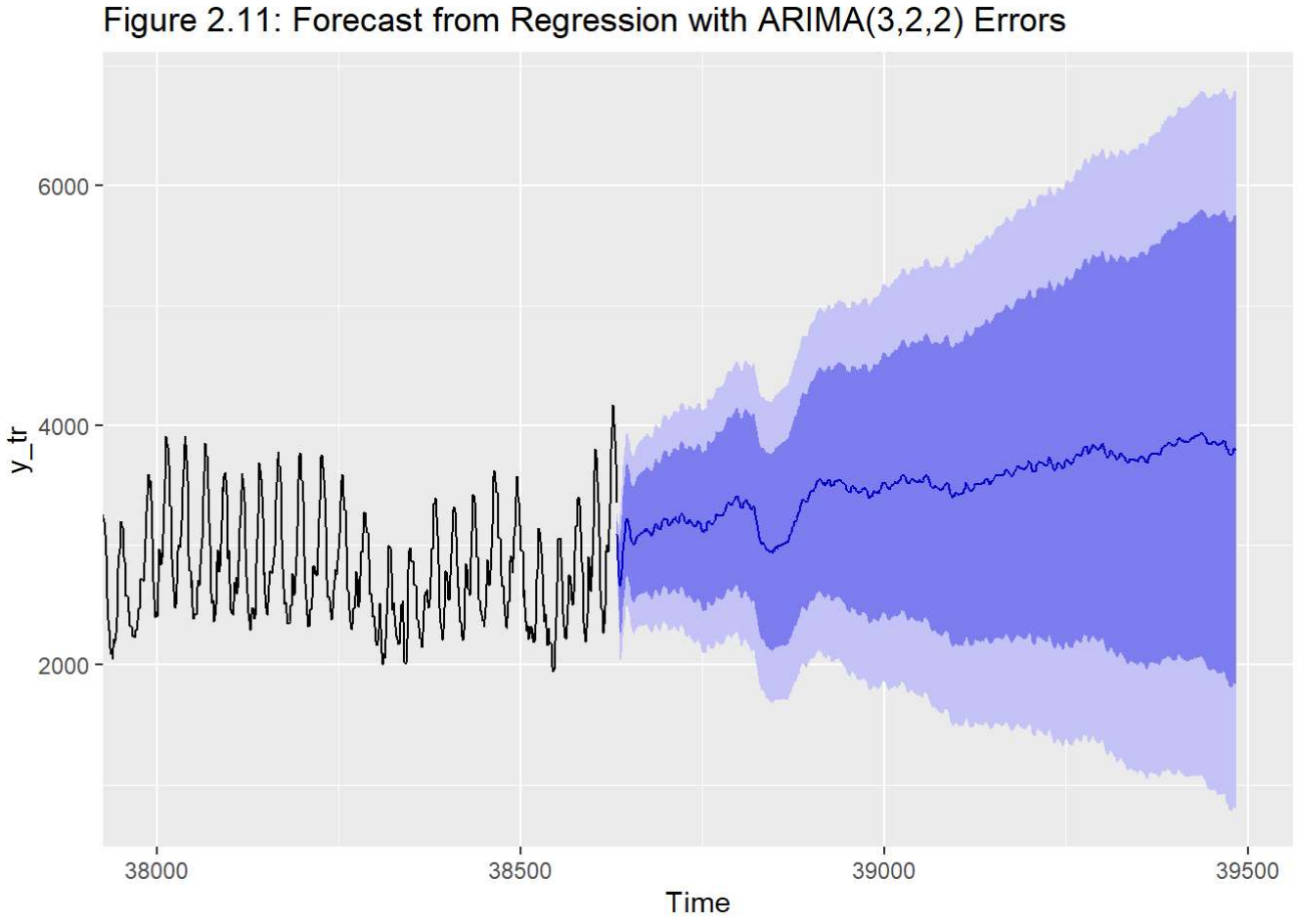
```
#check the residuals
checkresiduals(manual_arima_model)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(3,2,2) errors
## Q* = 364.27, df = 5, p-value < 2.2e-16
##
## Model df: 5.   Total lags used: 10
```

Forecast, Calculate RMSE, and Plot

```
manual_forecast<-forecast(manual_arima_model, xreg = X_val)

#RMSE
val_forecast_err <- SD_val$SDGE - manual_forecast$mean
manual_arima_rmse_val <- sqrt(mean(val_forecast_err^2))
manual_arima_rmse_tr <- sqrt(mean(manual_arima_model$residuals^2))

#plot
autoplot(manual_forecast, series = "Forecast") +
  coord_cartesian(xlim = c(38000, 39488)) +
  ggtitle("Figure 2.11: Forecast from Regression with ARIMA(3,2,2) Errors")
```

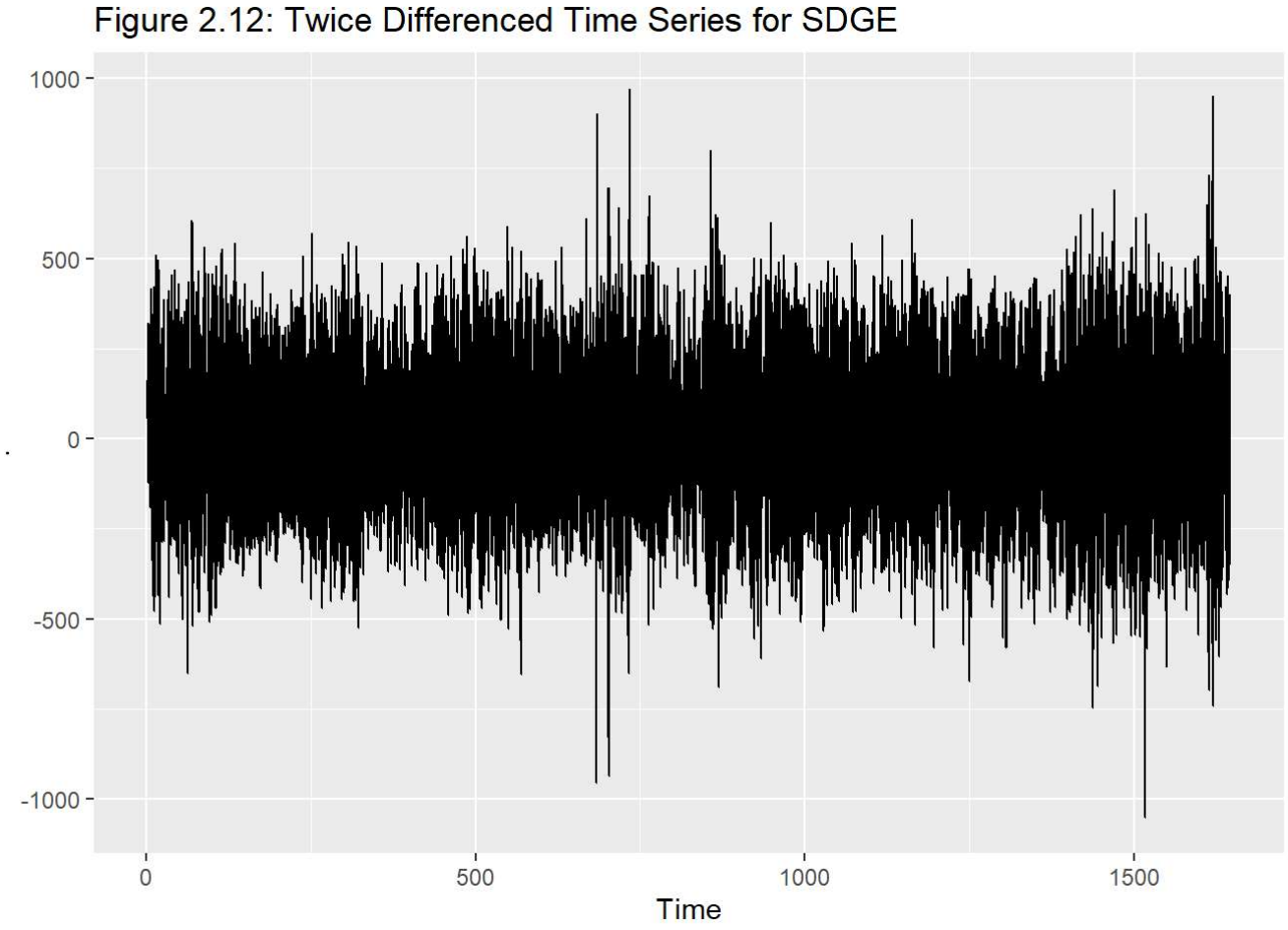


Manual ARIMA with Seasonality

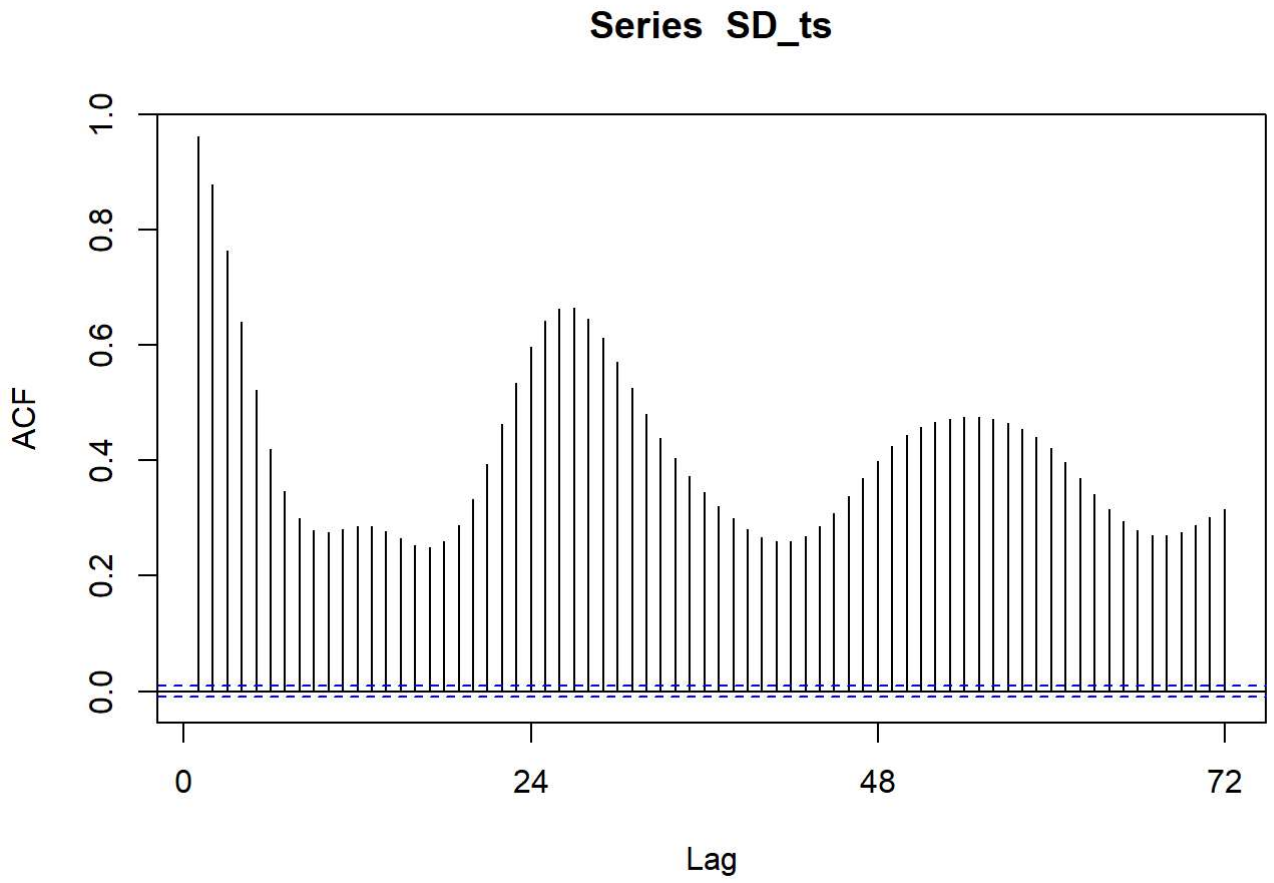
Create daily ts for easier seasonal eval

```
SD_ts <- ts(SD$SDGE, frequency = 24)
```

```
SD_ts %>% diff(lag = 24) %>% diff() %>% autoplot(main = "Figure 2.12: Twice Differenced Time Ser
ies for SDGE")
```

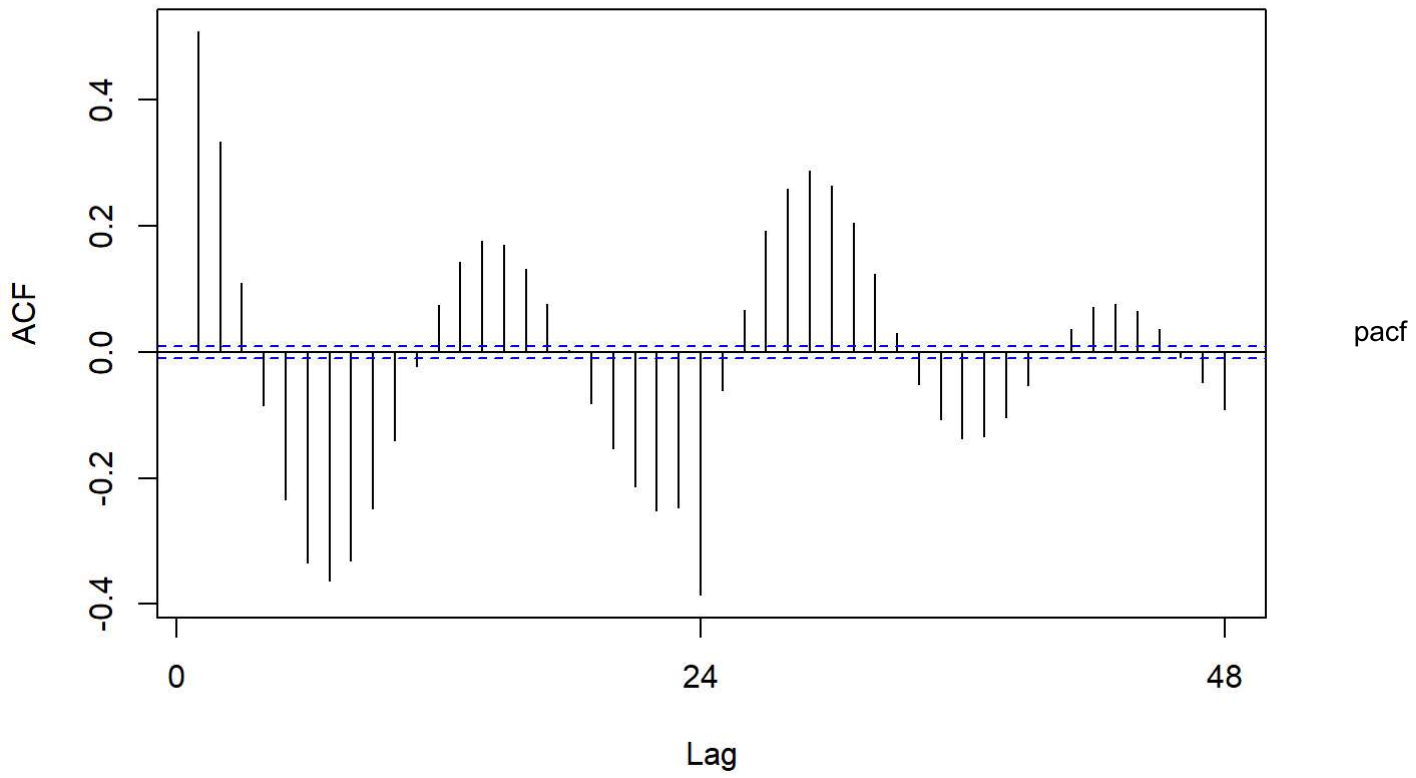



```
Acf(SD_ts, lag.max = 72)
```

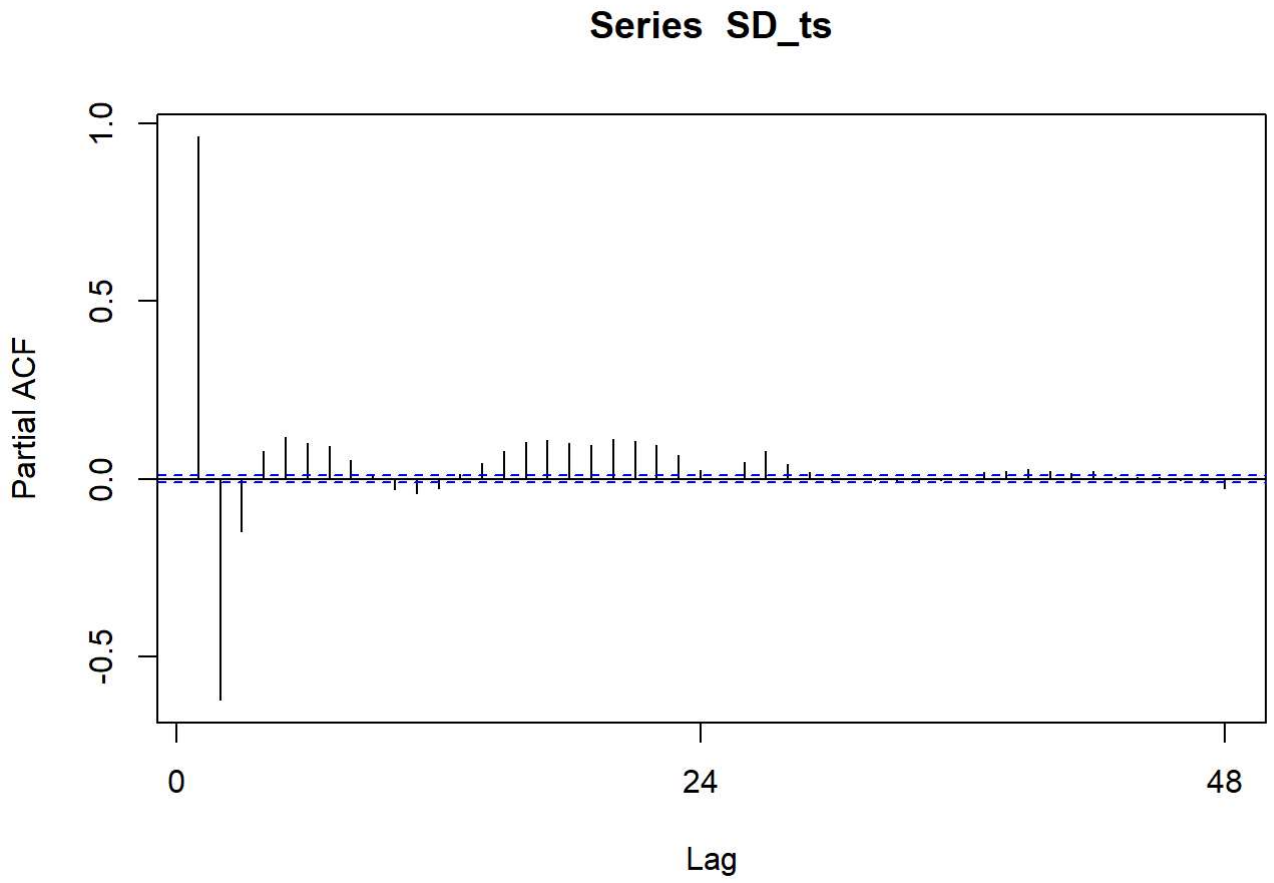



```
SD_ts %>% diff(lag = 24) %>% diff() %>% Acf(main = "Figure 2.13: Twice Differenced ACF Plot for
 SDGE")
```

Figure 2.13: Twice Differenced ACF Plot for SDGE

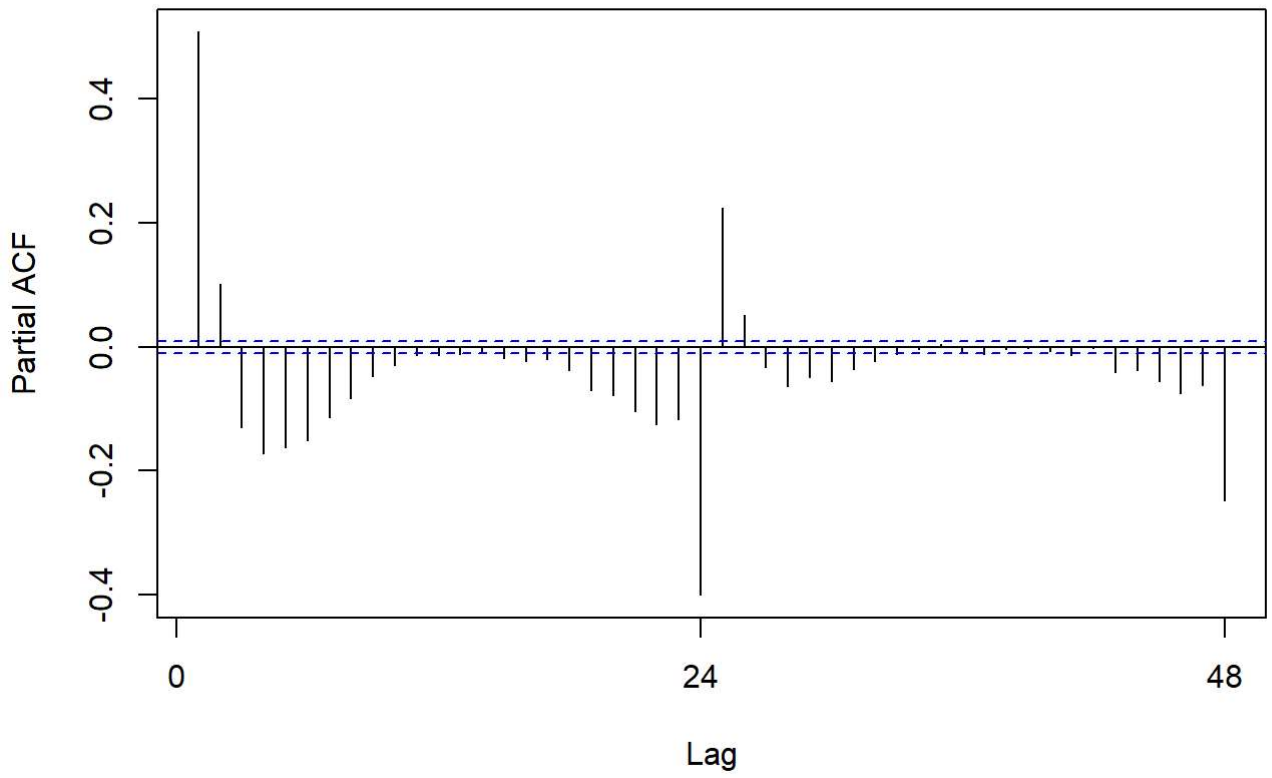



```
Pacf(SD_ts)
```




```
SD_ts %>% diff(lag = 24) %>% diff() %>% Pacf(main = "Figure 2.14: Twice Differenced PACF Plot fo
r SDGE")
```

Figure 2.14: Twice Differenced PACF Plot for SDGE

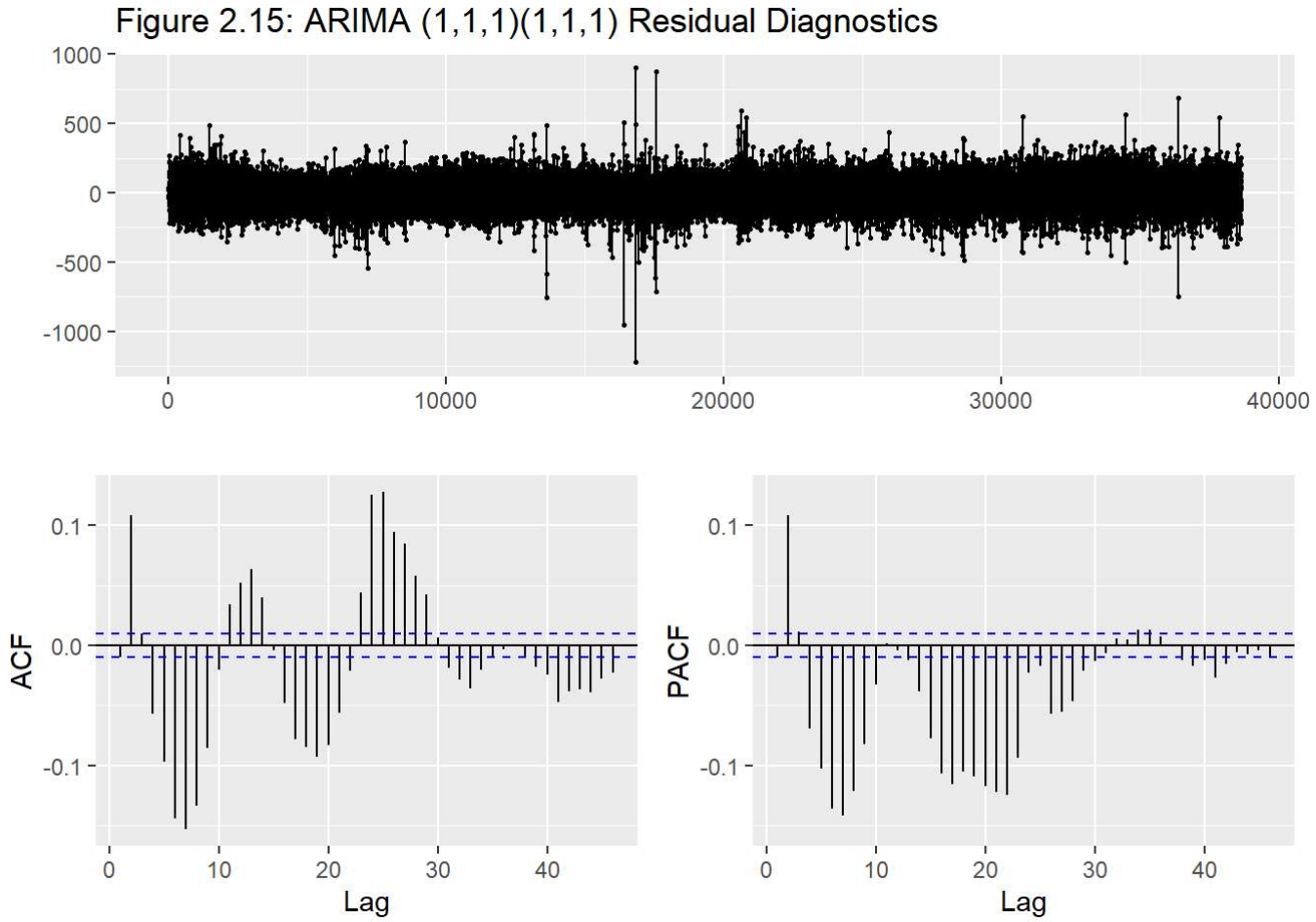


Fit model - Acf shows strong MA(1) for non-seasonal and seasonal on twice differenced chart, same story with Pacf for our AR(p) component. We took a non-seasonal and seasonal difference so we'll train a ARIMA(1,1,1)(1,1,1) model

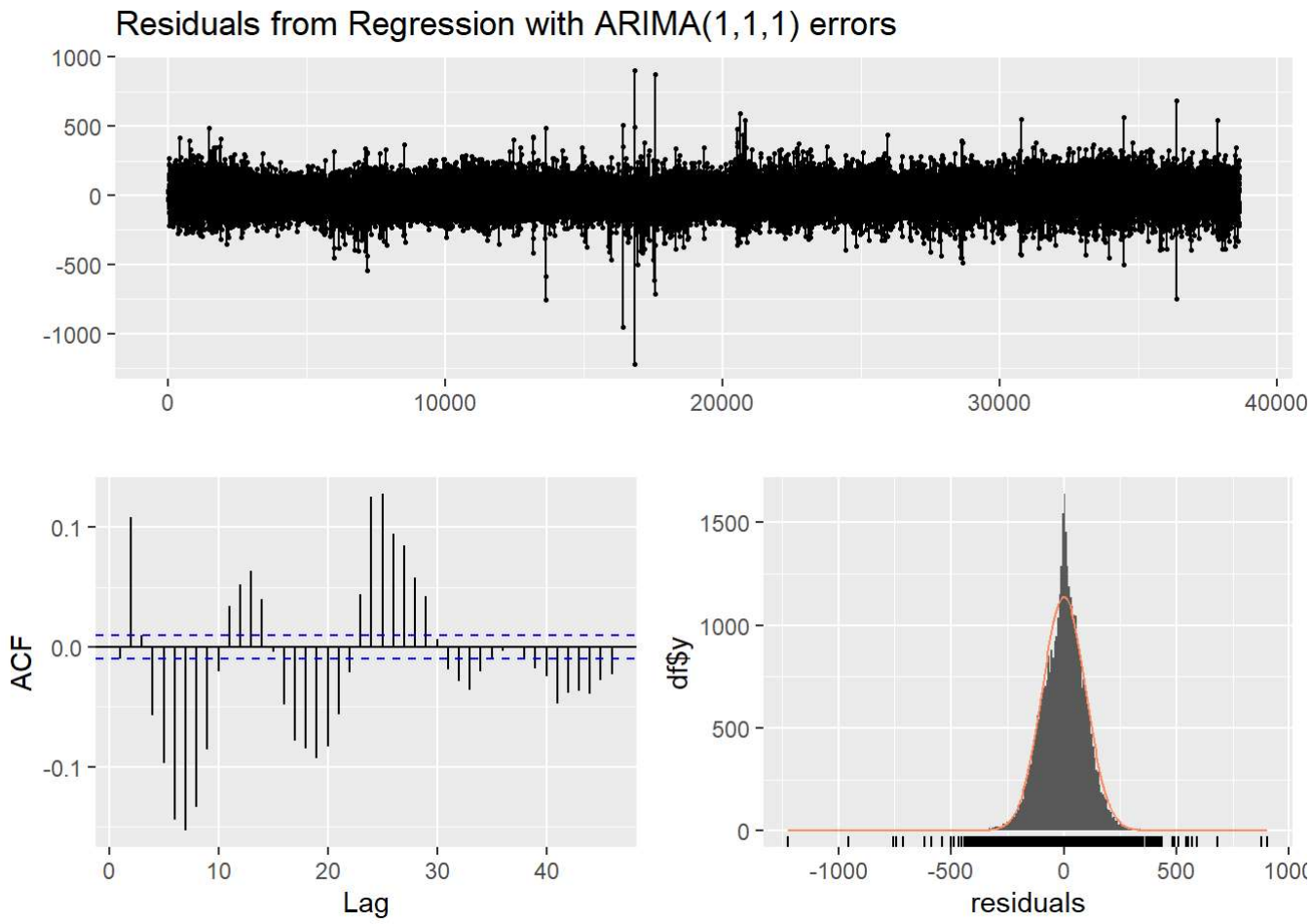
```
seasonal_arima <- Arima(y_tr, order=c(1,1,1), seasonal = c(1,1,1), xreg=X_tr)
summary(seasonal_arima)
```

```
## Series: y_tr
## Regression with ARIMA(1,1,1) errors
##
## Coefficients:
##          ar1      ma1  HourlyWetBulbTemperature  HourlyStationPressure
##       0.6354  -0.0906                    2.5471              1323.3350
## s.e.  0.0058   0.0070                    0.4177                45.7668
##
## sigma^2 = 9238:  log likelihood = -231185
## AIC=462380   AICc=462380   BIC=462422.8
##
## Training set error measures:
##                       ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.01162238 96.1095 73.41905 0.06089286 3.415765 0.8227883
##                       ACF1
## Training set -0.009848838
```

```
seasonal_arima %>% residuals() %>% ggtsdisplay(main = "Figure 2.15: ARIMA (1,1,1)(1,1,1) Residua
l Diagnostics")
```




```
seasonal_arima %>% checkresiduals()
```



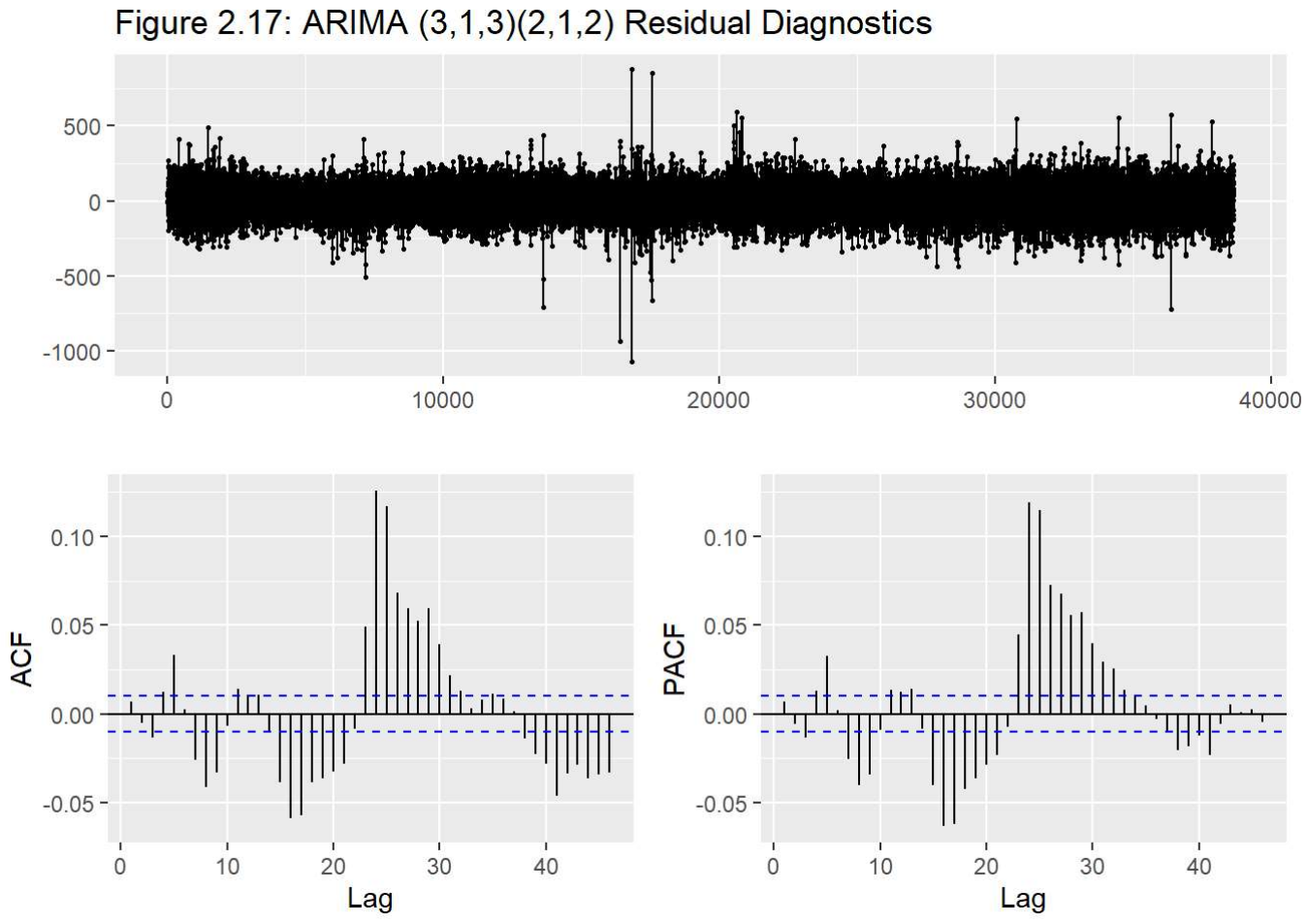

```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,1,1) errors
## Q* = 3654, df = 8, p-value < 2.2e-16
##
## Model df: 2.   Total lags used: 10
```

Continue to tune - Definitely didn't include enough components, we'll keep playing around until we find good values. (3,1,3) looks good for non-seasonal components.

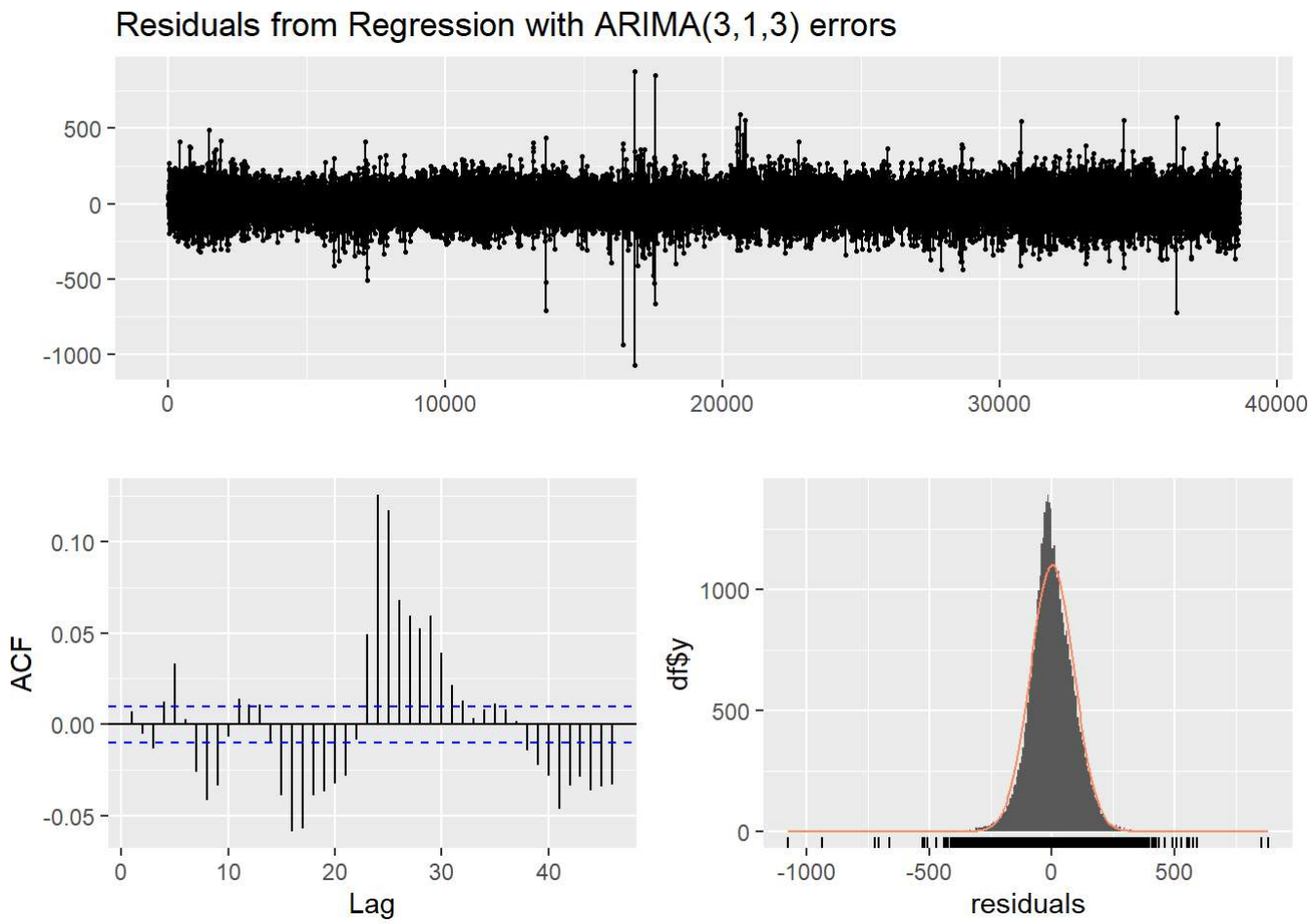
```
seasonal_arima_tuned <- Arima(y_tr, order=c(3,1,3), seasonal = c(2,1,2), xreg=X_tr)
summary(seasonal_arima_tuned)
```

```
## Series: y_tr
## Regression with ARIMA(3,1,3) errors
##
## Coefficients:
##          ar1      ar2     ar3      ma1     ma2      ma3
##       2.2965  -1.8639  0.5242  -1.8953  1.2200  -0.3198
## s.e.  0.0194   0.0339  0.0164   0.0202  0.0287   0.0101
##       HourlyWetBulbTemperature  HourlyStationPressure
##                         4.5467               400.5317
## s.e.                    0.4046                38.9464
##
## sigma^2 = 7882:  log likelihood = -228118.6
## AIC=456255.3   AICc=456255.3   BIC=456332.3
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.1581167 88.77287 67.47707 -0.1363844 3.145751 0.7561981
##                      ACF1
## Training set 0.007017214
```

```
seasonal_arima_tuned %>% residuals() %>% ggtsdisplay(main = "Figure 2.17: ARIMA (3,1,3)(2,1,2) Residual Diagnostics")
```



```
seasonal_arima_tuned %>% checkresiduals()
```

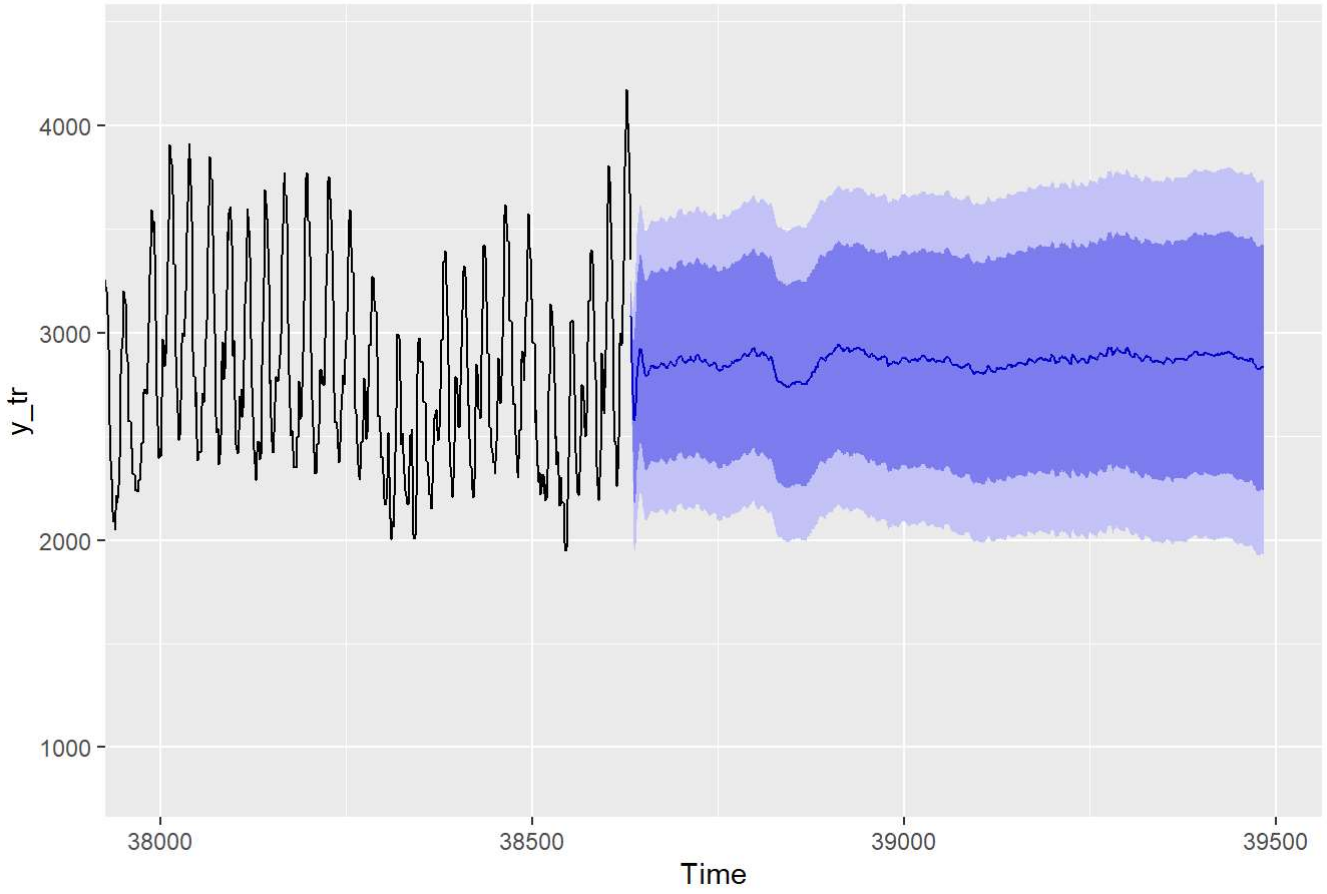



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(3,1,3) errors
## Q* = 195.89, df = 4, p-value < 2.2e-16
##
## Model df: 6.   Total lags used: 10
```

Forecast, Calculate RMSE, and Plot

```
seasonal_forecast <- forecast(seasonal_arima_tuned, xreg = X_val)

#RMSE
val_forecast_err <- SD_val$SDGE - seasonal_forecast$mean
seasonal_arima_rmse_val <- sqrt(mean(val_forecast_err^2))
seasonal_arima_rmse_tr <- sqrt(mean(seasonal_arima_tuned$residuals^2))

#plot
autoplot(seasonal_forecast, series = "Forecast") +
  coord_cartesian(xlim = c(38000, 39488)) +
  ggtitle("Figure 2.19: Forecasts for ARIMA(3,1,3)(2,1,2) Model")
```

Figure 2.19: Forecasts for ARIMA(3,1,3)(2,1,2) Model



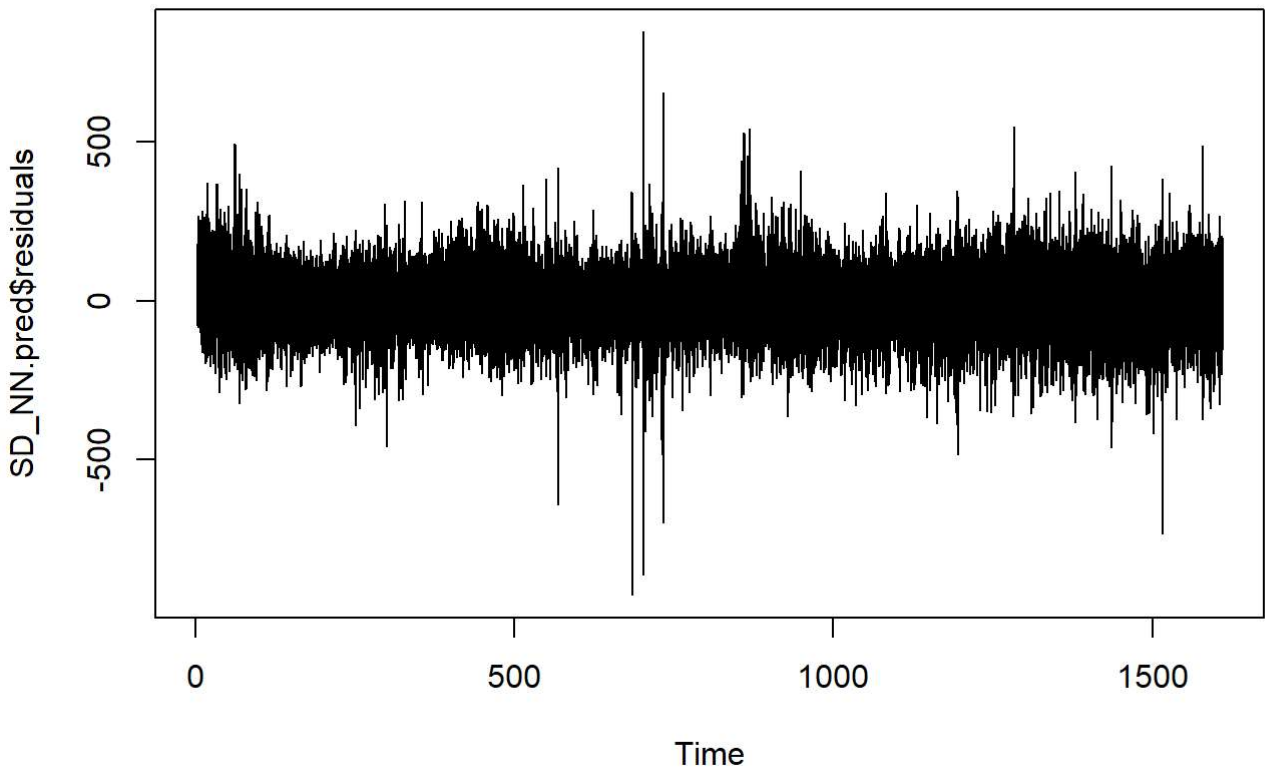
Neural Network

```
#size= (p+p+1)/2 = (3+2+1)/2 =3
set.seed(123)
SD_NN <- nnetar(y_tr_ts, p=3, P=2, size =3, xreg = X_tr)
```

```
#prediction
SD_NN.pred<-forecast(SD_NN, xreg = X_val)

# Not sure why but this prints out the entire data frame of forecasts, won't output to reduce ap
pendix clutter.
#summary(SD_NN.pred)
```

```
# Plot the errors for the training period
plot(SD_NN.pred$residuals, main = "Figure 2.20: Neural Net Residual Plot for Training Period")
```

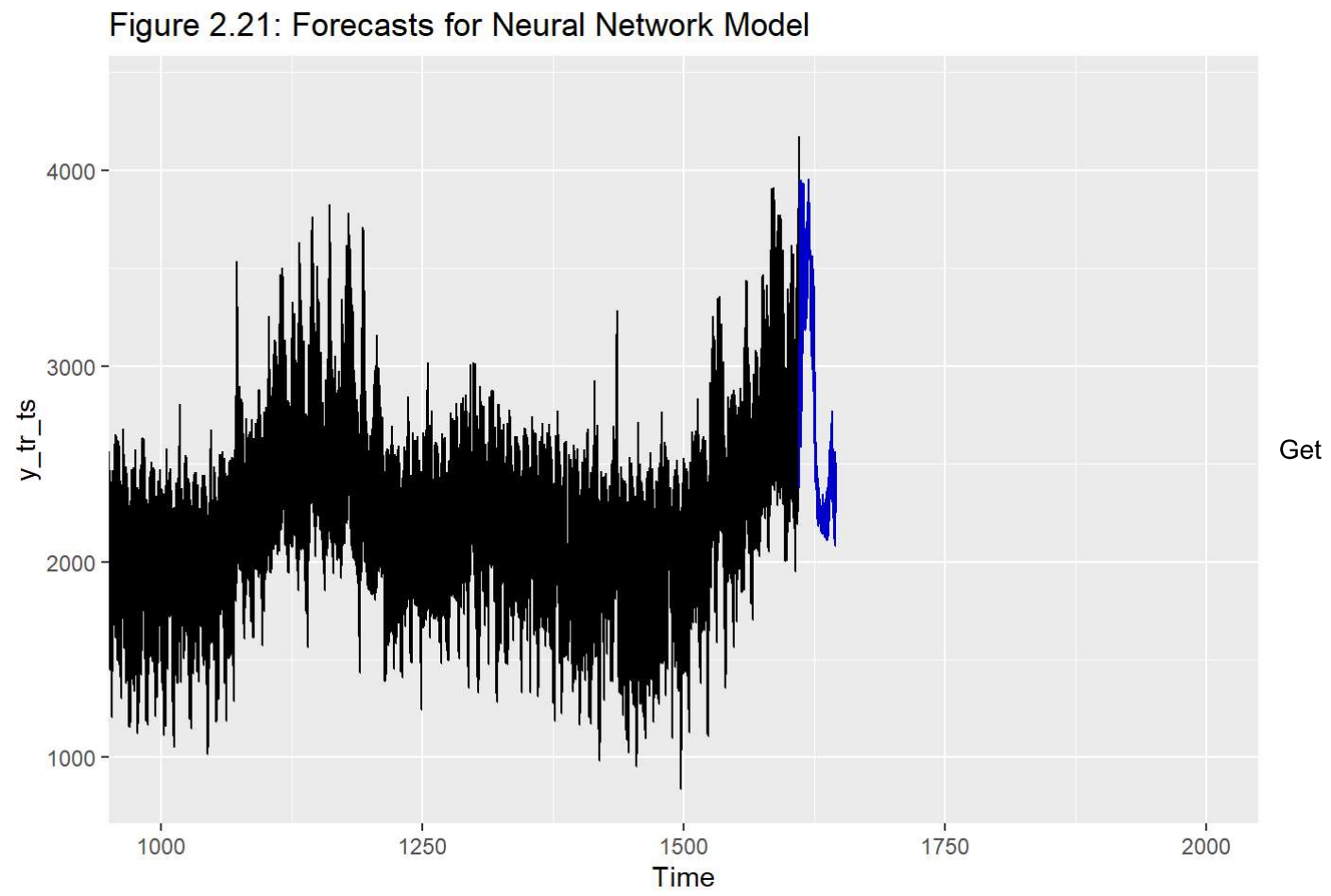
Figure 2.20: Neural Net Residual Plot for Training Period




```
#Accuracy & Plot
accuracy(SD_NN.pred, y_test)
```

```
##                       ME      RMSE       MAE        MPE      MAPE      MASE
## Training set  0.05947569  84.47837  63.86467 -0.1675212  2.996544 0.7157148
## Test set     59.74111315 430.80023 337.86279  0.7793615 11.949428 3.7863409
##                      ACF1
## Training set -0.001513646
## Test set               NA
```

```
autoplot(SD_NN.pred, series = "Forecast") +
  coord_cartesian(xlim = c(1000, 2000)) +
  ggtitle("Figure 2.21: Forecasts for Neural Network Model")
```



Get metrics

```
nnetar_rmse_tr <- sqrt(mean(SD_NN$residuals^2, na.rm = TRUE))
val_forecast_err <- SD_val$SDGE - SD_NN.pred$mean
nnetar_rmse_val <- sqrt(mean(val_forecast_err^2))
```

avNNet

Not included in final but leaving for future reference/discussion

```
modelFit <- avNNet(X_tr,
                   y_tr,
                   size = 5,
                   linout = TRUE,
                   trace = FALSE)
summary(modelFit)

nnpred1 <- predict(modelFit, X_val)
```

```
accuracy(nnpred1, y_test)
```

Regression

Also not included in final but left for reference.

Based on our time series, it looks like we have additive seasonality with no trend.

```
SD_reg_addseason <- tslm(y_tr_ts ~ season)
SD_reg_addseason_pred <- forecast(SD_reg_addseason, h = length(y_val_ts))
```

Forecast, Calculate RMSE, and Plot

```
reg_forecast <- forecast(SD_reg_addseason, h=length(y_test))

#RMSE
val_forecast_err <- SD_val$SDGE - reg_forecast$mean
reg_rmse_val <- sqrt(mean(val_forecast_err^2))
reg_rmse_tr <- sqrt(mean(SD_reg_addseason$residuals^2))

#plot
autoplot(reg_forecast, series = "Forecast") +
  coord_cartesian(xlim = c(1550, 1650)) +
  ggtitle("Figure 2.7: Forecasts from Linear Regression Model")
```

Section 3 - Results

Create Results df

```
Model <- c("Naive", "2-Hour MA", "Auto ARIMA", "Manual ARIMA", "Seasonal ARIMA", "ETS", "Neural
 Net.")
Tr_RMSE <- c(Naive_rmse_tr, MA_rmse_tr, auto_arima_rmse_tr, manual_arima_rmse_tr, seasonal_arima
_rmse_tr, ets_rmse_tr, nnetar_rmse_tr)
Val_RMSE <- c(Naive_rmse_val, MA_rmse_val, auto_arima_rmse_val, manual_arima_rmse_val, seasonal_
arima_rmse_val, ets_rmse_val, nnetar_rmse_val)

Results_df <- data.frame(Model, Tr_RMSE, Val_RMSE)

Results_df <- Results_df %>% mutate(BaselineComp_tr = ifelse(Tr_RMSE == Naive_rmse_tr, "Baselin
e", ifelse(Tr_RMSE < Naive_rmse_tr, "Better", "Worse")))
Results_df <- Results_df %>% mutate(BaselineComp_val = ifelse(Val_RMSE == Naive_rmse_val, "Basel
ine", ifelse(Val_RMSE < Naive_rmse_val, "Better", "Worse")))

Results_df <- Results_df %>% mutate_if(is.character, as.factor)
```

Plot Training RMSE's

```
ggplot(Results_df, aes(x = reorder(Model, -Tr_RMSE), y = Tr_RMSE, color = BaselineComp_tr)) +
  geom_segment(aes(x=reorder(Model, -Tr_RMSE), xend = reorder(Model, -Tr_RMSE), y = 0, yend = Tr
_RMSE)) +
  scale_color_manual(values = c("dodgerblue2", "green3", "firebrick2")) +
  geom_point(size = 9) + coord_flip() + labs(color = "Comparison to Baseline") +
  ylab("RMSE for Training Data (MWh)") + ggtitle("Figure 3.1: Model Results for Training Data")
 + xlab("Model") + geom_text(aes(label = round(Tr_RMSE)), color = "black", size = 2.5, fontface
 = "bold")
```

Plot Validation RMSE's - Reordering isn't working when I flip coordinates. Need to find workaround.

```
ggplot(Results_df, aes(x = reorder(Model, -Val_RMSE), y = Val_RMSE, color = BaselineComp_val)) +
  geom_segment(aes(x=reorder(Model, -Val_RMSE), xend = reorder(Model, -Val_RMSE), y = 0, yend =
 Val_RMSE)) +
  scale_color_manual(values = c("dodgerblue2", "green3", "firebrick2")) +
  geom_point(size = 9) + coord_flip() + labs(color = "Comparison to Baseline") +
  ylab("RMSE for Validation Data (MWh)") + ggtitle("Figure 3.2: Model Results for Validation Dat
a") + xlab("Model") + geom_text(aes(label = round(Val_RMSE)), color = "black", size = 2.5, fontf
ace = "bold")
```

Figure 3.2: Model Results for Validation Data