

Forecasting San Diego Power Consumption

Hunter Blum, Mackenzie Carter, Saba Alemayehu

2022-11-06

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. Usually 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(getContext()$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 03:00:00" "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 ...  
## $ 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				

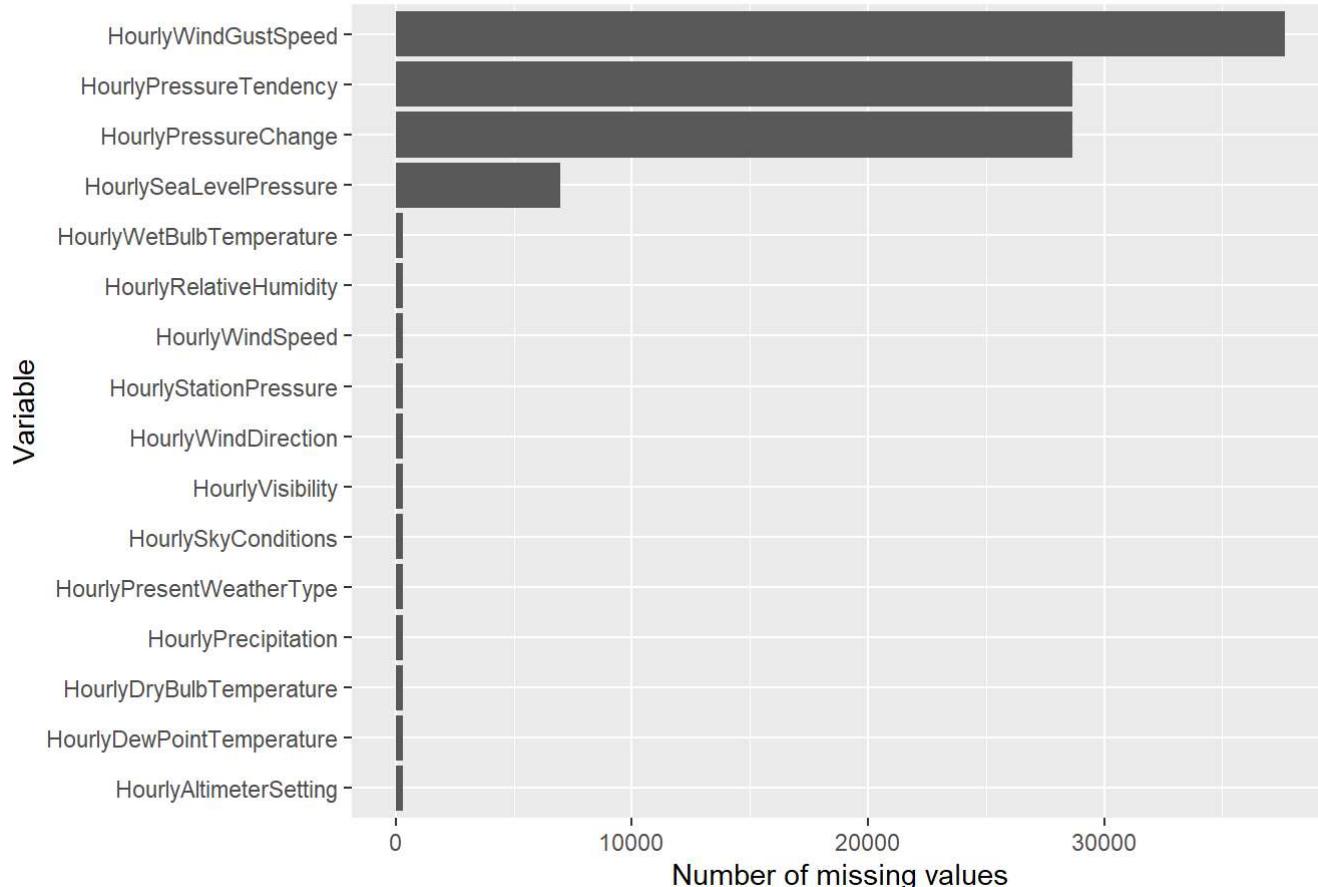
```
## HourlyVisibility*      0.02
## HourlyWetBulbTemperature 0.03
## HourlyWindDirection*    0.07
## HourlyWindGustSpeed     0.11
## HourlyWindSpeed         0.02
```

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 Variable') +
  coord_flip()
```

Figure 1.1: Missing Values by Variable



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, HourlyWindGustSpeed))

# 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

```

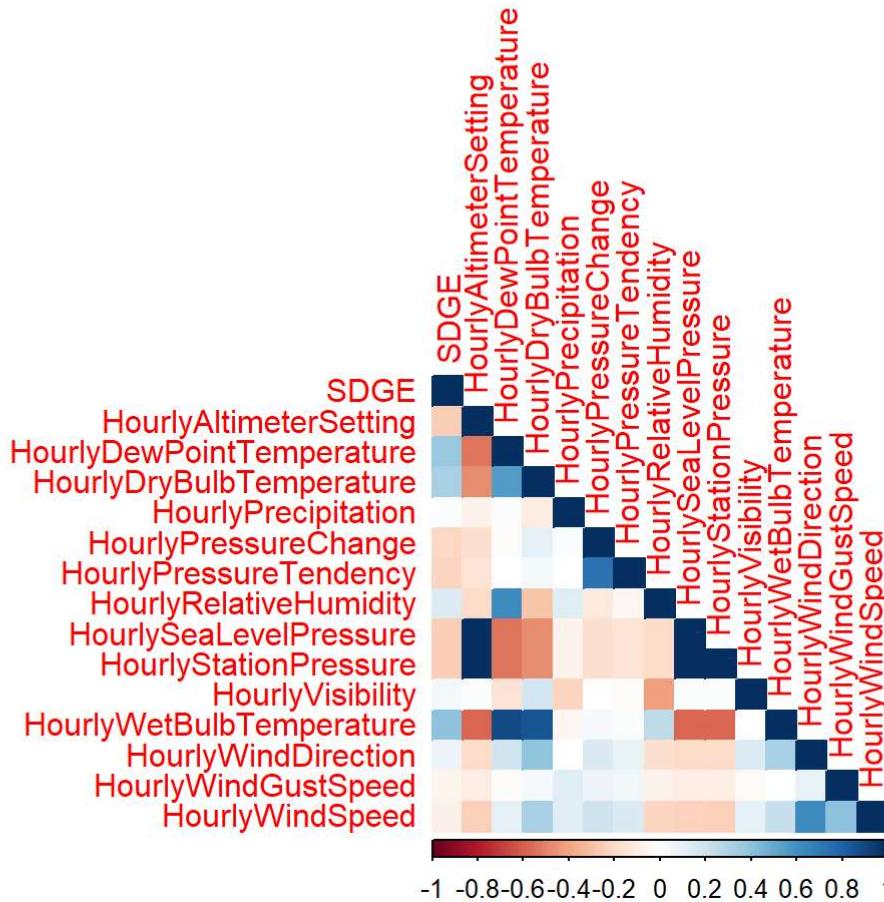
The grubbs test shows our max value for wind speed, 33, is an outlier. This makes sense because wind speed ranges in the single digits- it is likely that 33 was a typo.

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")
```

Figure 1.2. CORRELATION BETWEEN NUMERIC FEATURES

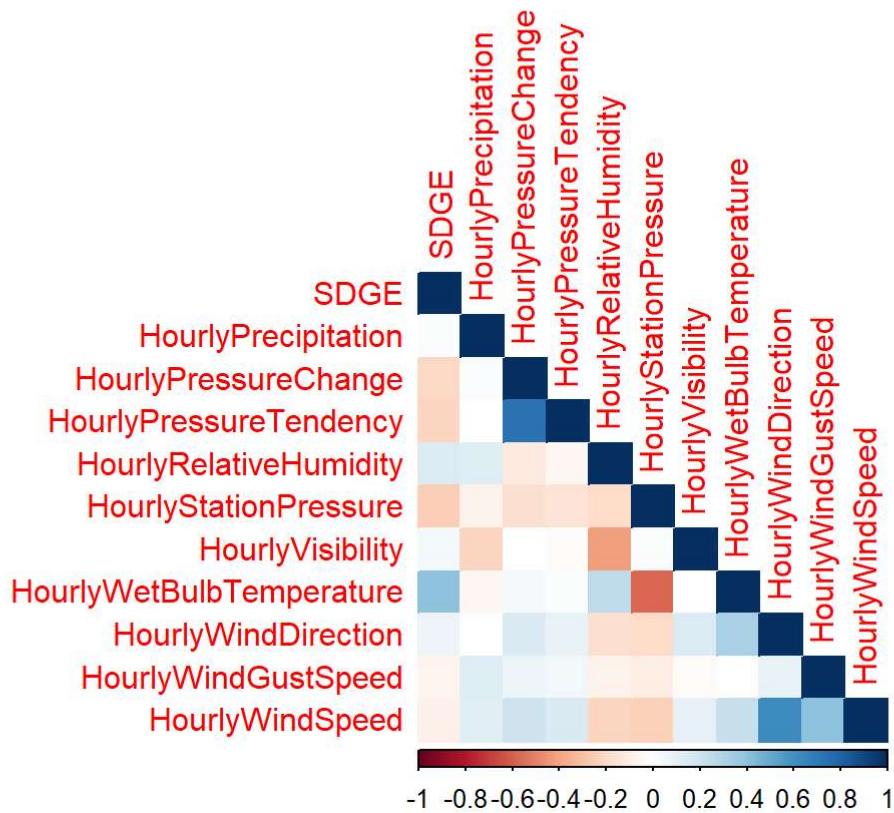


As

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.

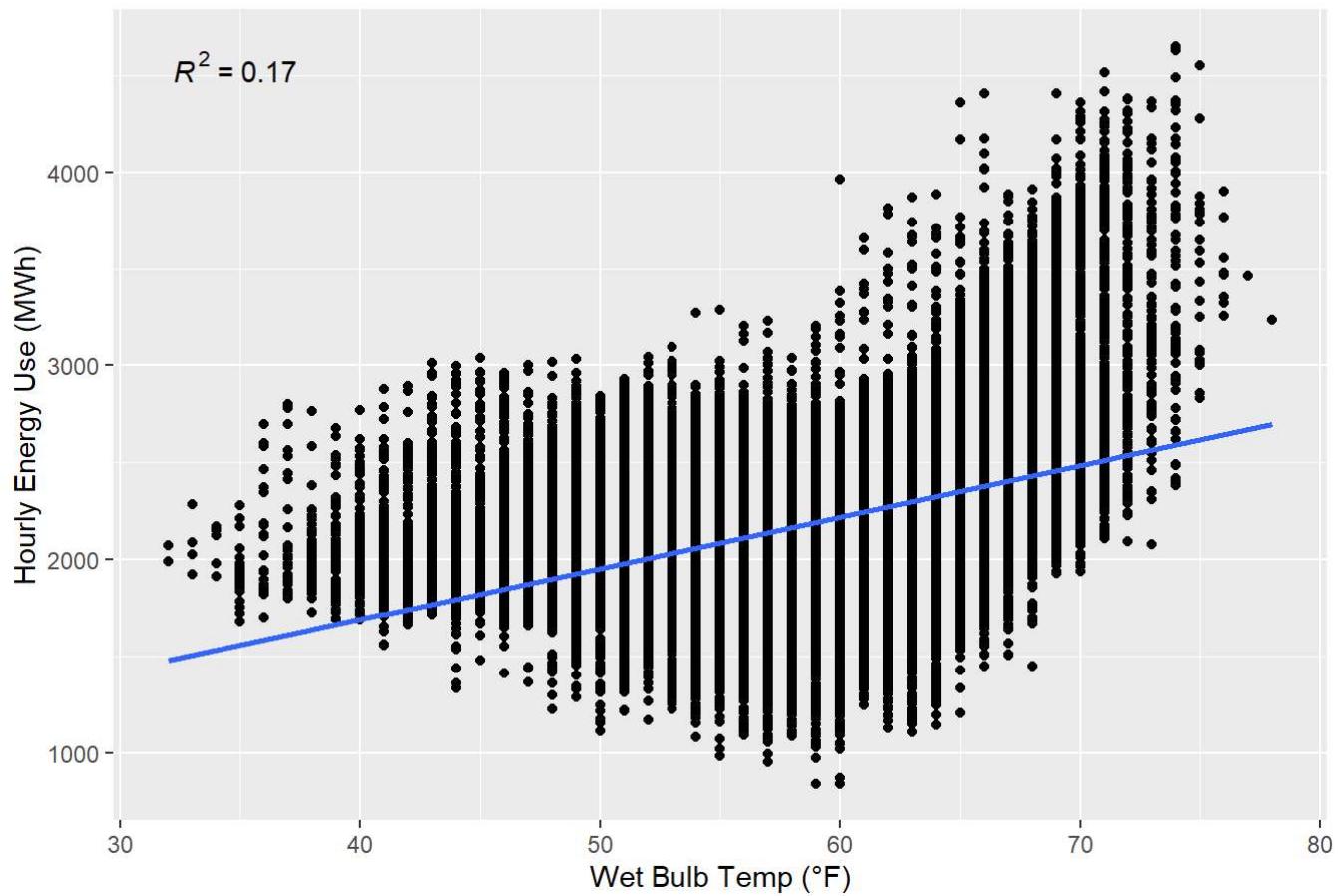
Correlations below look much better

```
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

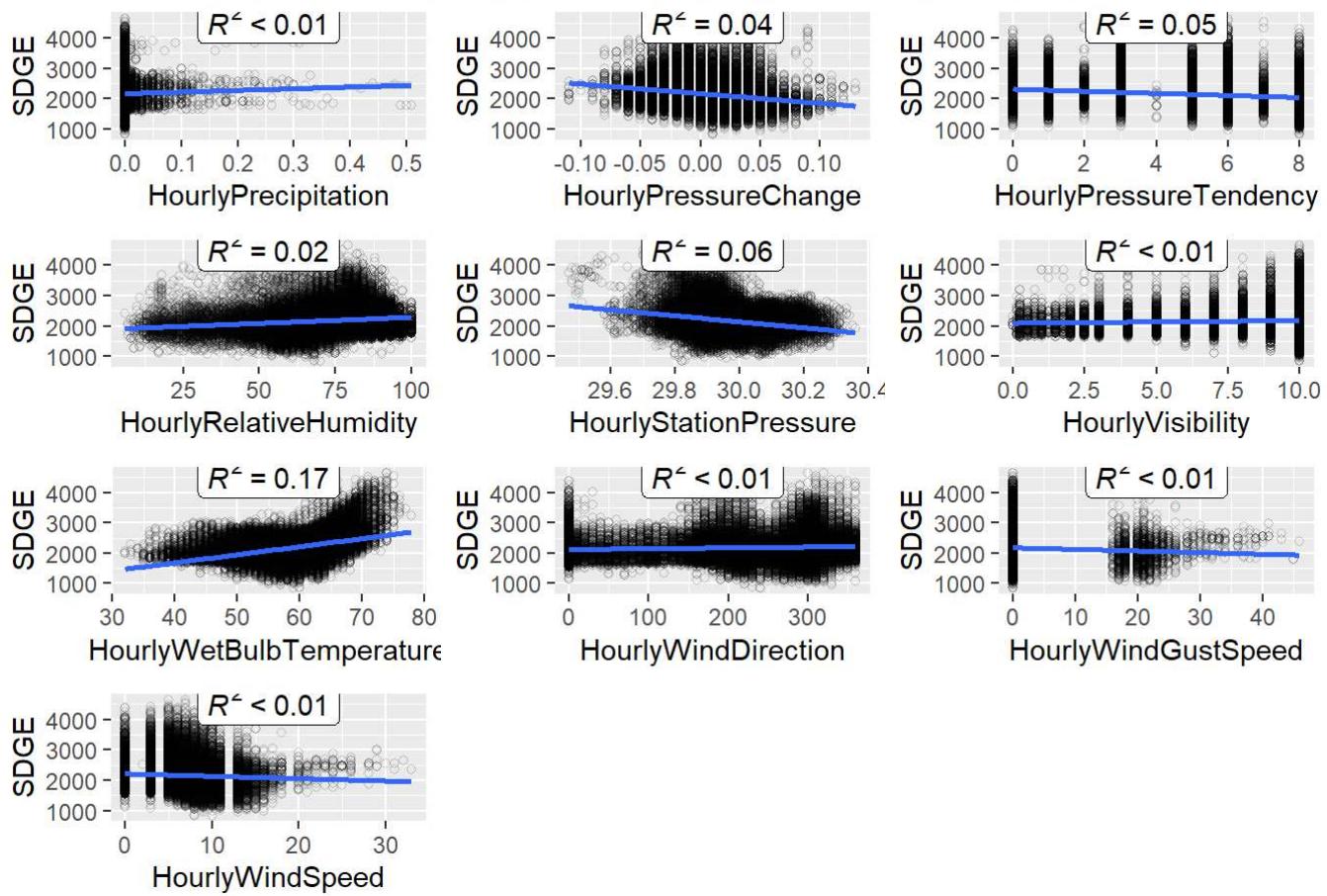


Scatter plots for numeric features compared to hourly energy use - may need to explore non-linear options.

```
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]))
}

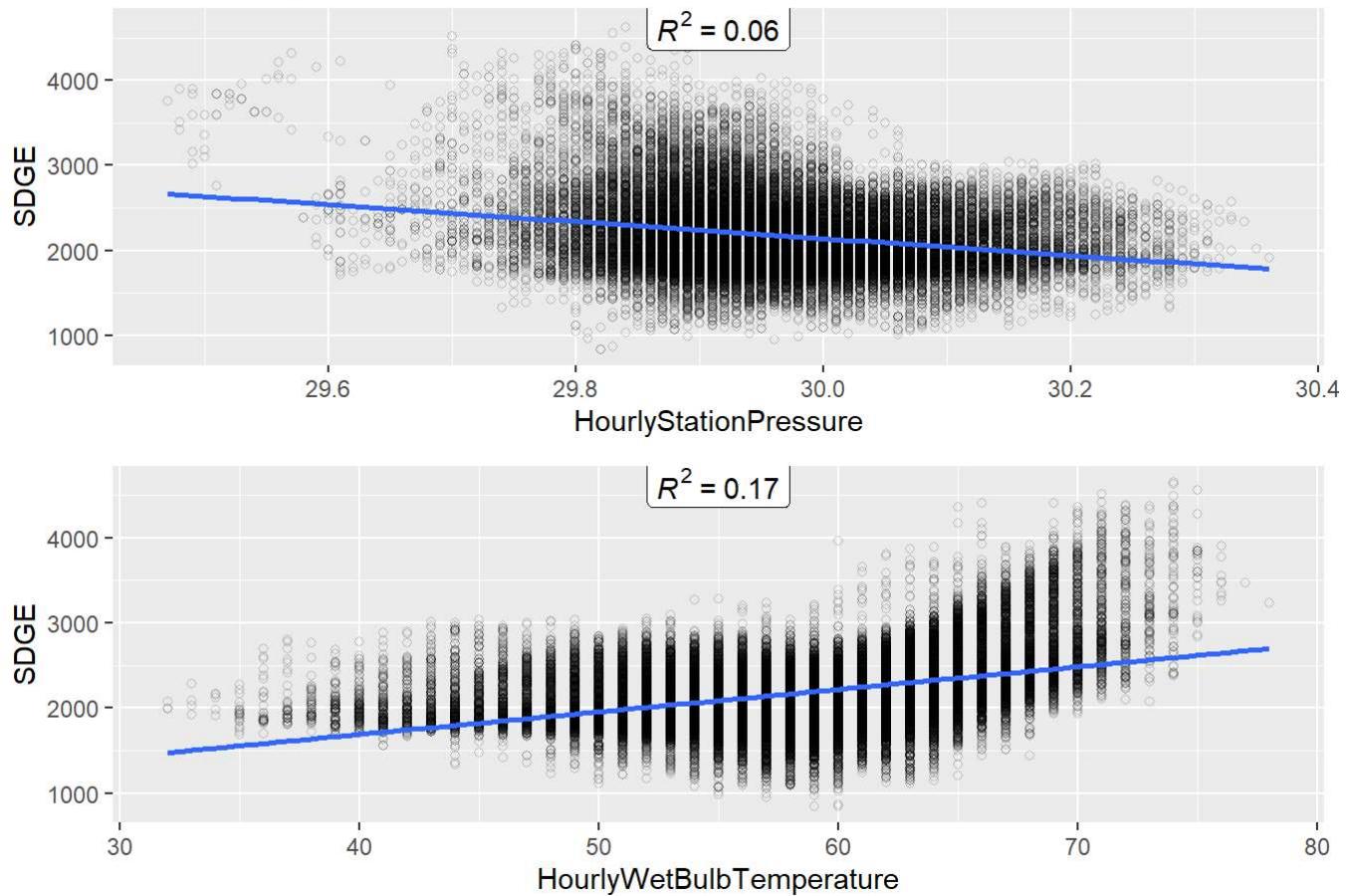
num_cols <- c(2:11)
Scatterplots <- lapply(num_cols, ScatPlotter.SD)
grid.arrange(grobs = Scatterplots, top = "Figure 1.5: Scatterplots for Each Numeric Predictor v
s. SDGE")
```

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

```

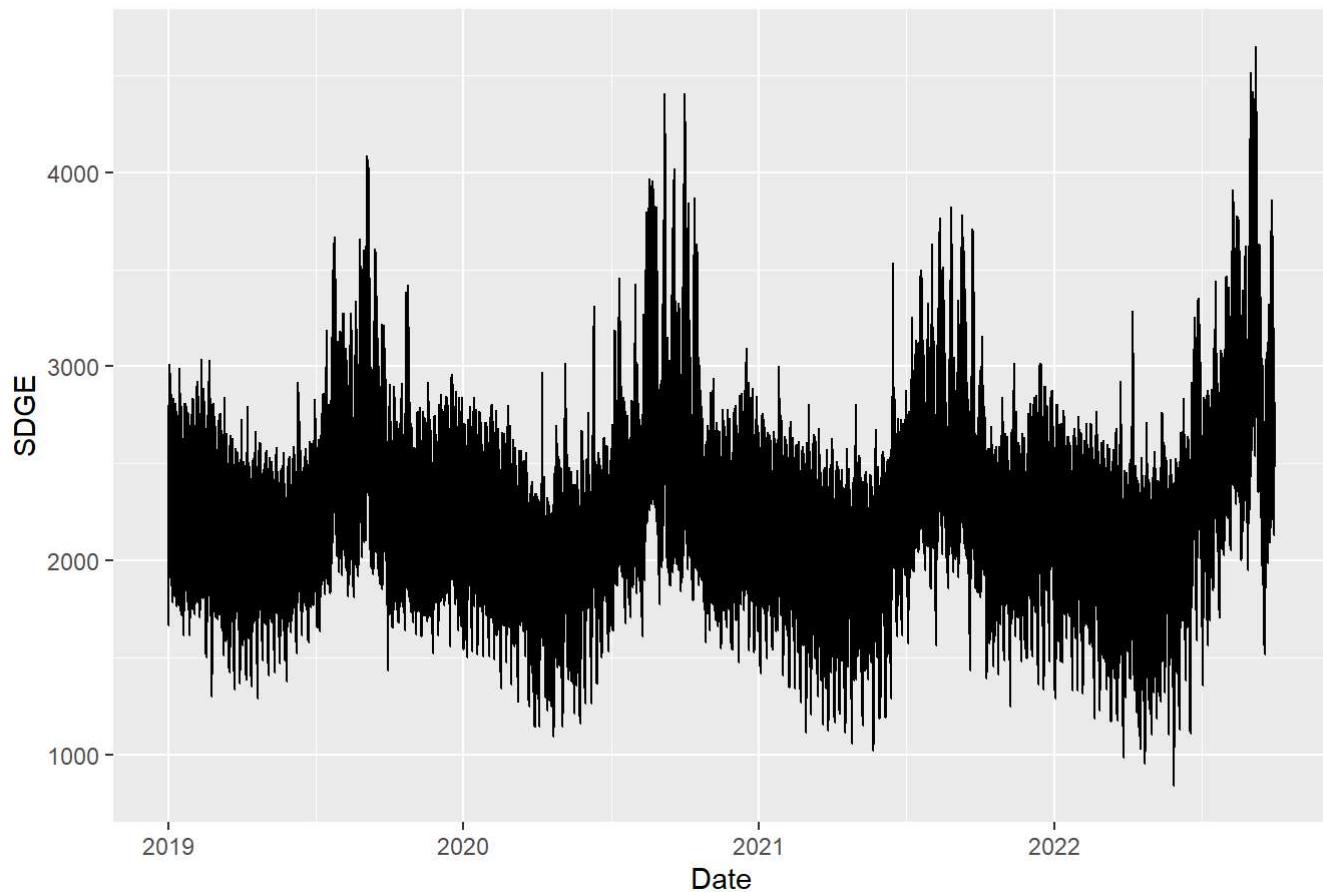
num_wdate <- num2
num_wdate$Date <- SD$Date

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)

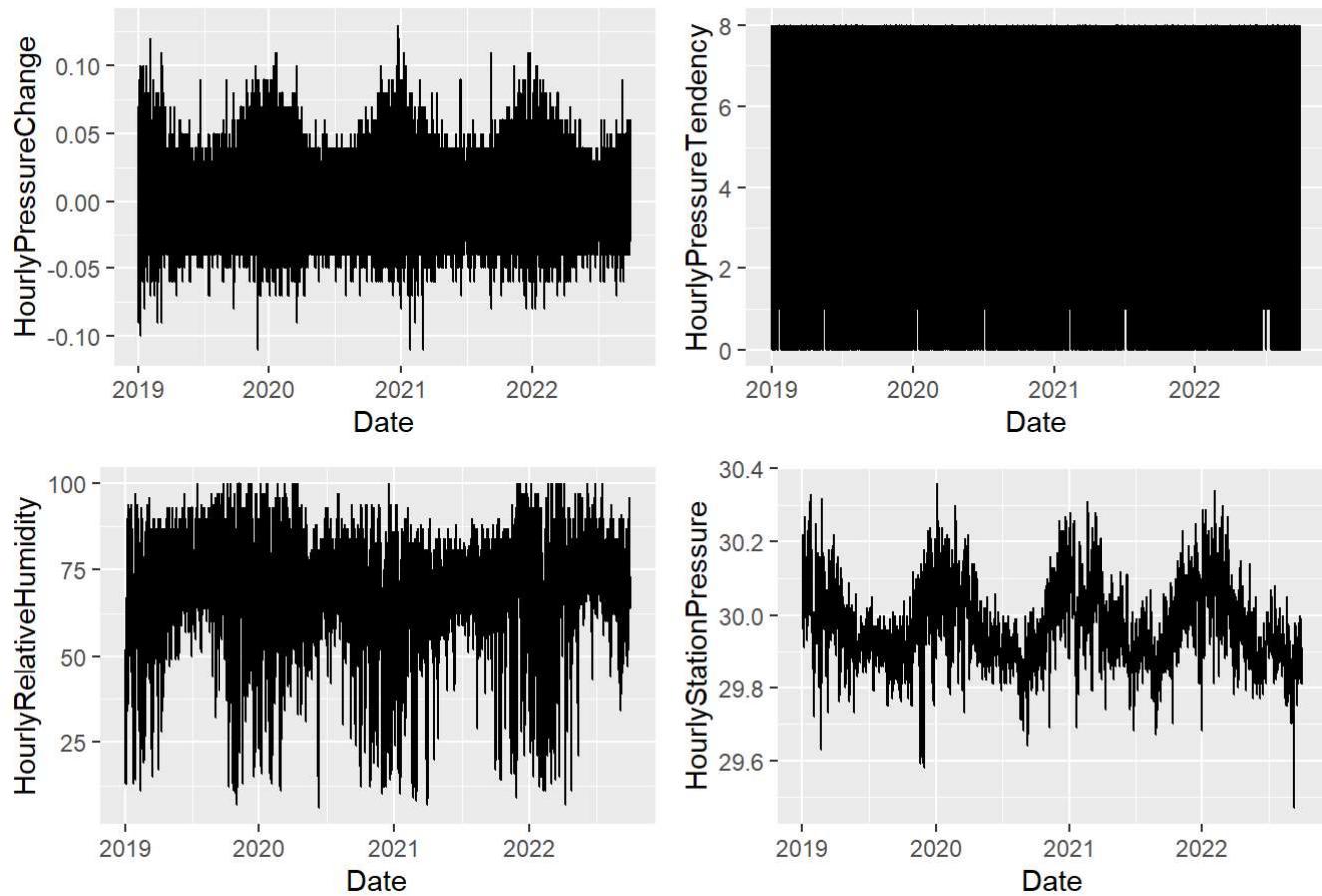
plot(TS_target)

```

Figure 1.7: Time Series for SDGE

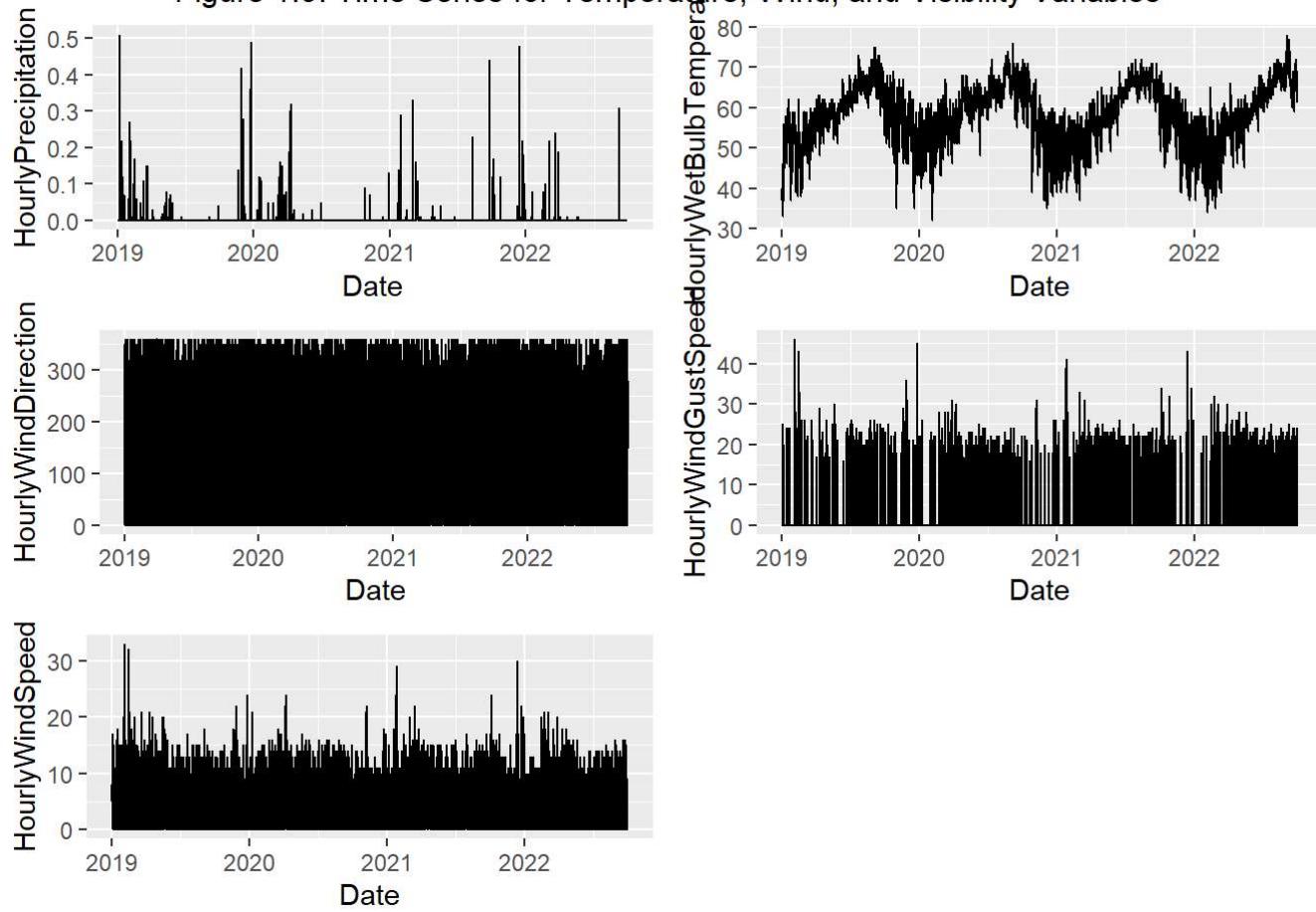
```
grid.arrange(grobs = TS_PHP, top = "Figure 1.8: Time Series for Pressure, Humidity, and Precipitation Variables")
```

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



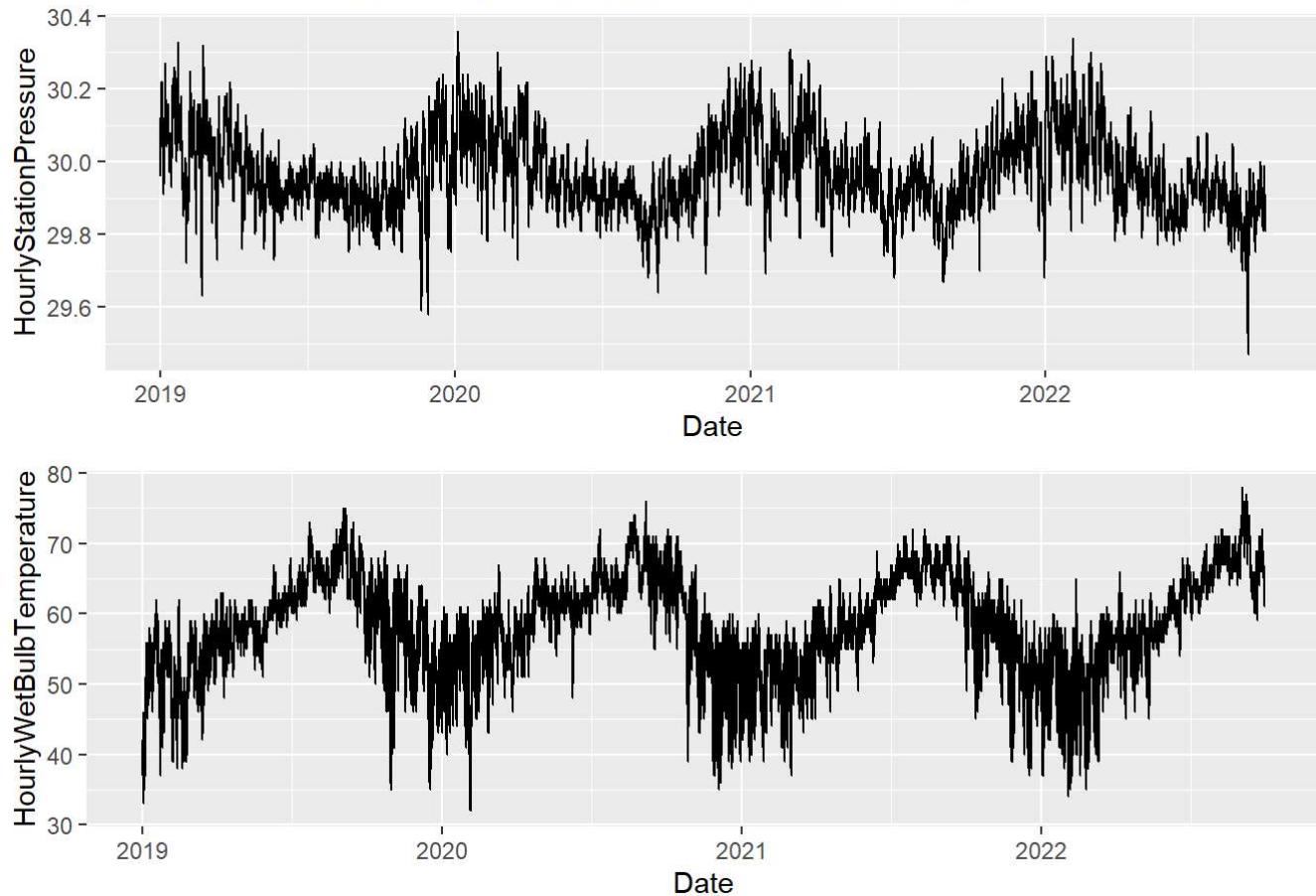
```
grid.arrange(grobs = TS_TWV, top = "Figure 1.9: Time Series for Temperature, Wind, and Visibility Variables")
```

Figure 1.9: Time Series for Temperature, Wind, and Visibility Variables



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

Figure 1.10 Time Series for Selected Predictors

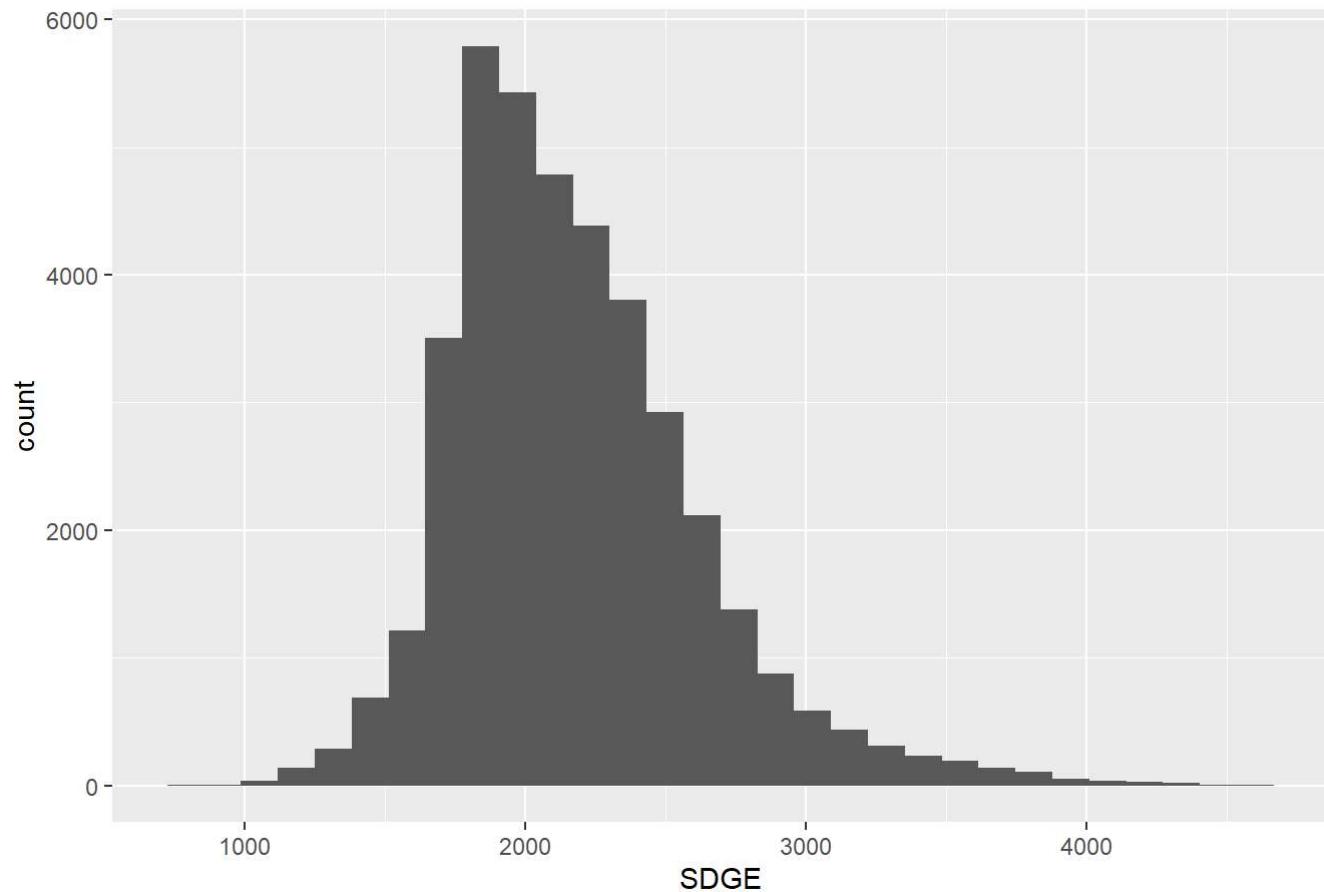


Distributions - also will make this look better tomorrow.

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

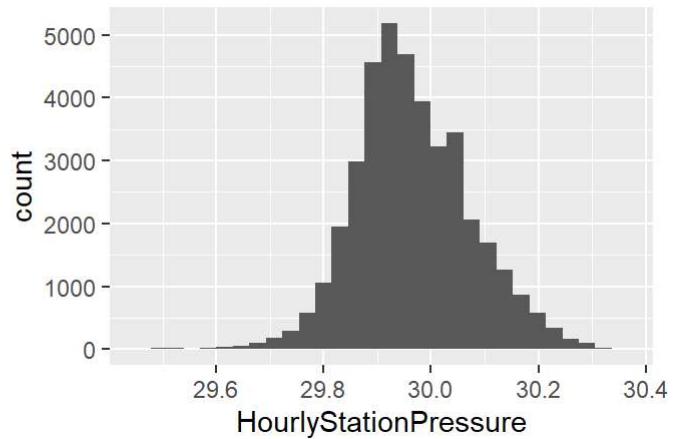
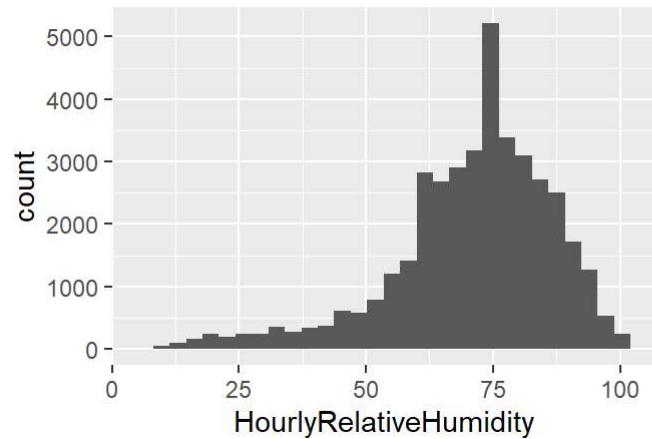
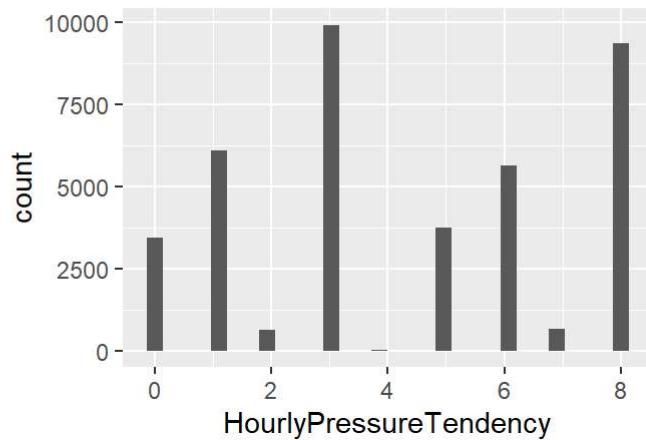
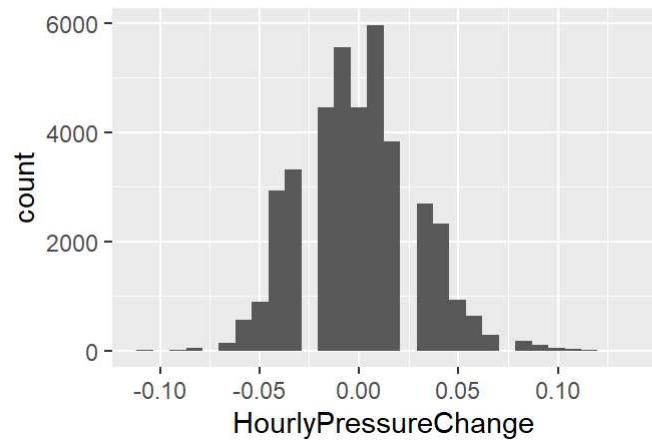
Hist_target <- Hist_plotter(1) + ggtitle("Figure X.x: Distribution for SDGE")
Hist_PHP <- lapply(PHP_cols, Hist_plotter)
Hist_TWV <- lapply(TWV_cols, Hist_plotter)
Hist_paper <- lapply(paper_cols, Hist_plotter)

plot(Hist_target)
```

Figure X.x: Distribution for SDGE

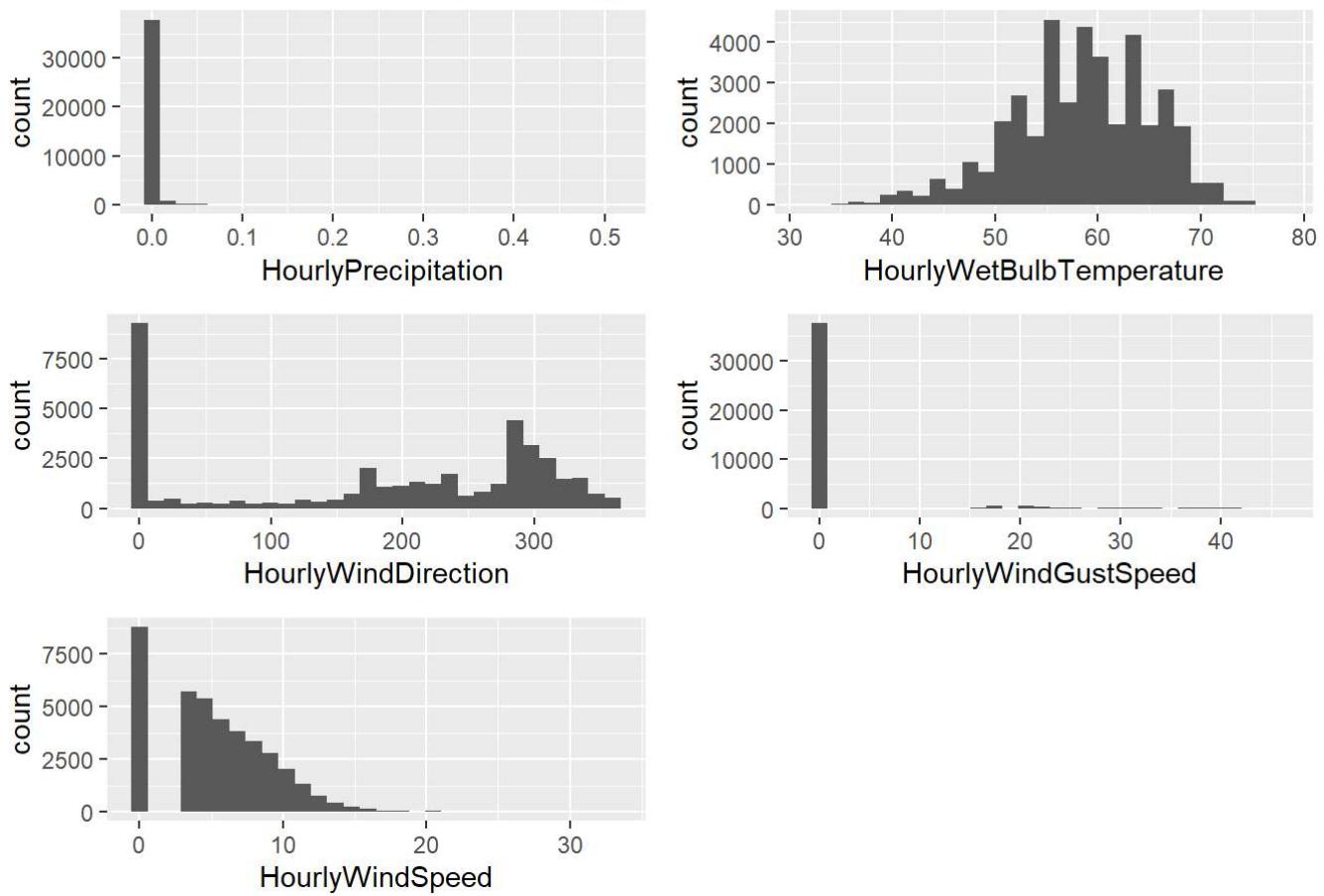
```
grid.arrange(grobs = Hist_PHP, top = "Figure X.x: Distribution for Pressure, Humidity, and Precipitation Variables")
```

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



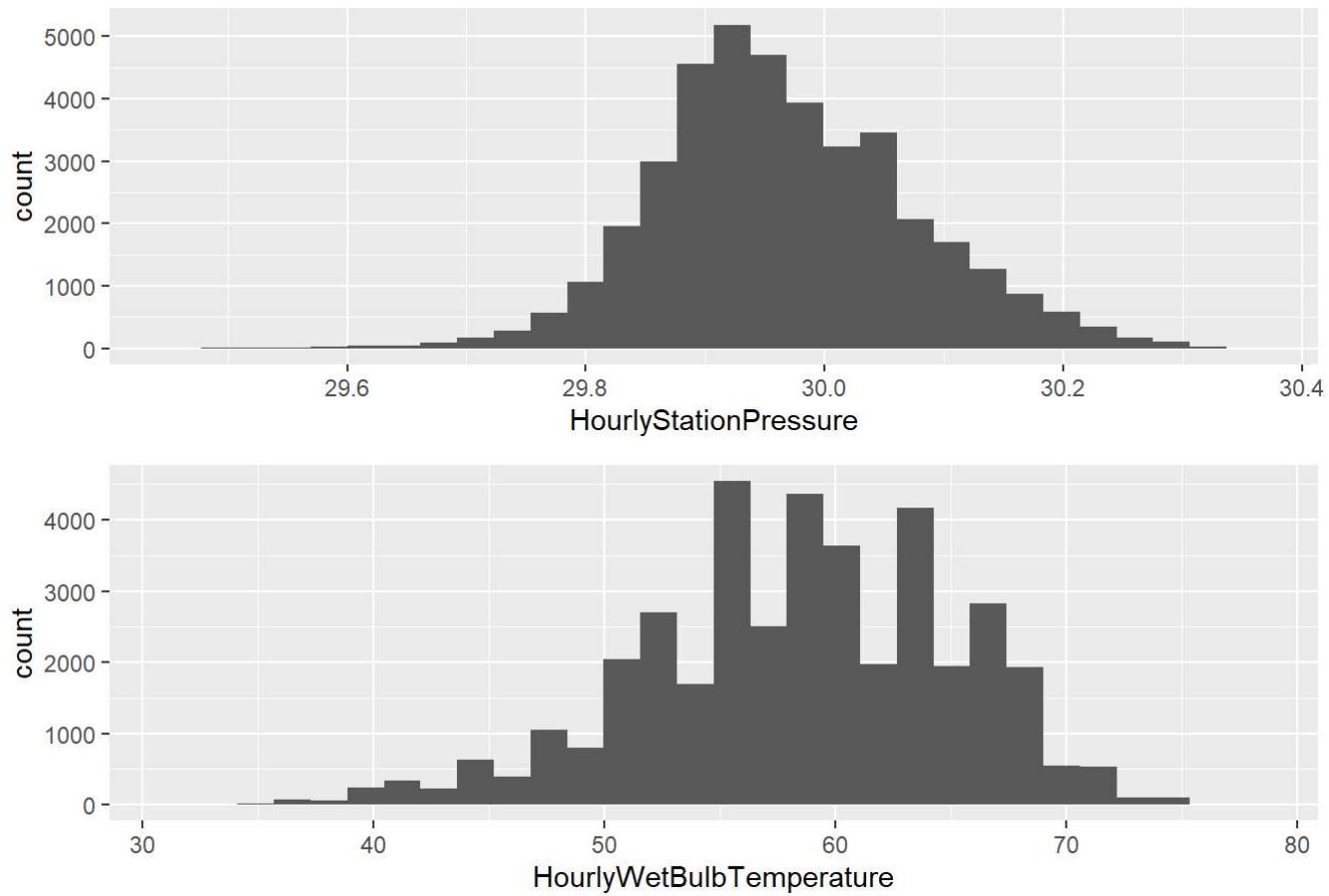
```
grid.arrange(grobs = Hist_TWV, top = "Figure X.x: Distribution for Temperature, Wind, and Visibility Variables")
```

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 forever to run.
```

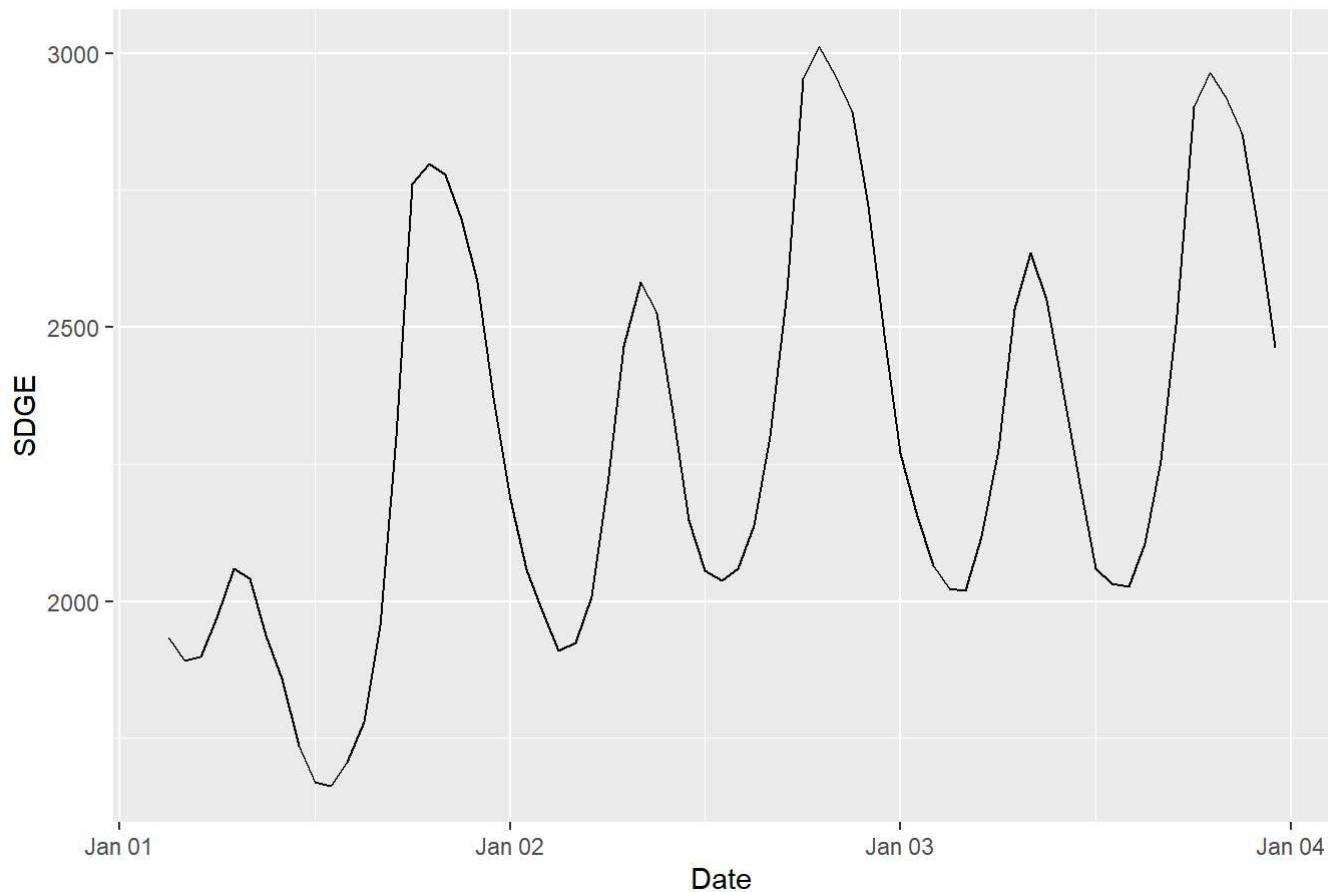
```
WT_tukey <- TukeyHSD(WT_aov)
```

Target EDA

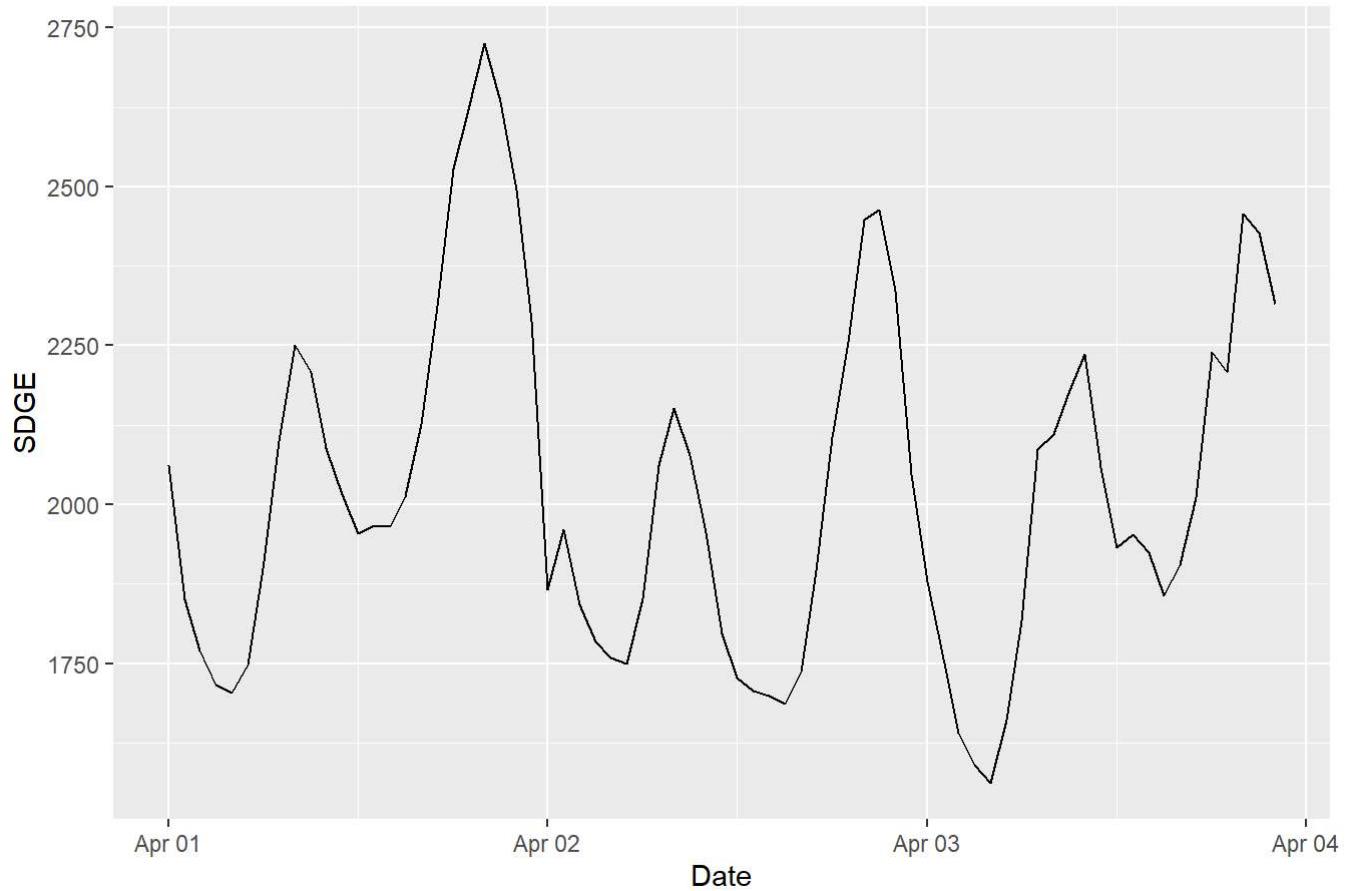
We've already seen yearly seasonality in SDGE from time series above, lets evaluate a few days to see if there is also a daily seasonality. We'll evaluate the first couple days in Jan, Apr, Jul, and Oct. We definitely see a daily-based seasonality. So, we'll have to account for seasonality across the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

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

Figure X.x: Seasonality in January

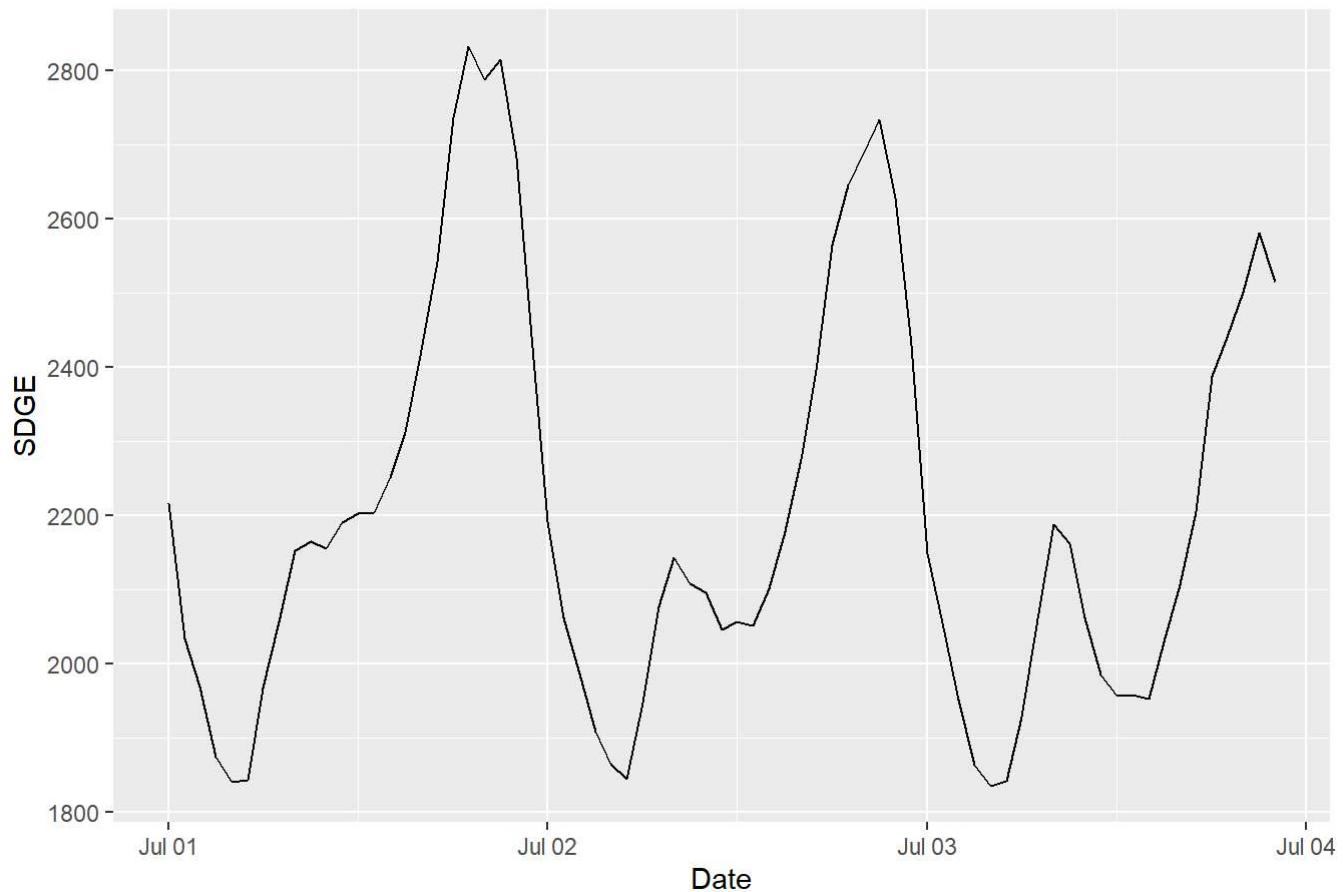


```
SD %>% filter(Date < '2019-04-03 18:00:00') %>% filter(Date > '2019-03-31 18:00:00') %>% ggplot  
(aes(x = Date, y = SDGE)) + geom_line() + ggtitle("Figure 1.14: Seasonality in April")
```

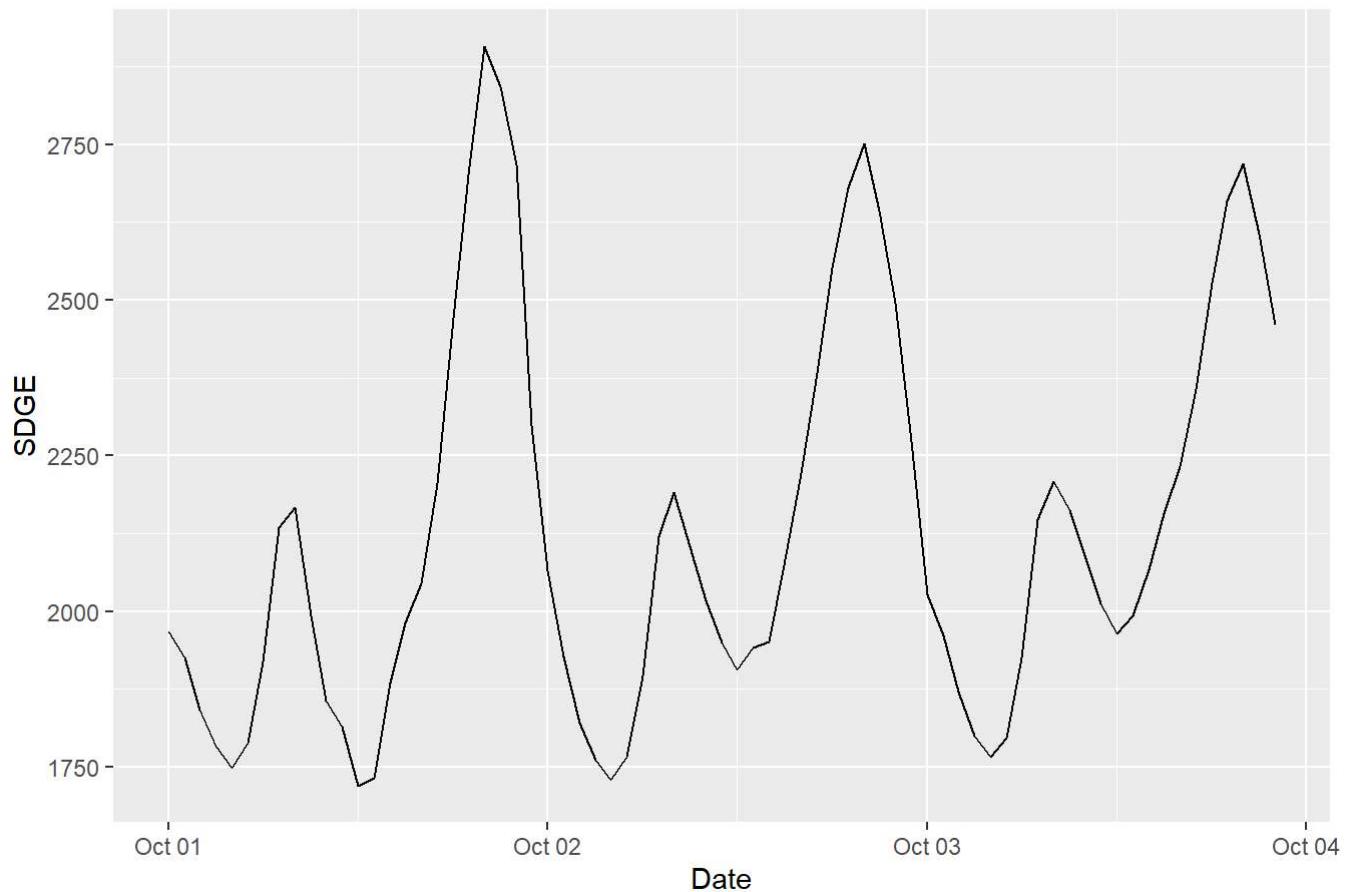
Figure 1.14: Seasonality in April

```
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 data.")

```

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

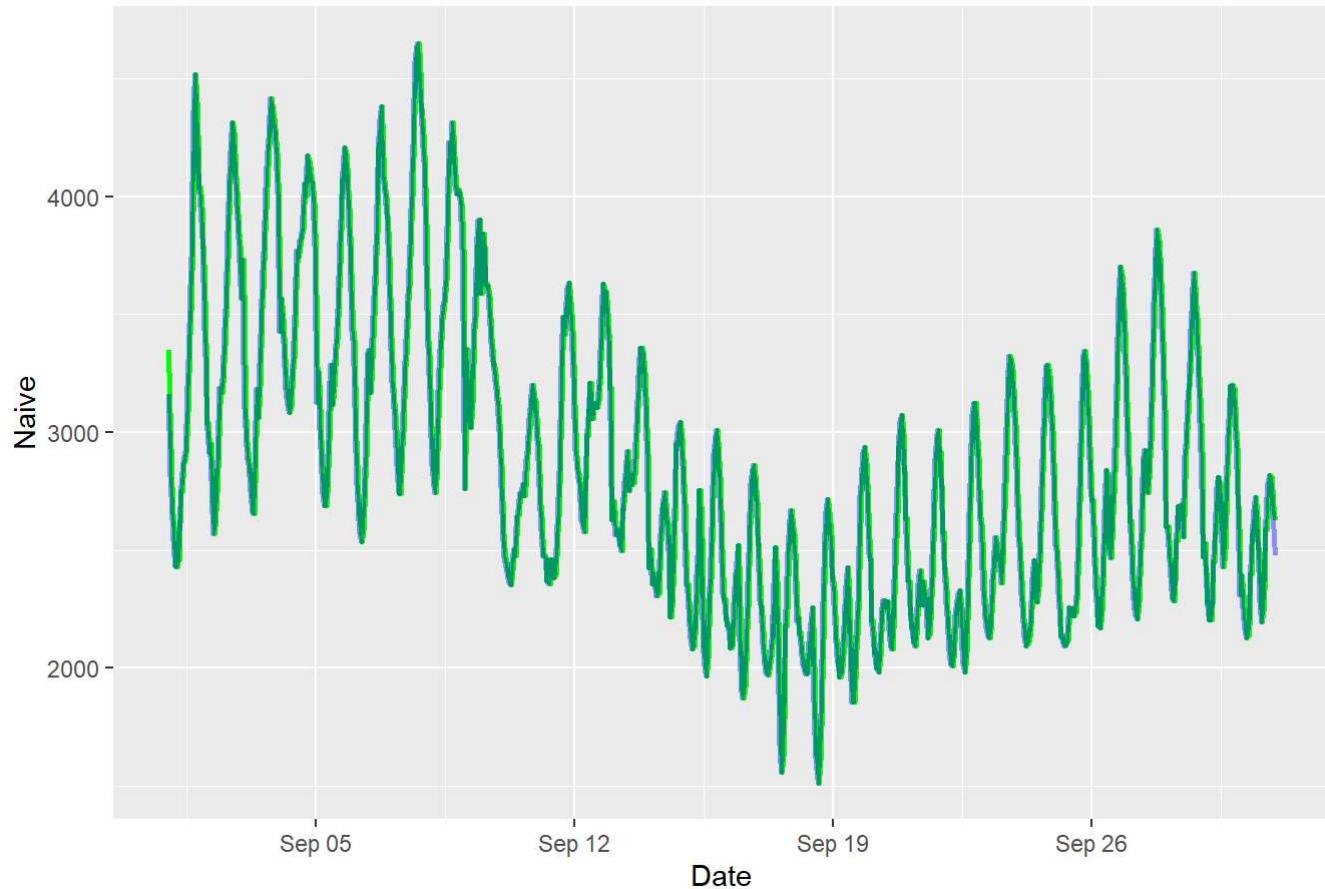
Plot on validation data

```

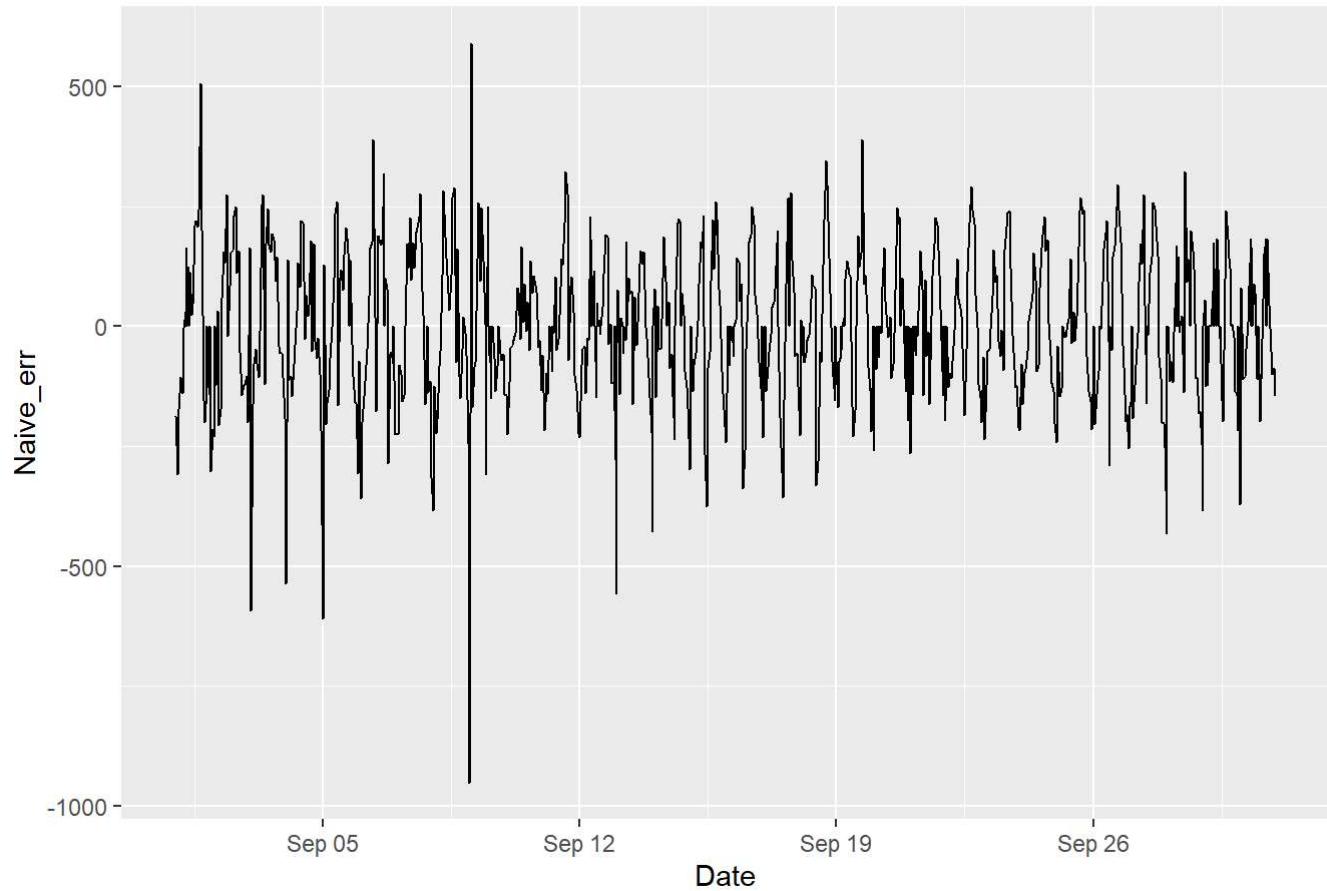
ggplot(SD_val, aes(x = Date)) +
  geom_line(aes(y = Naive), color = 'green', linewidth = 1) +
  geom_line(aes(y = SDGE), color = 'blue', alpha = 0.4, linewidth = 1) +
  ggtitle("Figure 2.1: Naive Forecasts on Validation Data")

```

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

```

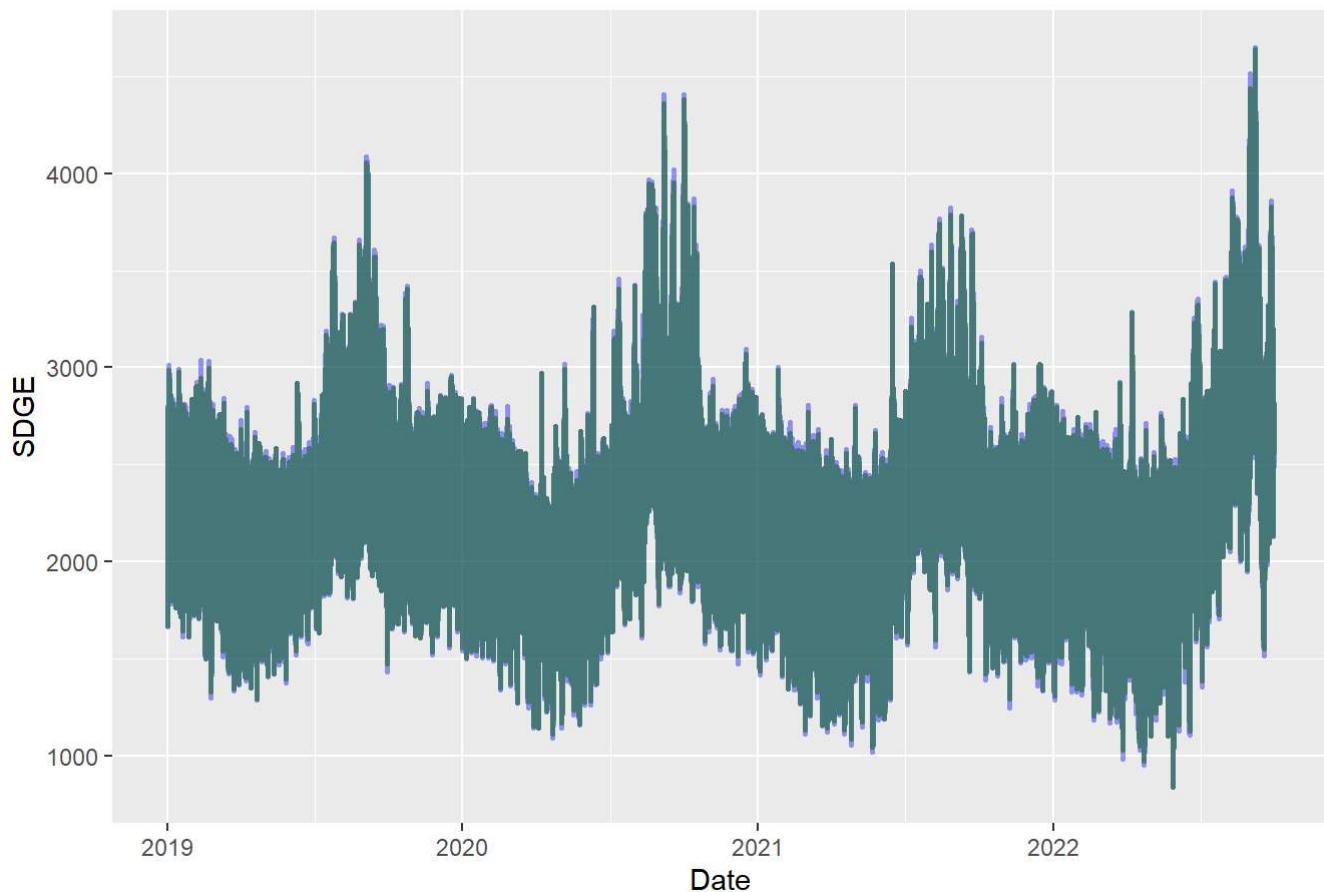
DailyMA_2 <- rollmean(SD_madf$SDGE, k = 2)
DailyMA_2 <- c(rep(NA, 1), DailyMA_2)

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")

```

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

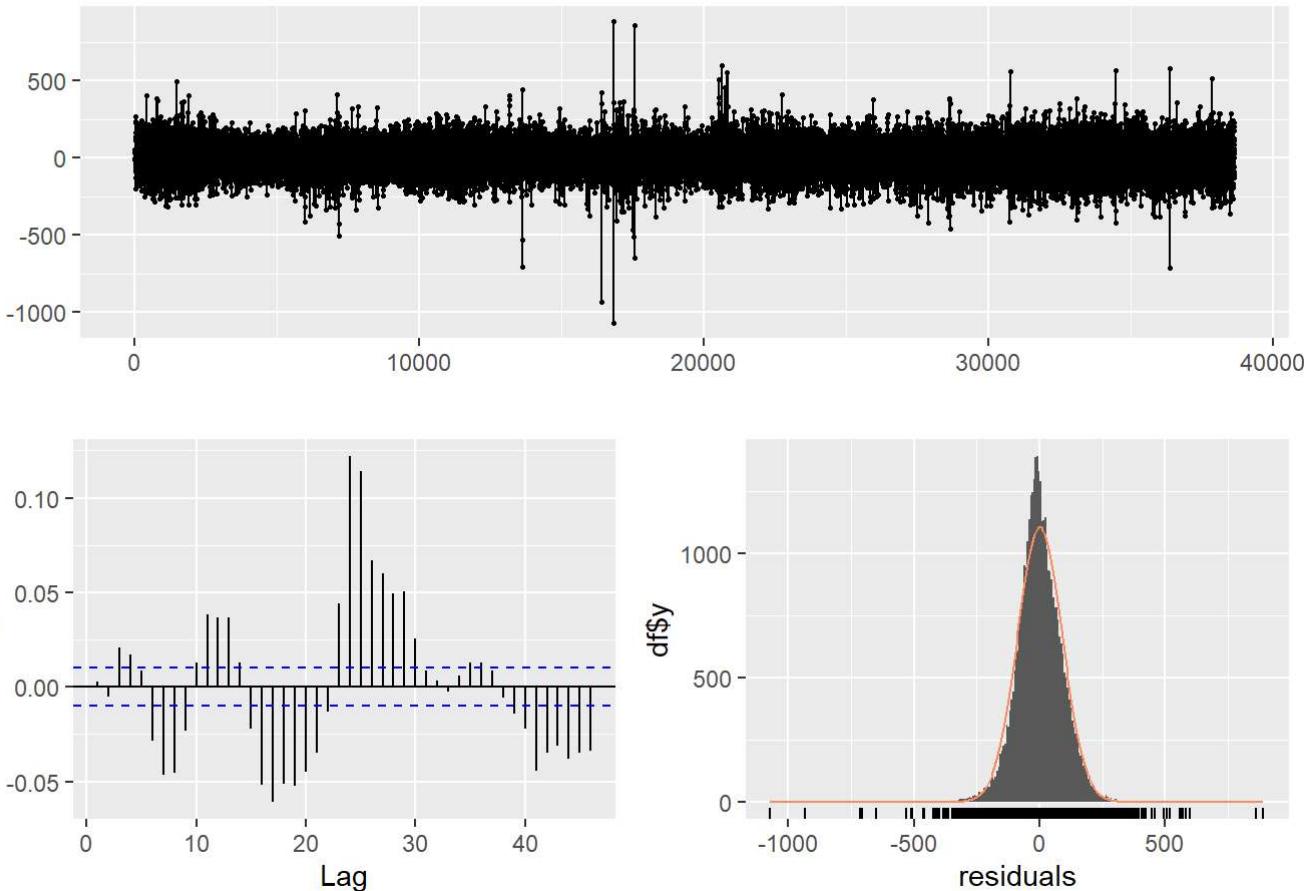
```
set.seed(123)
auto_arima_model <- auto.arima(y_tr, xreg = X_tr, seasonal = TRUE, stepwise = FALSE)
summary(auto_arima_model)
```

```
## 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)
```

Residuals from Regression with ARIMA(2,1,3) errors



```
##  
## 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

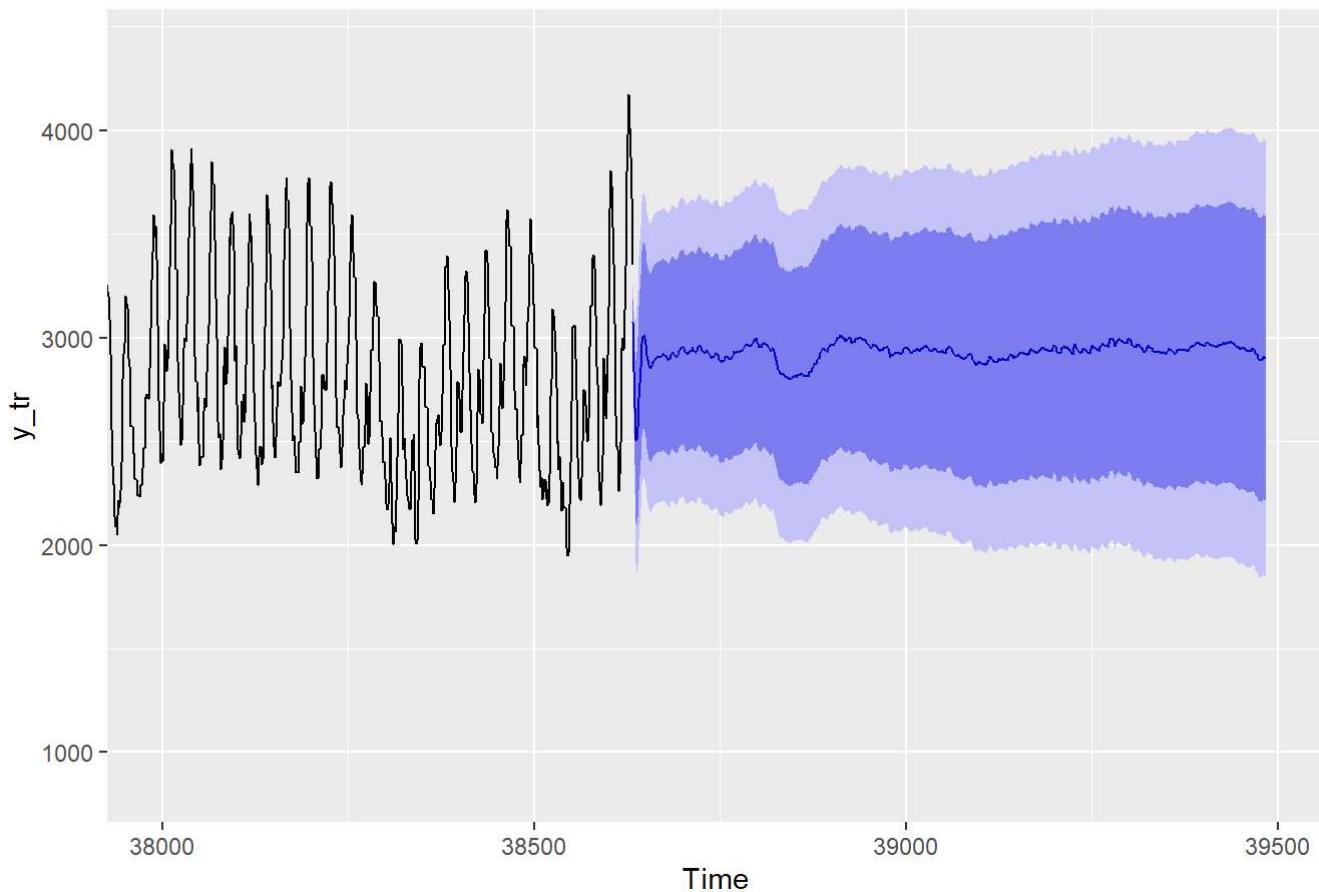
```
val_forecast <- forecast(auto_arima_model, xreg = X_val)

# 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

```

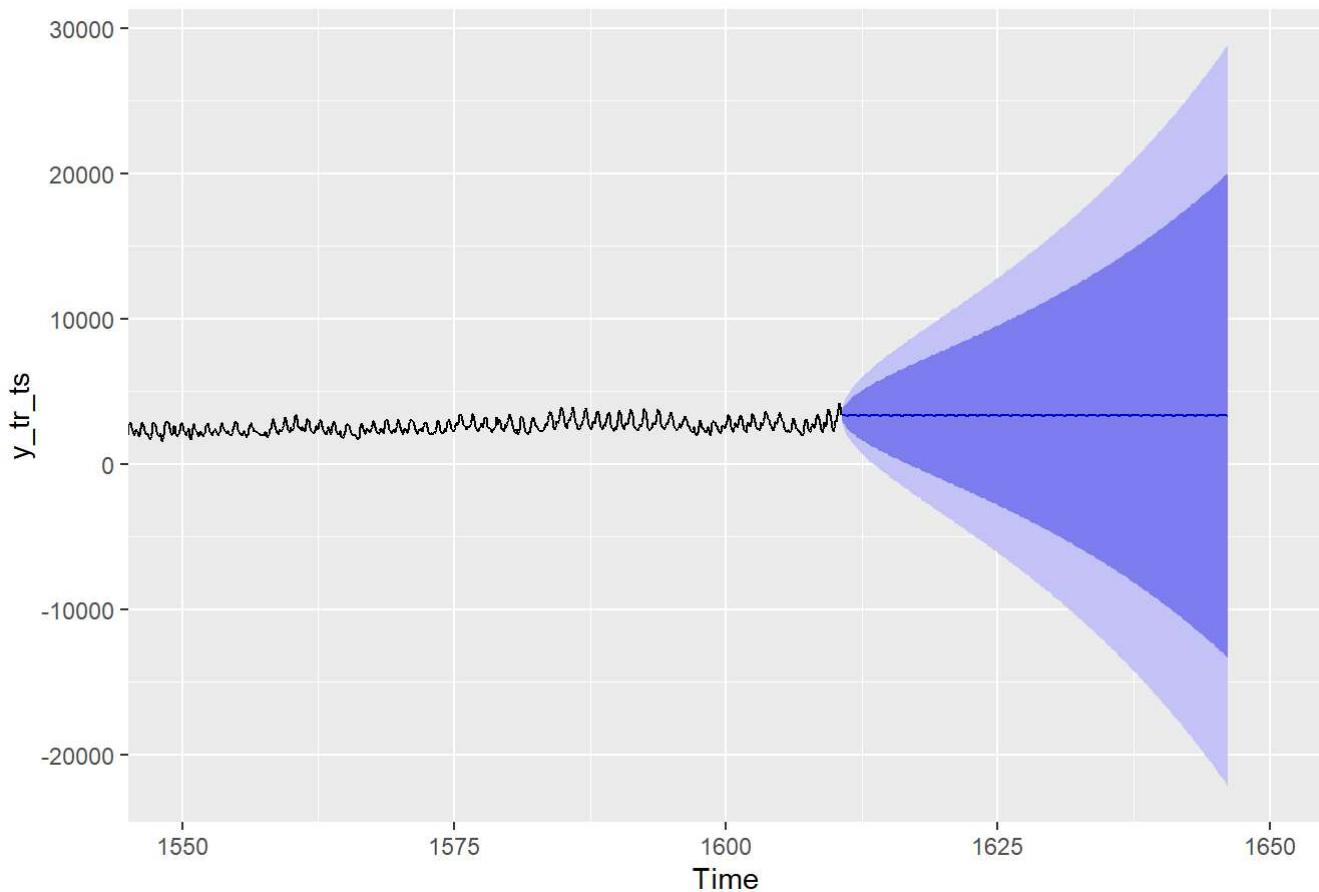
ets_forecast <- forecast(SD_ets, h=length(y_test))

#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)")

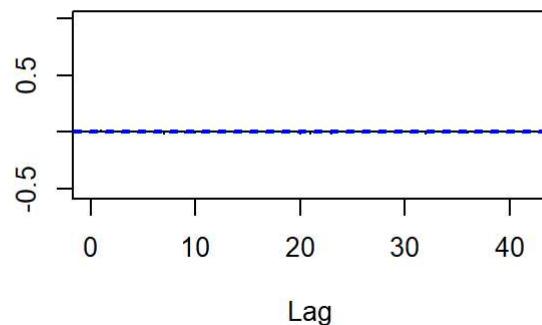
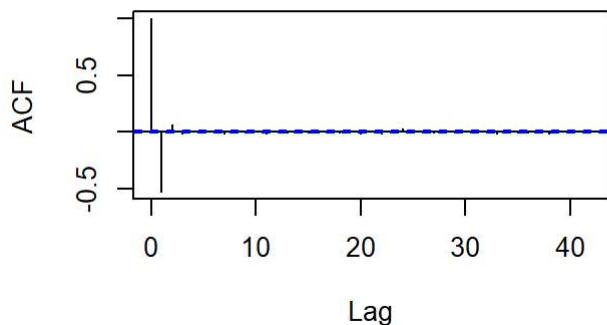
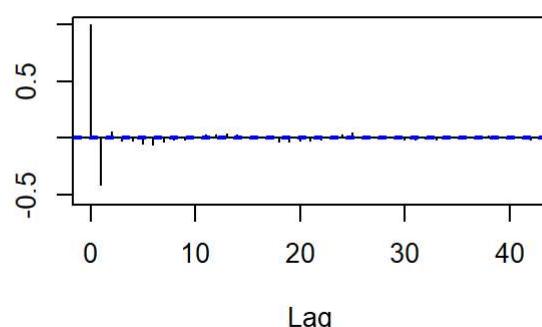
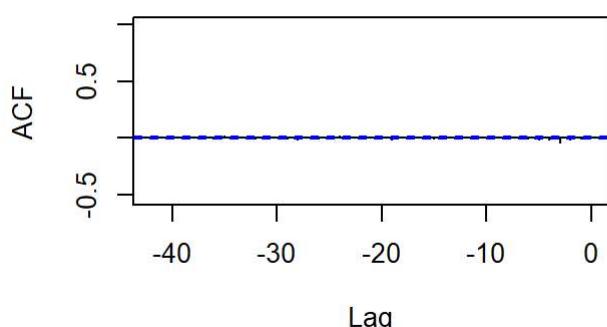
```

Figure 2.6: Forecasts from ETS(M,N,A)

Manual ARIMA

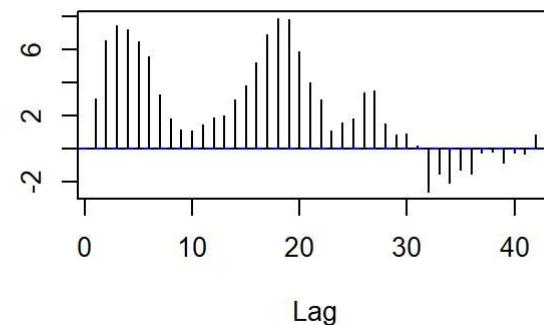
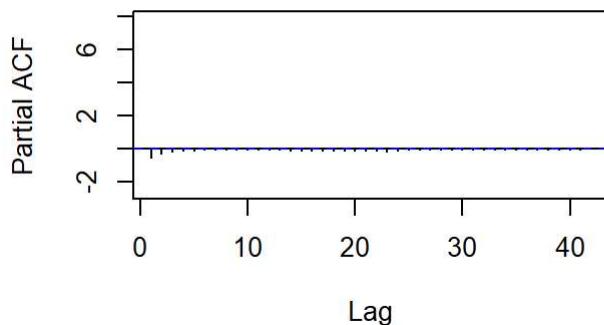
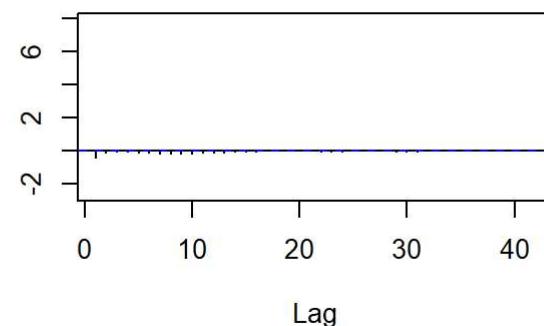
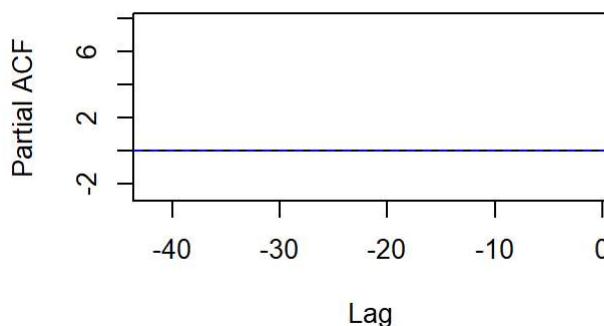
Twice differenced ACF plot

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

!8: Twice-Differenced ACF Plots for Numeric Predictors**!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")
```

.9: Twice-Differenced PACF Plots for Numeri**.9: Twice-Differenced PACF Plots for Numeri****Model**

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

```

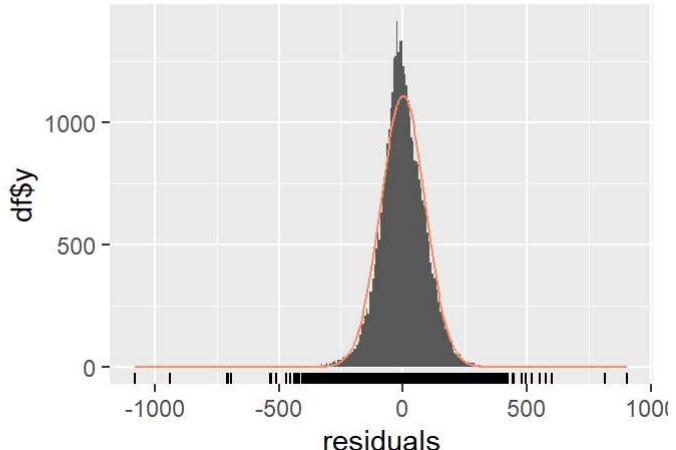
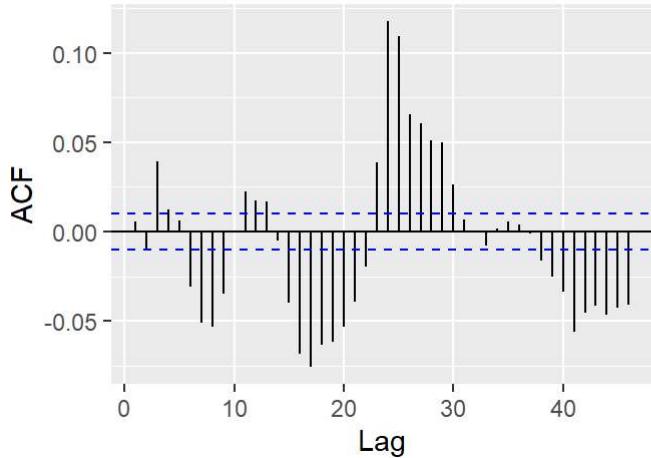
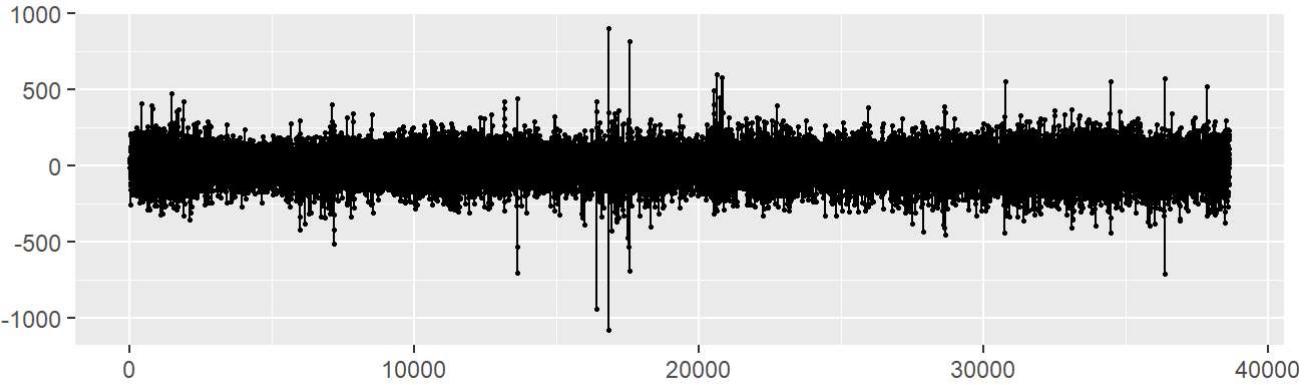
## Series: y_tr
## Regression with ARIMA(3,2,2) errors
##
## Coefficients:
##             ar1      ar2      ar3      ma1      ma2  HourlyWetBulbTemperature
##             1.3593  -0.2603 -0.2348 -1.9505  0.9506                      3.8320
## s.e.    0.0053   0.0087  0.0050   0.0023  0.0023                      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)
```

Residuals from Regression with ARIMA(3,2,2) errors

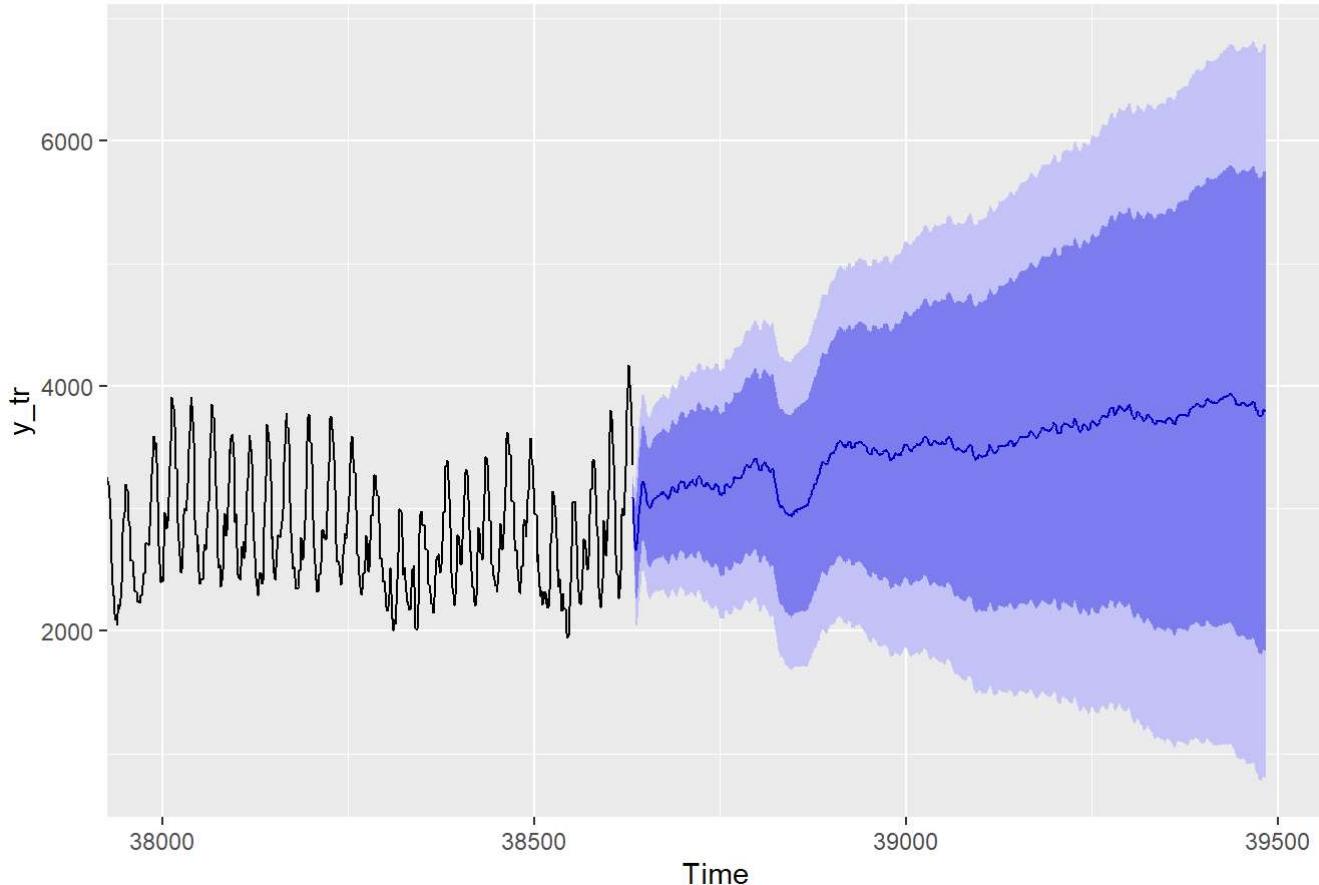


```
##  
## Ljung-Box test  
##  
## data: Residuals from Regression with ARIMA(3,2,2) errors  
## Q* = 364.27, df = 5, p-value < 2.2e-16  
##  
## Model df: 5. Total lags used: 10
```

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

Figure 2.11: Forecast from Regression with ARIMA(3,2,2) Errors



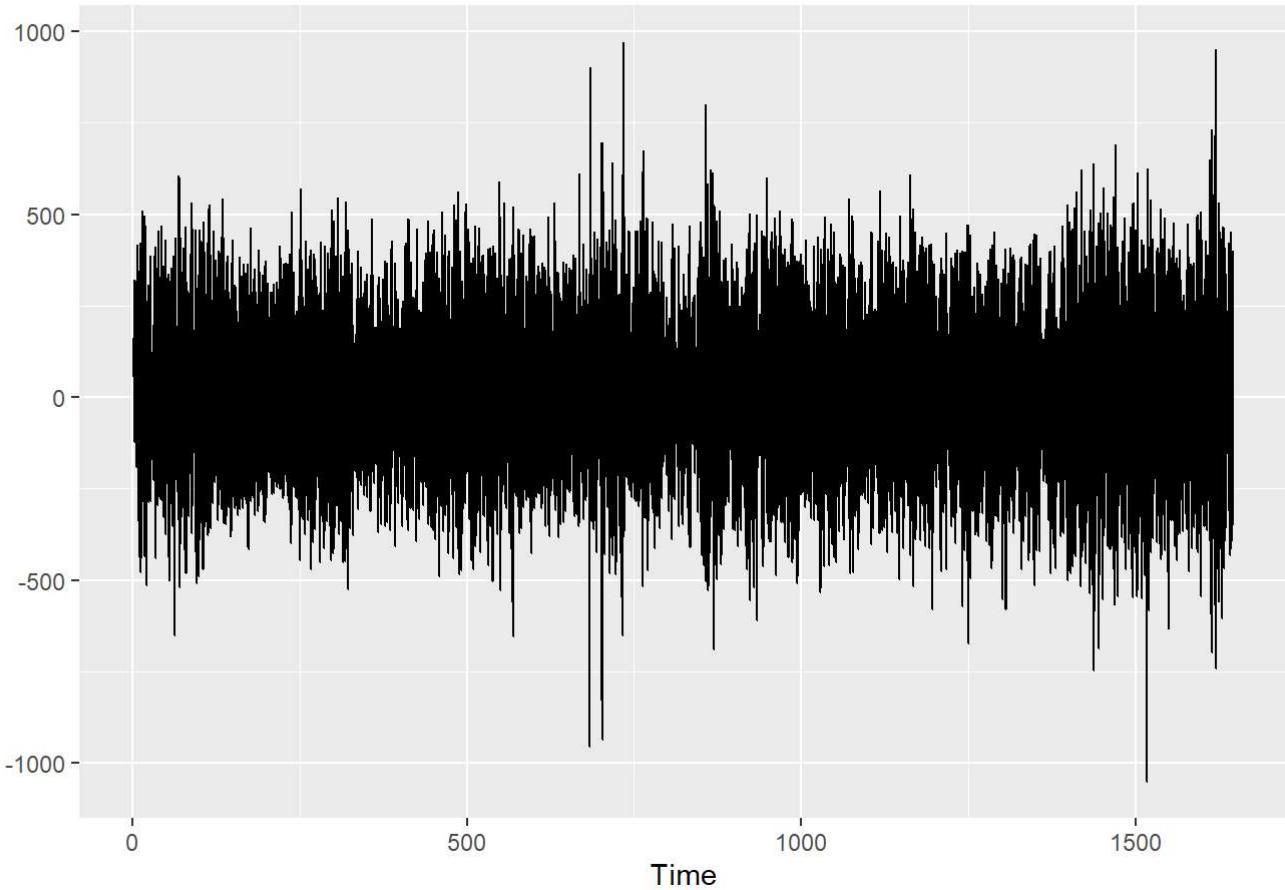
Manual ARIMA with Seasonality

Create daily ts for easier seasonal eval

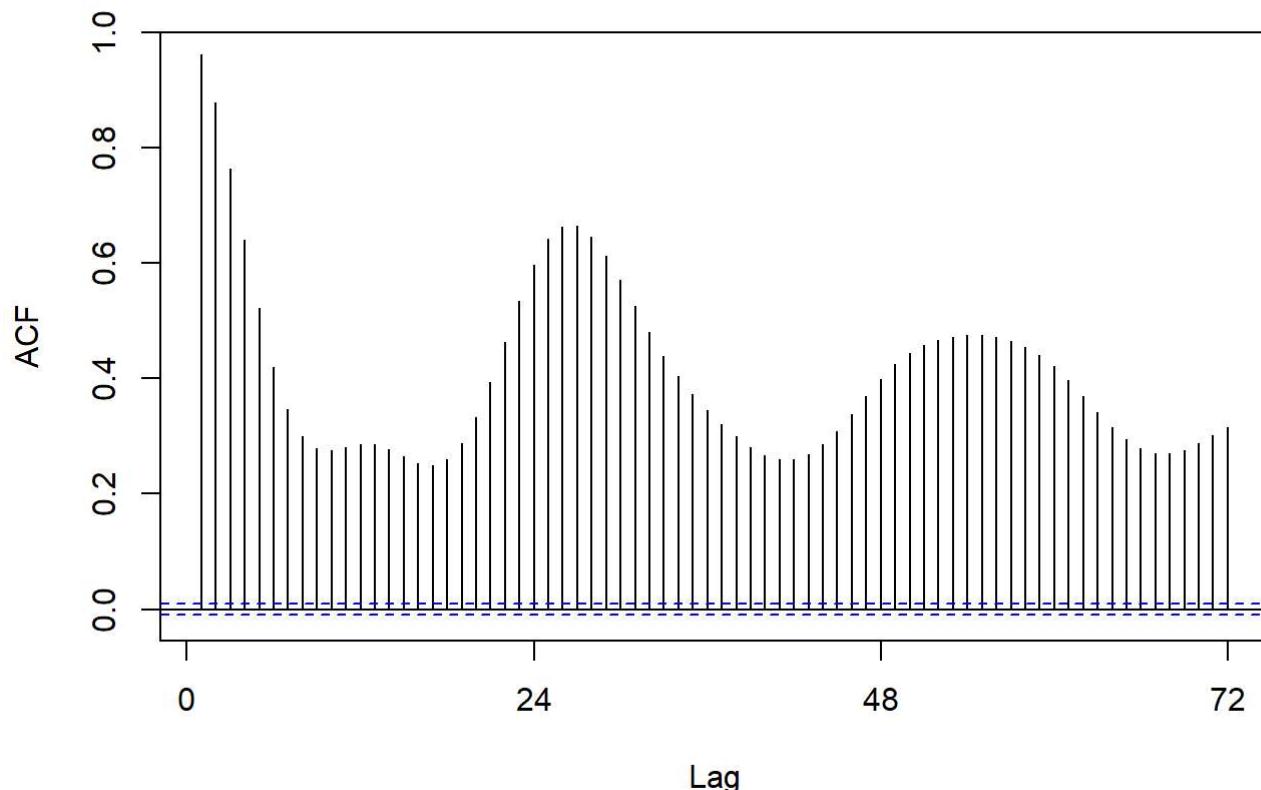
```
SD_ts <- ts(SD$SDGE, frequency = 24)
```

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

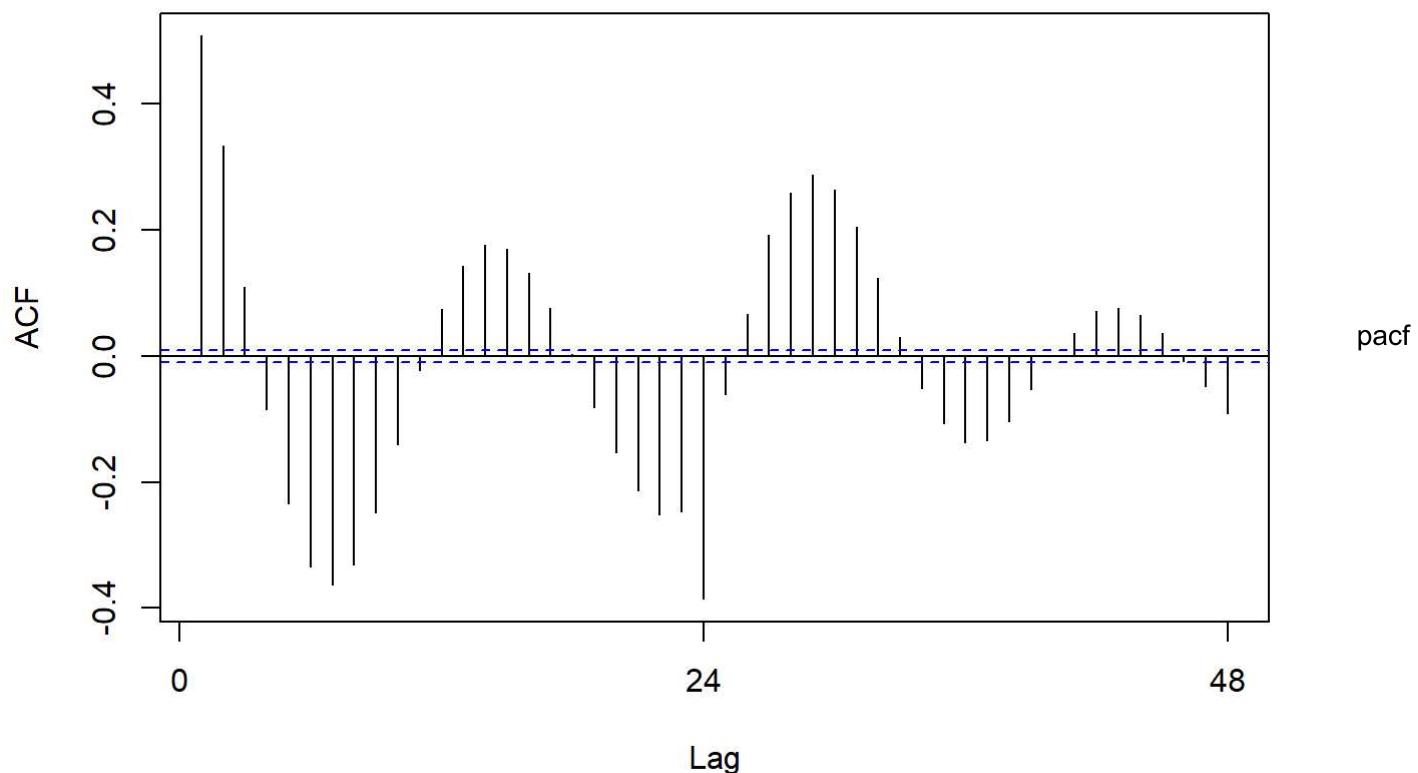
Figure 2.12: Twice Differenced Time Series for SDGE



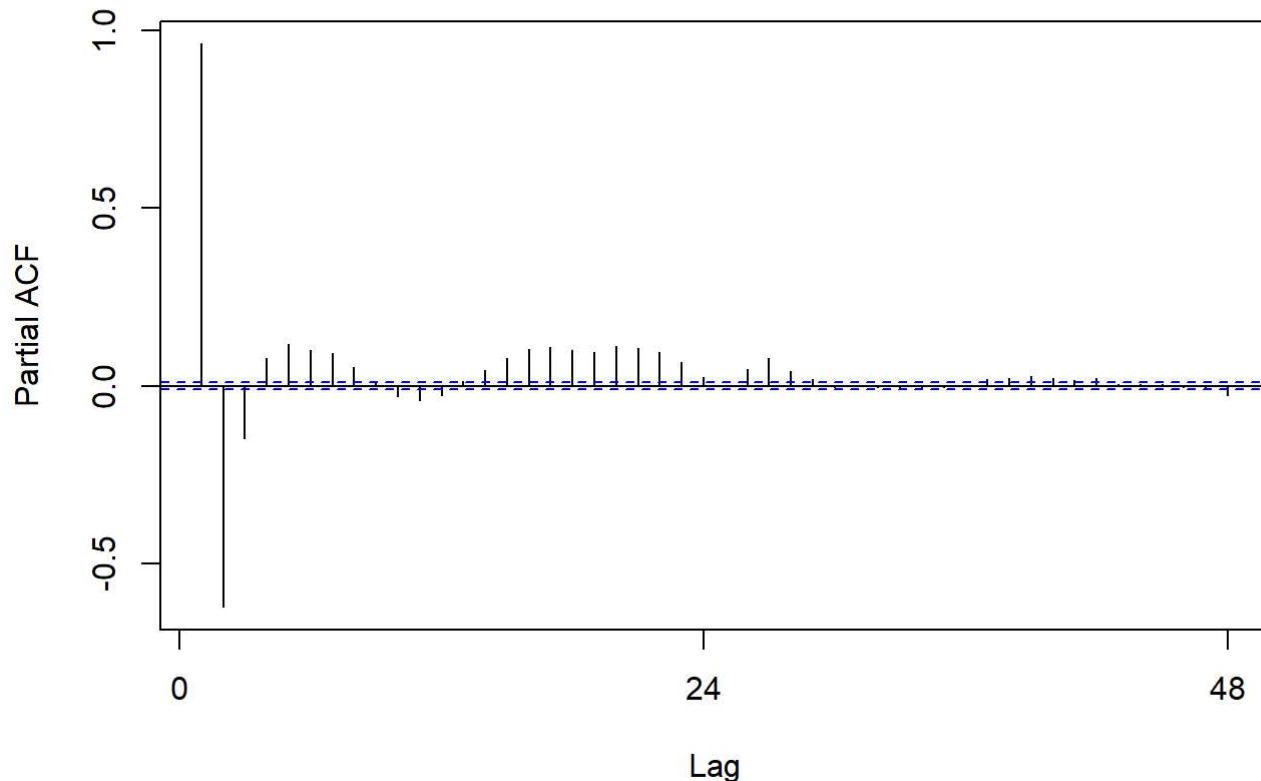
```
Acf(SD_ts, lag.max = 72)
```

Series SD_ts

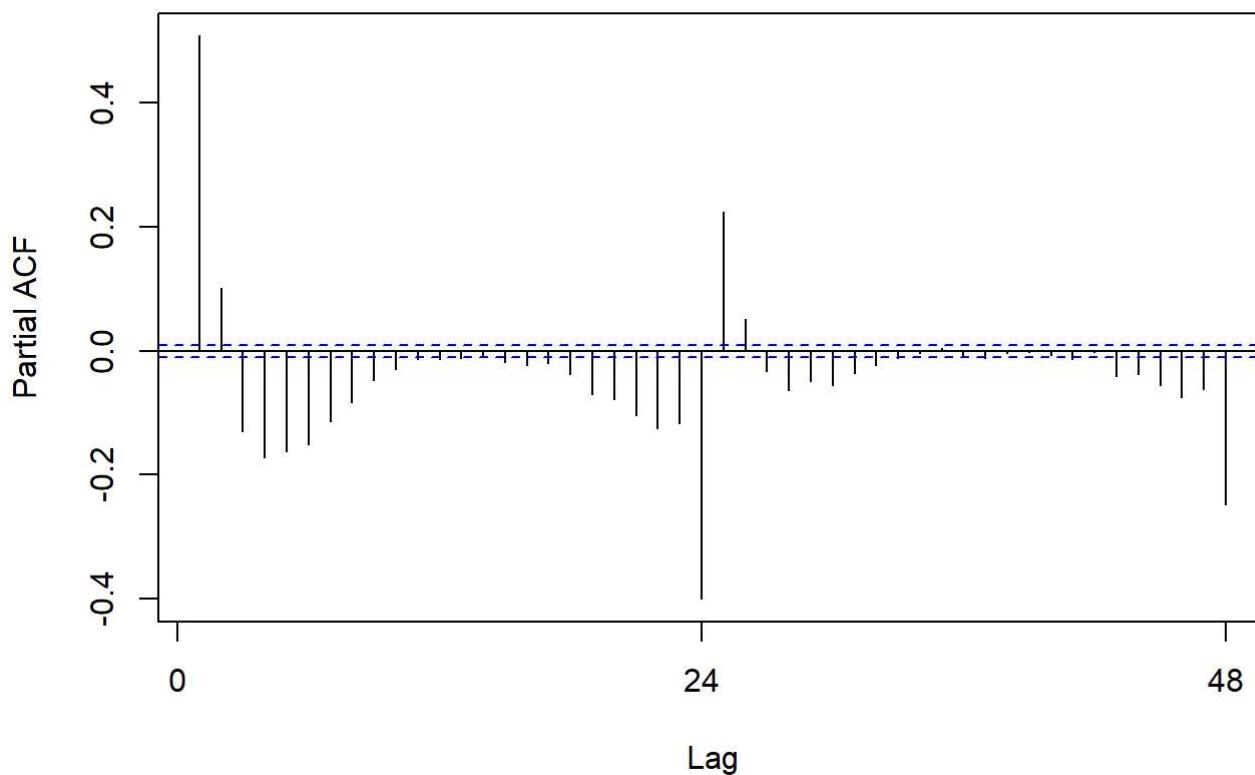
```
SD_ts %>% diff(lag = 24) %>% diff() %>% Acf(main = "Figure 2.13: Twice Differenced ACF Plot for SDGE")
```

Figure 2.13: Twice Differenced ACF Plot for SDGE

```
Pacf(SD_ts)
```

Series 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

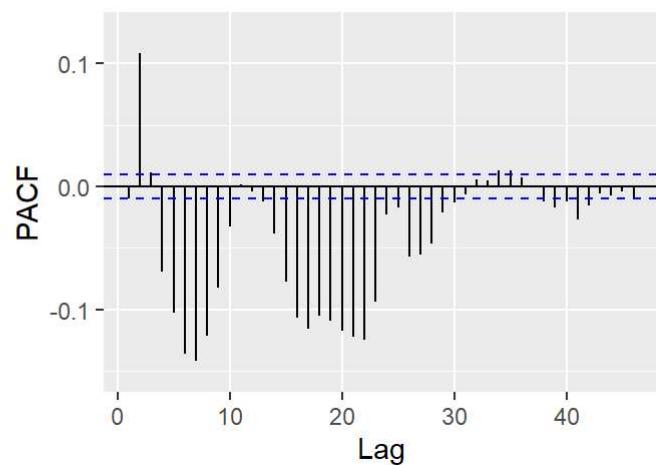
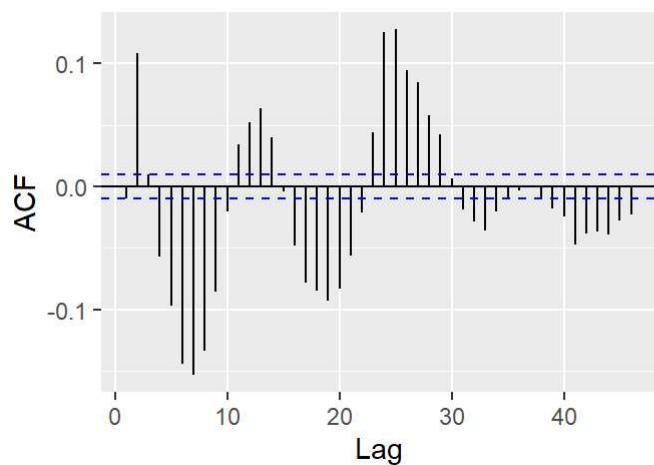
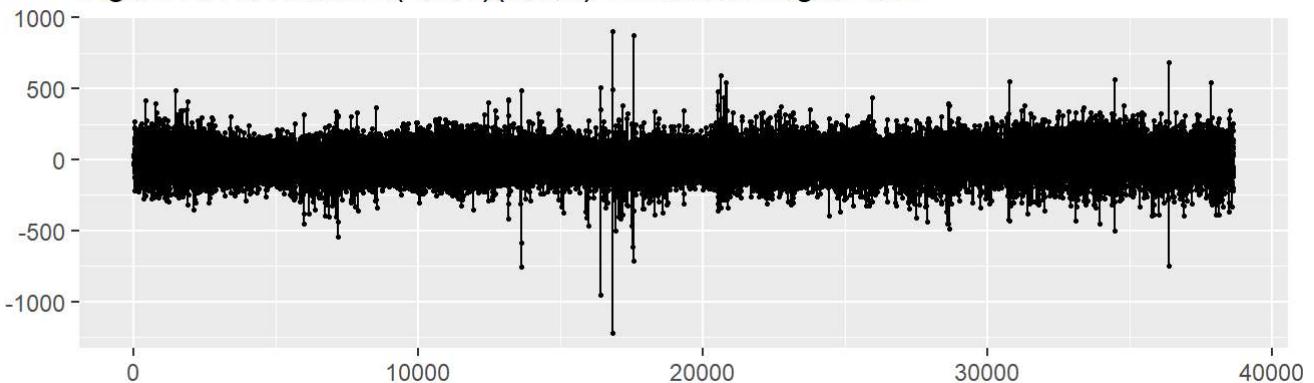
Fit model - Acf shows strong MA(1) for non-seasonal and seasonal on twice differenced chart, same story with Pacf for our AR(p) component. We took a non-seasonal and seasonal difference so we'll train a ARIMA(1,1,1) (1,1,1) model

```
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
```

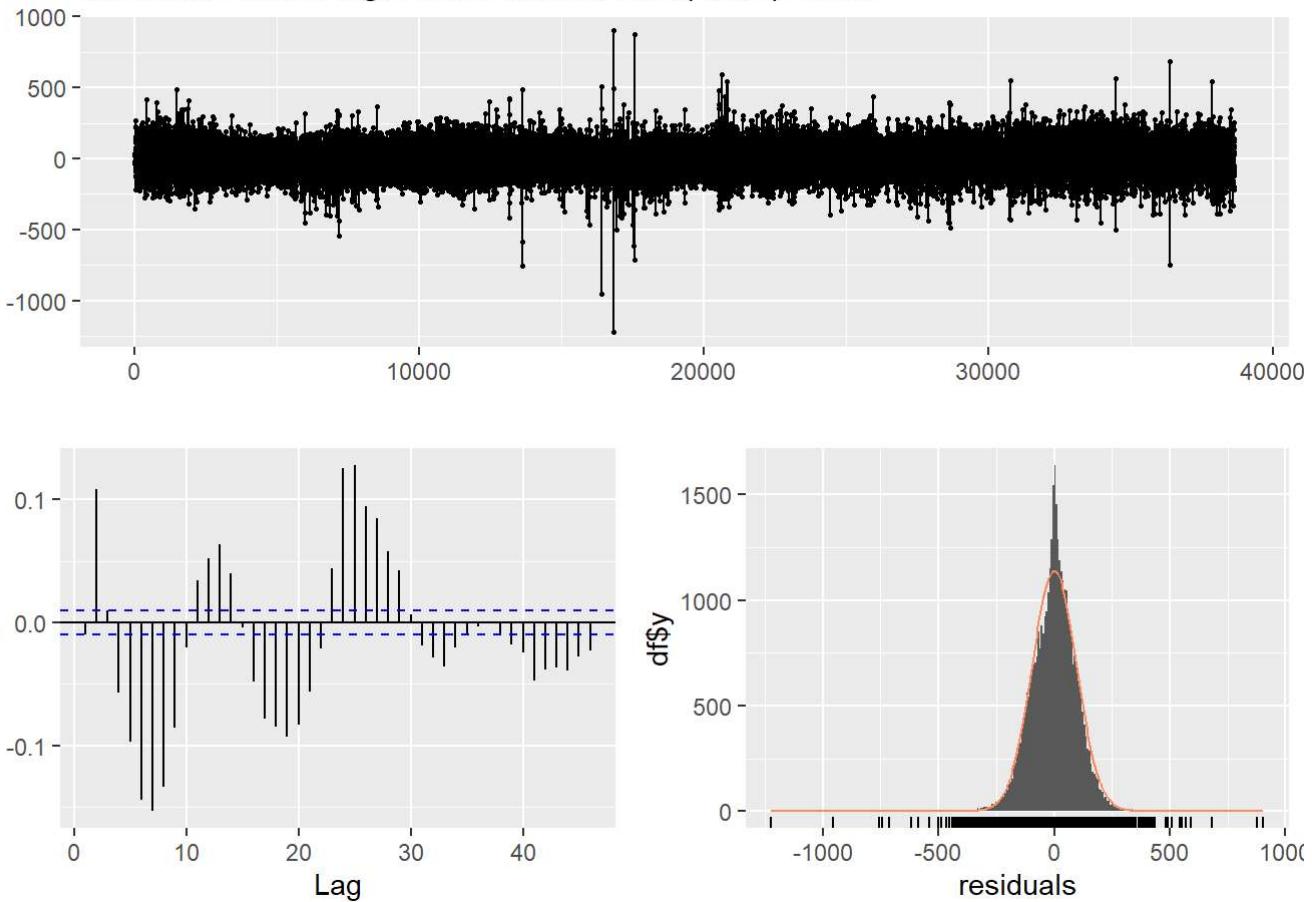
```
seasonal_arima %>% residuals() %>% ggtsdisplay(main = "Figure 2.15: ARIMA (1,1,1)(1,1,1) Residua  
l Diagnostics")
```

Figure 2.15: ARIMA (1,1,1)(1,1,1) Residual Diagnostics



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

Residuals from Regression with ARIMA(1,1,1) errors



```
##  
## 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)
```

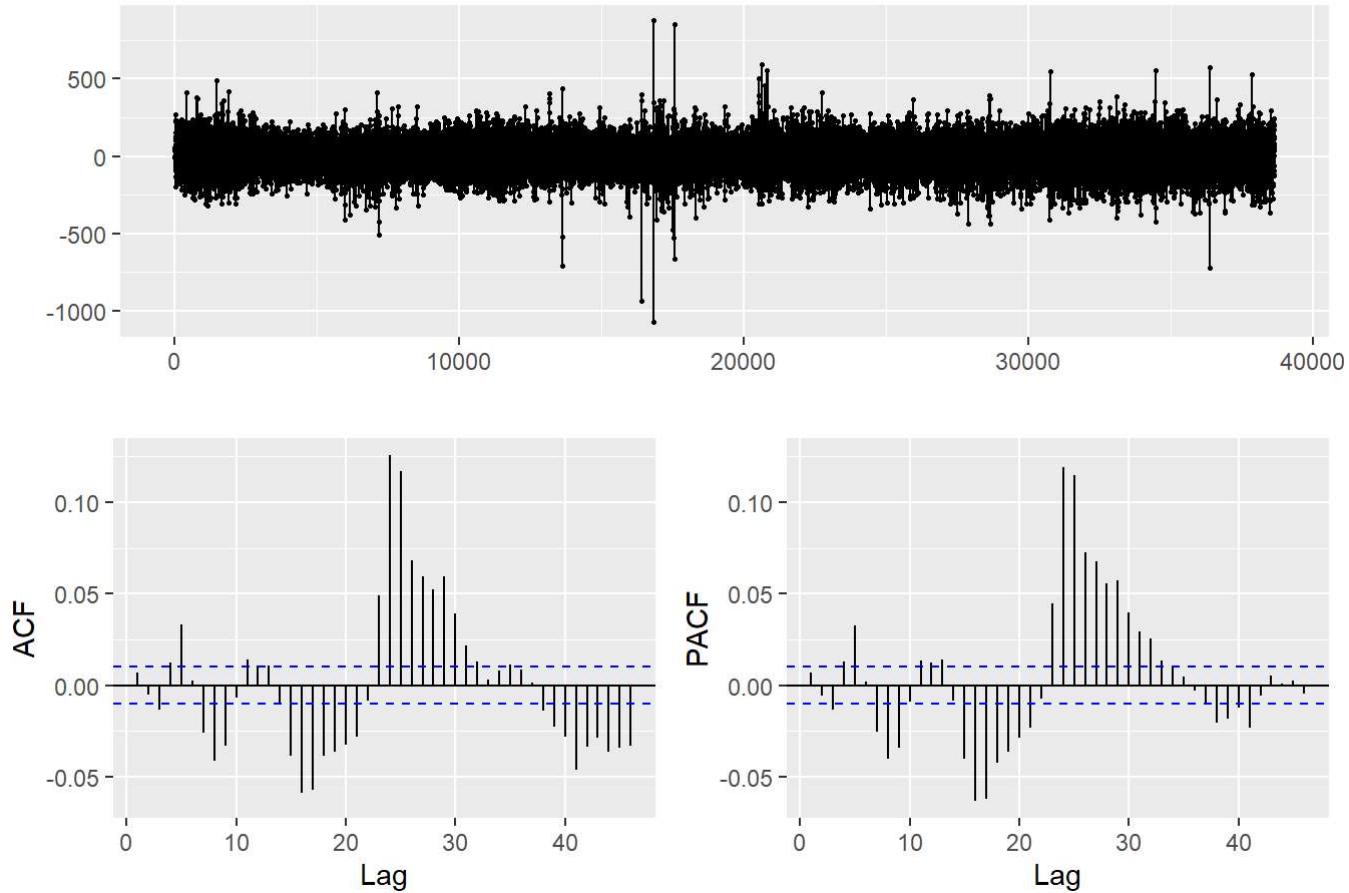
```

## 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

```

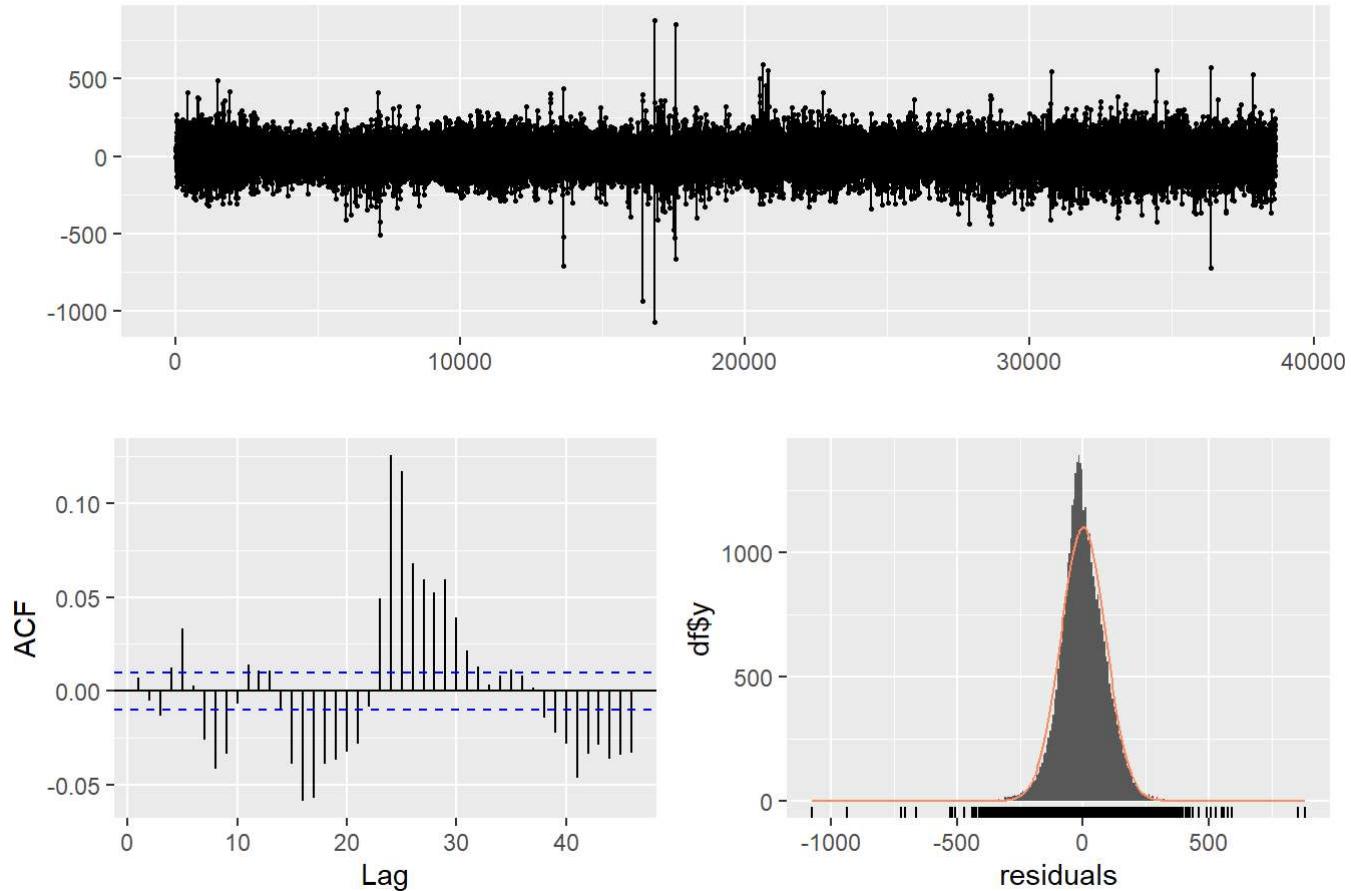
```
seasonal_arima_tuned %>% residuals() %>% ggtsdisplay(main = "Figure 2.17: ARIMA (3,1,3)(2,1,2) R  
esidual Diagnostics")
```

Figure 2.17: ARIMA (3,1,3)(2,1,2) Residual Diagnostics



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

Residuals from Regression with ARIMA(3,1,3) errors

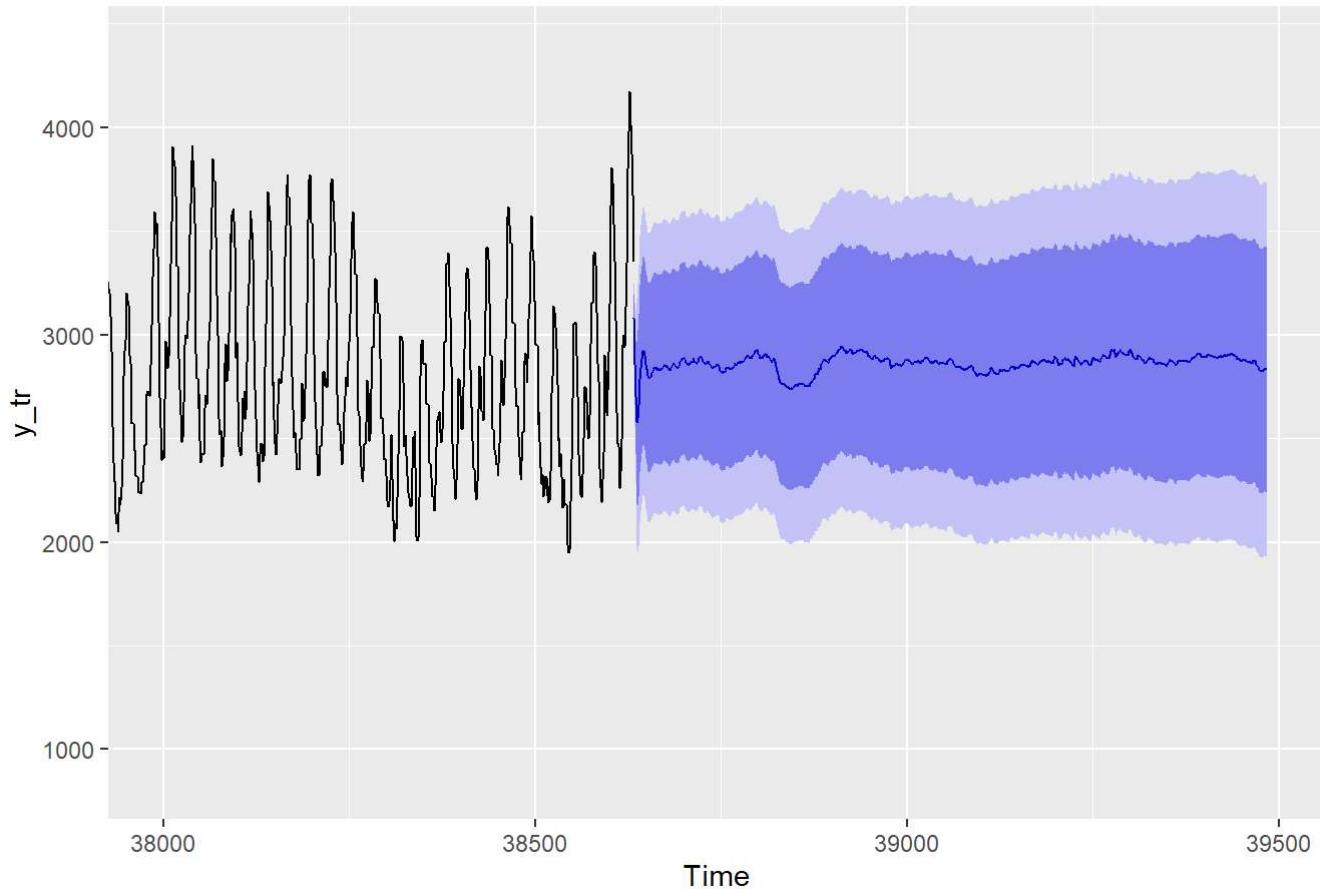


```
##  
## Ljung-Box test  
##  
## data: Residuals from Regression with ARIMA(3,1,3) errors  
## Q* = 195.89, df = 4, p-value < 2.2e-16  
##  
## Model df: 6. Total lags used: 10
```

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)
```

```
##  
## Forecast method: NNAR(3,2,3)[24]  
##  
## Model Information:  
##  
## Average of 20 networks, each of which is  
## a 7-3-1 network with 28 weights  
## options were - linear output units  
##  
## Error measures:  
##  
## Training set 0.05947569 84.47837 63.86467 -0.1675212 2.996544 0.2140053  
##  
## ACF1  
## Training set -0.001513646  
##  
## Forecasts:  
##  
## Point Forecast  
## 1610.67 3126.726  
## 1610.71 2918.711  
## 1610.75 2755.391  
## 1610.79 2625.649  
## 1610.83 2528.579  
## 1610.88 2451.930  
## 1610.92 2403.283  
## 1610.96 2380.128  
## 1611.00 2387.518  
## 1611.04 2427.880  
## 1611.08 2494.204  
## 1611.12 2580.189  
## 1611.17 2681.131  
## 1611.21 2804.462  
## 1611.25 2948.966  
## 1611.29 3112.657  
## 1611.33 3299.365  
## 1611.38 3484.470  
## 1611.42 3664.057  
## 1611.46 3810.800  
## 1611.50 3912.391  
## 1611.54 3949.720  
## 1611.58 3937.921  
## 1611.62 3865.512  
## 1611.67 3758.767  
## 1611.71 3616.648  
## 1611.75 3468.379  
## 1611.79 3316.618  
## 1611.83 3172.513  
## 1611.88 3039.096  
## 1611.92 2921.049  
## 1611.96 2815.141  
## 1612.00 2726.007  
## 1612.04 2657.651  
## 1612.08 2611.385
```

## 1612.12	2588.353
## 1612.17	2587.823
## 1612.21	2610.782
## 1612.25	2656.684
## 1612.29	2736.988
## 1612.33	2844.555
## 1612.38	2985.592
## 1612.42	3145.438
## 1612.46	3320.764
## 1612.50	3515.286
## 1612.54	3691.120
## 1612.58	3830.096
## 1612.62	3922.392
## 1612.67	3953.476
## 1612.71	3932.394
## 1612.75	3866.746
## 1612.79	3767.078
## 1612.83	3644.311
## 1612.88	3511.345
## 1612.92	3378.145
## 1612.96	3252.470
## 1613.00	3138.379
## 1613.04	3043.335
## 1613.08	2959.115
## 1613.12	2888.910
## 1613.17	2830.484
## 1613.21	2790.840
## 1613.25	2772.407
## 1613.29	2778.824
## 1613.33	2807.603
## 1613.38	2860.956
## 1613.42	2945.237
## 1613.46	3077.502
## 1613.50	3234.627
## 1613.54	3410.866
## 1613.58	3592.436
## 1613.62	3755.033
## 1613.67	3876.305
## 1613.71	3934.026
## 1613.75	3937.822
## 1613.79	3902.870
## 1613.83	3833.210
## 1613.88	3742.941
## 1613.92	3640.876
## 1613.96	3543.994
## 1614.00	3466.639
## 1614.04	3397.384
## 1614.08	3333.046
## 1614.12	3274.894
## 1614.17	3224.885
## 1614.21	3180.907
## 1614.25	3124.709

## 1614.29	3089.135
## 1614.33	3069.024
## 1614.38	3066.479
## 1614.42	3090.538
## 1614.46	3140.367
## 1614.50	3210.397
## 1614.54	3304.902
## 1614.58	3413.997
## 1614.62	3544.466
## 1614.67	3672.348
## 1614.71	3787.980
## 1614.75	3873.700
## 1614.79	3923.727
## 1614.83	3934.756
## 1614.88	3907.250
## 1614.92	3852.507
## 1614.96	3774.641
## 1615.00	3682.070
## 1615.04	3588.423
## 1615.08	3507.812
## 1615.12	3439.824
## 1615.17	3386.179
## 1615.21	3344.616
## 1615.25	3311.174
## 1615.29	3293.028
## 1615.33	3285.042
## 1615.38	3288.152
## 1615.42	3301.632
## 1615.46	3320.014
## 1615.50	3339.556
## 1615.54	3366.059
## 1615.58	3407.299
## 1615.62	3452.787
## 1615.67	3512.483
## 1615.71	3572.052
## 1615.75	3632.842
## 1615.79	3677.325
## 1615.83	3705.454
## 1615.88	3706.402
## 1615.92	3696.141
## 1615.96	3661.656
## 1616.00	3614.055
## 1616.04	3555.928
## 1616.08	3492.118
## 1616.12	3435.204
## 1616.17	3383.477
## 1616.21	3338.452
## 1616.25	3298.743
## 1616.29	3264.265
## 1616.33	3242.352
## 1616.38	3207.314
## 1616.42	3183.145

## 1616.46	3184.660
## 1616.50	3196.364
## 1616.54	3220.502
## 1616.58	3274.988
## 1616.62	3330.851
## 1616.67	3400.564
## 1616.71	3464.342
## 1616.75	3524.959
## 1616.79	3575.908
## 1616.83	3613.778
## 1616.88	3635.272
## 1616.92	3639.659
## 1616.96	3629.284
## 1617.00	3605.323
## 1617.04	3571.404
## 1617.08	3530.110
## 1617.12	3485.717
## 1617.17	3432.852
## 1617.21	3378.418
## 1617.25	3324.319
## 1617.29	3279.159
## 1617.33	3240.138
## 1617.38	3213.429
## 1617.42	3195.081
## 1617.46	3191.429
## 1617.50	3205.168
## 1617.54	3239.116
## 1617.58	3289.883
## 1617.62	3348.939
## 1617.67	3409.320
## 1617.71	3466.369
## 1617.75	3527.837
## 1617.79	3594.486
## 1617.83	3657.905
## 1617.88	3704.364
## 1617.92	3731.392
## 1617.96	3738.477
## 1618.00	3727.184
## 1618.04	3701.613
## 1618.08	3665.919
## 1618.12	3619.538
## 1618.17	3577.359
## 1618.21	3537.718
## 1618.25	3502.568
## 1618.29	3474.452
## 1618.33	3446.047
## 1618.38	3414.176
## 1618.42	3381.679
## 1618.46	3356.688
## 1618.50	3346.843
## 1618.54	3369.289
## 1618.58	3424.216

## 1618.62	3491.079
## 1618.67	3557.310
## 1618.71	3631.328
## 1618.75	3704.973
## 1618.79	3761.858
## 1618.83	3799.336
## 1618.88	3828.414
## 1618.92	3837.678
## 1618.96	3834.112
## 1619.00	3810.834
## 1619.04	3793.615
## 1619.08	3776.165
## 1619.12	3771.018
## 1619.17	3771.638
## 1619.21	3777.657
## 1619.25	3786.233
## 1619.29	3794.844
## 1619.33	3801.357
## 1619.38	3798.022
## 1619.42	3788.386
## 1619.46	3788.534
## 1619.50	3790.576
## 1619.54	3798.140
## 1619.58	3807.843
## 1619.62	3836.740
## 1619.67	3864.635
## 1619.71	3894.601
## 1619.75	3920.764
## 1619.79	3935.172
## 1619.83	3949.858
## 1619.88	3956.386
## 1619.92	3957.822
## 1619.96	3952.828
## 1620.00	3944.648
## 1620.04	3933.447
## 1620.08	3922.143
## 1620.12	3904.964
## 1620.17	3889.409
## 1620.21	3869.986
## 1620.25	3839.190
## 1620.29	3809.183
## 1620.33	3782.762
## 1620.38	3755.069
## 1620.42	3732.732
## 1620.46	3707.543
## 1620.50	3678.454
## 1620.54	3654.760
## 1620.58	3634.799
## 1620.62	3612.850
## 1620.67	3595.410
## 1620.71	3579.364
## 1620.75	3558.105

## 1620.79	3543.556
## 1620.83	3540.654
## 1620.88	3545.598
## 1620.92	3555.717
## 1620.96	3567.671
## 1621.00	3577.653
## 1621.04	3591.561
## 1621.08	3594.310
## 1621.12	3596.912
## 1621.17	3593.240
## 1621.21	3591.732
## 1621.25	3585.177
## 1621.29	3583.299
## 1621.33	3581.878
## 1621.38	3580.573
## 1621.42	3579.564
## 1621.46	3577.303
## 1621.50	3573.029
## 1621.54	3550.901
## 1621.58	3528.491
## 1621.62	3499.468
## 1621.67	3467.088
## 1621.71	3422.492
## 1621.75	3374.927
## 1621.79	3325.395
## 1621.83	3279.176
## 1621.88	3239.042
## 1621.92	3207.423
## 1621.96	3184.319
## 1622.00	3176.569
## 1622.04	3178.372
## 1622.08	3190.003
## 1622.12	3206.152
## 1622.17	3225.630
## 1622.21	3252.017
## 1622.25	3286.893
## 1622.29	3331.637
## 1622.33	3380.578
## 1622.38	3430.793
## 1622.42	3471.415
## 1622.46	3504.500
## 1622.50	3526.044
## 1622.54	3531.102
## 1622.58	3521.929
## 1622.62	3499.537
## 1622.67	3460.306
## 1622.71	3408.759
## 1622.75	3347.996
## 1622.79	3283.964
## 1622.83	3219.605
## 1622.88	3159.248
## 1622.92	3099.610

## 1622.96	3053.503
## 1623.00	3017.297
## 1623.04	2992.486
## 1623.08	2978.559
## 1623.12	2978.909
## 1623.17	2990.538
## 1623.21	3012.484
## 1623.25	3043.096
## 1623.29	3080.068
## 1623.33	3122.632
## 1623.38	3175.928
## 1623.42	3234.239
## 1623.46	3296.270
## 1623.50	3350.093
## 1623.54	3405.887
## 1623.58	3461.484
## 1623.62	3502.811
## 1623.67	3539.026
## 1623.71	3562.605
## 1623.75	3564.696
## 1623.79	3552.446
## 1623.83	3533.475
## 1623.88	3499.506
## 1623.92	3447.637
## 1623.96	3379.858
## 1624.00	3304.287
## 1624.04	3224.817
## 1624.08	3139.635
## 1624.12	3065.003
## 1624.17	3002.349
## 1624.21	2947.593
## 1624.25	2907.122
## 1624.29	2880.656
## 1624.33	2869.355
## 1624.38	2876.543
## 1624.42	2901.196
## 1624.46	2933.830
## 1624.50	2976.886
## 1624.54	3025.439
## 1624.58	3078.582
## 1624.62	3133.798
## 1624.67	3187.976
## 1624.71	3238.134
## 1624.75	3282.087
## 1624.79	3317.530
## 1624.83	3348.741
## 1624.88	3379.040
## 1624.92	3403.000
## 1624.96	3411.150
## 1625.00	3405.744
## 1625.04	3372.784
## 1625.08	3298.371

## 1625.12	3213.557
## 1625.17	3117.091
## 1625.21	3012.100
## 1625.25	2909.985
## 1625.29	2820.274
## 1625.33	2744.766
## 1625.38	2686.514
## 1625.42	2641.555
## 1625.46	2613.294
## 1625.50	2600.731
## 1625.54	2603.358
## 1625.58	2620.078
## 1625.62	2645.860
## 1625.67	2680.096
## 1625.71	2718.486
## 1625.75	2758.754
## 1625.79	2801.464
## 1625.83	2843.583
## 1625.88	2881.995
## 1625.92	2910.523
## 1625.96	2930.519
## 1626.00	2947.068
## 1626.04	2957.517
## 1626.08	2966.622
## 1626.12	2968.756
## 1626.17	2962.356
## 1626.21	2938.407
## 1626.25	2894.386
## 1626.29	2834.712
## 1626.33	2763.382
## 1626.38	2691.126
## 1626.42	2617.061
## 1626.46	2549.343
## 1626.50	2490.838
## 1626.54	2443.415
## 1626.58	2408.750
## 1626.62	2384.139
## 1626.67	2370.694
## 1626.71	2366.514
## 1626.75	2374.030
## 1626.79	2390.743
## 1626.83	2413.219
## 1626.88	2439.857
## 1626.92	2467.214
## 1626.96	2493.832
## 1627.00	2517.558
## 1627.04	2538.773
## 1627.08	2558.995
## 1627.12	2576.928
## 1627.17	2592.780
## 1627.21	2605.604
## 1627.25	2613.364

## 1627.29	2614.364
## 1627.33	2610.440
## 1627.38	2598.432
## 1627.42	2573.851
## 1627.46	2538.677
## 1627.50	2495.169
## 1627.54	2447.482
## 1627.58	2399.374
## 1627.62	2353.933
## 1627.67	2310.546
## 1627.71	2272.186
## 1627.75	2242.864
## 1627.79	2223.062
## 1627.83	2213.702
## 1627.88	2214.087
## 1627.92	2223.288
## 1627.96	2239.384
## 1628.00	2260.938
## 1628.04	2286.596
## 1628.08	2312.963
## 1628.12	2338.658
## 1628.17	2362.100
## 1628.21	2381.595
## 1628.25	2397.095
## 1628.29	2409.769
## 1628.33	2417.843
## 1628.38	2423.817
## 1628.42	2426.606
## 1628.46	2428.721
## 1628.50	2428.510
## 1628.54	2424.950
## 1628.58	2414.249
## 1628.62	2398.325
## 1628.67	2377.991
## 1628.71	2353.619
## 1628.75	2324.276
## 1628.79	2293.371
## 1628.83	2262.684
## 1628.88	2234.931
## 1628.92	2212.005
## 1628.96	2195.648
## 1629.00	2185.342
## 1629.04	2182.571
## 1629.08	2186.867
## 1629.12	2197.701
## 1629.17	2214.534
## 1629.21	2236.662
## 1629.25	2262.012
## 1629.29	2288.798
## 1629.33	2315.090
## 1629.38	2338.796
## 1629.42	2357.723

## 1629.46	2369.390
## 1629.50	2376.481
## 1629.54	2378.385
## 1629.58	2377.492
## 1629.62	2371.375
## 1629.67	2365.115
## 1629.71	2354.885
## 1629.75	2343.860
## 1629.79	2331.384
## 1629.83	2318.077
## 1629.88	2302.582
## 1629.92	2285.456
## 1629.96	2269.645
## 1630.00	2252.839
## 1630.04	2236.748
## 1630.08	2220.736
## 1630.12	2207.782
## 1630.17	2198.785
## 1630.21	2195.293
## 1630.25	2197.427
## 1630.29	2204.616
## 1630.33	2215.986
## 1630.38	2231.021
## 1630.42	2249.398
## 1630.46	2269.222
## 1630.50	2288.347
## 1630.54	2304.013
## 1630.58	2314.925
## 1630.62	2319.857
## 1630.67	2321.282
## 1630.71	2319.016
## 1630.75	2316.215
## 1630.79	2310.788
## 1630.83	2304.221
## 1630.88	2297.020
## 1630.92	2289.917
## 1630.96	2280.093
## 1631.00	2268.630
## 1631.04	2254.056
## 1631.08	2237.664
## 1631.12	2221.371
## 1631.17	2203.875
## 1631.21	2187.524
## 1631.25	2172.788
## 1631.29	2161.263
## 1631.33	2155.486
## 1631.38	2154.910
## 1631.42	2159.496
## 1631.46	2168.173
## 1631.50	2180.495
## 1631.54	2194.566
## 1631.58	2209.608

## 1631.62	2224.250
## 1631.67	2237.341
## 1631.71	2248.733
## 1631.75	2257.865
## 1631.79	2265.189
## 1631.83	2269.894
## 1631.88	2272.421
## 1631.92	2273.089
## 1631.96	2277.108
## 1632.00	2279.678
## 1632.04	2284.668
## 1632.08	2285.006
## 1632.12	2279.971
## 1632.17	2270.417
## 1632.21	2258.348
## 1632.25	2241.420
## 1632.29	2222.573
## 1632.33	2202.752
## 1632.38	2184.003
## 1632.42	2167.194
## 1632.46	2154.298
## 1632.50	2145.788
## 1632.54	2141.999
## 1632.58	2141.600
## 1632.62	2145.390
## 1632.67	2152.503
## 1632.71	2162.230
## 1632.75	2173.103
## 1632.79	2184.136
## 1632.83	2193.830
## 1632.88	2203.505
## 1632.92	2212.921
## 1632.96	2224.037
## 1633.00	2235.280
## 1633.04	2247.621
## 1633.08	2259.585
## 1633.12	2273.289
## 1633.17	2289.850
## 1633.21	2306.198
## 1633.25	2323.912
## 1633.29	2334.806
## 1633.33	2342.450
## 1633.38	2344.584
## 1633.42	2342.111
## 1633.46	2330.946
## 1633.50	2314.049
## 1633.54	2292.790
## 1633.58	2265.670
## 1633.62	2237.329
## 1633.67	2210.248
## 1633.71	2187.195
## 1633.75	2169.079

## 1633.79	2154.965
## 1633.83	2147.212
## 1633.88	2145.341
## 1633.92	2148.647
## 1633.96	2154.689
## 1634.00	2163.424
## 1634.04	2172.783
## 1634.08	2182.926
## 1634.12	2192.383
## 1634.17	2202.862
## 1634.21	2215.617
## 1634.25	2231.360
## 1634.29	2246.862
## 1634.33	2262.022
## 1634.38	2277.342
## 1634.42	2289.280
## 1634.46	2296.966
## 1634.50	2299.867
## 1634.54	2303.287
## 1634.58	2303.606
## 1634.62	2296.304
## 1634.67	2282.131
## 1634.71	2264.828
## 1634.75	2241.236
## 1634.79	2217.152
## 1634.83	2193.140
## 1634.88	2171.878
## 1634.92	2152.521
## 1634.96	2138.377
## 1635.00	2128.989
## 1635.04	2125.037
## 1635.08	2124.416
## 1635.12	2126.555
## 1635.17	2131.775
## 1635.21	2138.335
## 1635.25	2146.123
## 1635.29	2153.949
## 1635.33	2162.154
## 1635.38	2171.976
## 1635.42	2187.017
## 1635.46	2200.595
## 1635.50	2213.985
## 1635.54	2229.173
## 1635.58	2245.225
## 1635.62	2261.816
## 1635.67	2283.149
## 1635.71	2299.502
## 1635.75	2315.846
## 1635.79	2321.690
## 1635.83	2320.038
## 1635.88	2312.096
## 1635.92	2295.329

## 1635.96	2273.206
## 1636.00	2248.205
## 1636.04	2222.413
## 1636.08	2198.608
## 1636.12	2176.532
## 1636.17	2158.690
## 1636.21	2145.349
## 1636.25	2136.265
## 1636.29	2131.019
## 1636.33	2130.065
## 1636.38	2136.959
## 1636.42	2149.054
## 1636.46	2163.574
## 1636.50	2180.766
## 1636.54	2202.017
## 1636.58	2228.005
## 1636.62	2255.174
## 1636.67	2285.544
## 1636.71	2313.689
## 1636.75	2331.184
## 1636.79	2346.158
## 1636.83	2355.177
## 1636.88	2358.402
## 1636.92	2358.807
## 1636.96	2355.045
## 1637.00	2343.715
## 1637.04	2327.478
## 1637.08	2306.796
## 1637.12	2279.770
## 1637.17	2250.244
## 1637.21	2219.730
## 1637.25	2189.765
## 1637.29	2158.955
## 1637.33	2132.429
## 1637.38	2117.049
## 1637.42	2111.077
## 1637.46	2111.624
## 1637.50	2118.896
## 1637.54	2134.726
## 1637.58	2156.419
## 1637.62	2183.734
## 1637.67	2209.035
## 1637.71	2239.661
## 1637.75	2273.360
## 1637.79	2305.448
## 1637.83	2334.834
## 1637.88	2358.904
## 1637.92	2374.250
## 1637.96	2379.554
## 1638.00	2374.284
## 1638.04	2364.716
## 1638.08	2350.894

## 1638.12	2336.592
## 1638.17	2317.876
## 1638.21	2296.301
## 1638.25	2271.995
## 1638.29	2245.226
## 1638.33	2214.354
## 1638.38	2186.943
## 1638.42	2164.878
## 1638.46	2149.196
## 1638.50	2136.401
## 1638.54	2132.666
## 1638.58	2136.695
## 1638.62	2147.853
## 1638.67	2167.850
## 1638.71	2193.309
## 1638.75	2226.310
## 1638.79	2261.793
## 1638.83	2299.057
## 1638.88	2331.331
## 1638.92	2363.640
## 1638.96	2389.919
## 1639.00	2411.262
## 1639.04	2423.796
## 1639.08	2432.557
## 1639.12	2433.008
## 1639.17	2427.049
## 1639.21	2415.524
## 1639.25	2396.470
## 1639.29	2373.115
## 1639.33	2345.345
## 1639.38	2314.833
## 1639.42	2282.042
## 1639.46	2255.314
## 1639.50	2234.477
## 1639.54	2221.010
## 1639.58	2212.070
## 1639.62	2209.155
## 1639.67	2211.693
## 1639.71	2222.155
## 1639.75	2241.611
## 1639.79	2267.713
## 1639.83	2299.972
## 1639.88	2337.803
## 1639.92	2381.134
## 1639.96	2424.324
## 1640.00	2467.896
## 1640.04	2507.145
## 1640.08	2537.913
## 1640.12	2560.758
## 1640.17	2575.212
## 1640.21	2581.391
## 1640.25	2575.429

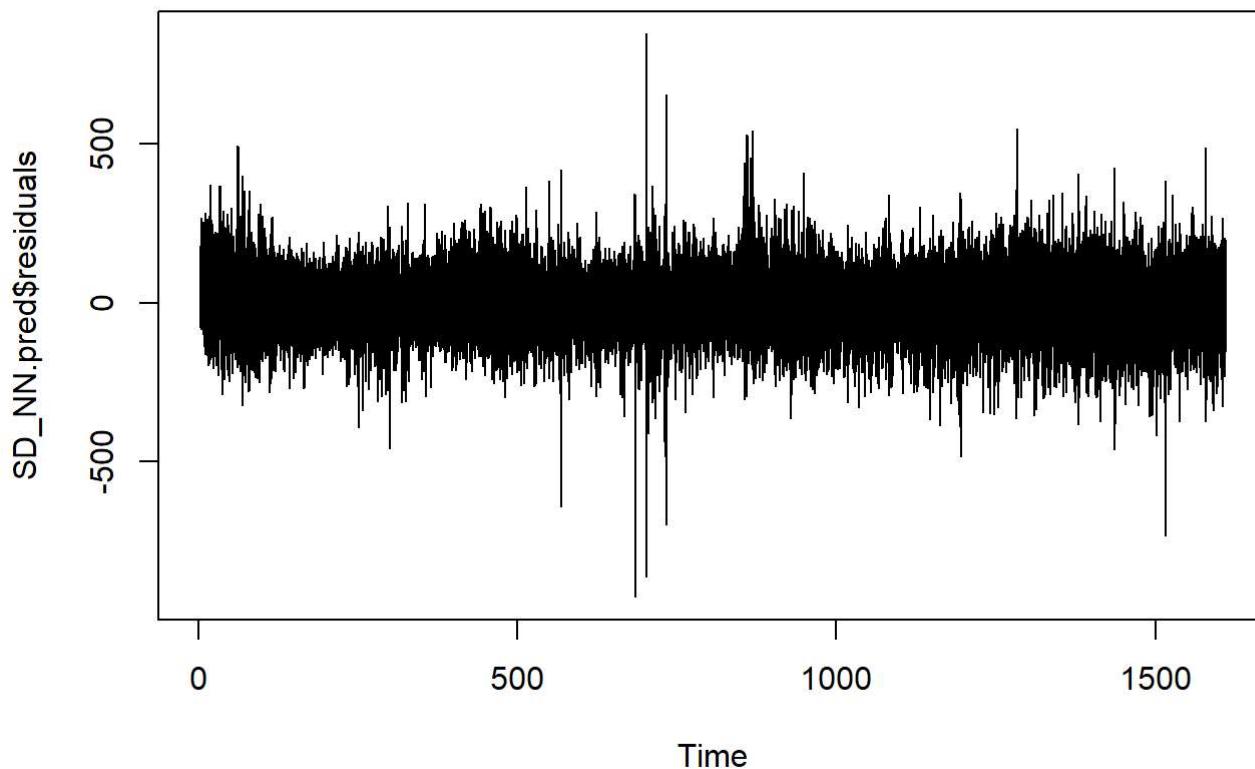
## 1640.29	2559.802
## 1640.33	2540.661
## 1640.38	2517.380
## 1640.42	2491.302
## 1640.46	2460.858
## 1640.50	2428.388
## 1640.54	2395.320
## 1640.58	2363.177
## 1640.62	2339.751
## 1640.67	2325.169
## 1640.71	2323.992
## 1640.75	2327.228
## 1640.79	2340.448
## 1640.83	2361.006
## 1640.88	2392.231
## 1640.92	2430.775
## 1640.96	2476.833
## 1641.00	2525.148
## 1641.04	2571.740
## 1641.08	2618.323
## 1641.12	2657.737
## 1641.17	2688.518
## 1641.21	2704.681
## 1641.25	2709.717
## 1641.29	2704.009
## 1641.33	2689.187
## 1641.38	2670.880
## 1641.42	2644.152
## 1641.46	2612.959
## 1641.50	2578.200
## 1641.54	2541.851
## 1641.58	2506.041
## 1641.62	2472.396
## 1641.67	2441.937
## 1641.71	2416.691
## 1641.75	2392.788
## 1641.79	2376.765
## 1641.83	2362.630
## 1641.88	2353.009
## 1641.92	2348.751
## 1641.96	2353.967
## 1642.00	2363.740
## 1642.04	2383.583
## 1642.08	2411.039
## 1642.12	2448.229
## 1642.17	2491.471
## 1642.21	2541.459
## 1642.25	2592.927
## 1642.29	2642.304
## 1642.33	2689.768
## 1642.38	2730.858
## 1642.42	2758.904

## 1642.46	2771.041
## 1642.50	2767.936
## 1642.54	2752.045
## 1642.58	2724.674
## 1642.62	2688.492
## 1642.67	2641.227
## 1642.71	2584.277
## 1642.75	2522.593
## 1642.79	2460.657
## 1642.83	2403.275
## 1642.88	2350.256
## 1642.92	2305.576
## 1642.96	2269.742
## 1643.00	2243.596
## 1643.04	2227.275
## 1643.08	2220.759
## 1643.12	2224.436
## 1643.17	2237.519
## 1643.21	2259.457
## 1643.25	2289.289
## 1643.29	2324.051
## 1643.33	2362.707
## 1643.38	2403.504
## 1643.42	2446.106
## 1643.46	2484.444
## 1643.50	2521.417
## 1643.54	2550.871
## 1643.58	2571.337
## 1643.62	2576.328
## 1643.67	2571.047
## 1643.71	2553.276
## 1643.75	2523.731
## 1643.79	2484.830
## 1643.83	2437.178
## 1643.88	2389.188
## 1643.92	2338.477
## 1643.96	2289.520
## 1644.00	2243.548
## 1644.04	2202.362
## 1644.08	2167.790
## 1644.12	2140.964
## 1644.17	2122.588
## 1644.21	2113.479
## 1644.25	2114.715
## 1644.29	2124.998
## 1644.33	2145.729
## 1644.38	2174.524
## 1644.42	2214.387
## 1644.46	2263.191
## 1644.50	2320.284
## 1644.54	2382.010
## 1644.58	2441.273

```
## 1644.62      2491.667
## 1644.67      2529.388
## 1644.71      2554.386
## 1644.75      2565.062
## 1644.79      2561.416
## 1644.83      2543.501
## 1644.88      2513.317
## 1644.92      2473.718
## 1644.96      2428.749
## 1645.00      2380.680
## 1645.04      2330.033
## 1645.08      2279.754
## 1645.12      2231.862
## 1645.17      2189.053
## 1645.21      2152.698
## 1645.25      2122.262
## 1645.29      2099.584
## 1645.33      2086.161
## 1645.38      2082.277
## 1645.42      2086.996
## 1645.46      2100.759
## 1645.50      2124.025
## 1645.54      2157.063
## 1645.58      2199.935
## 1645.62      2250.309
## 1645.67      2304.538
## 1645.71      2359.201
## 1645.75      2408.832
## 1645.79      2450.332
## 1645.83      2480.947
## 1645.88      2499.112
## 1645.92      2503.876
## 1645.96      2491.725
## 1646.00      2464.714
## 1646.04      2425.627
## 1646.08      2375.604
## 1646.12      2320.070
```

```
# Plot the errors for the training period
```

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

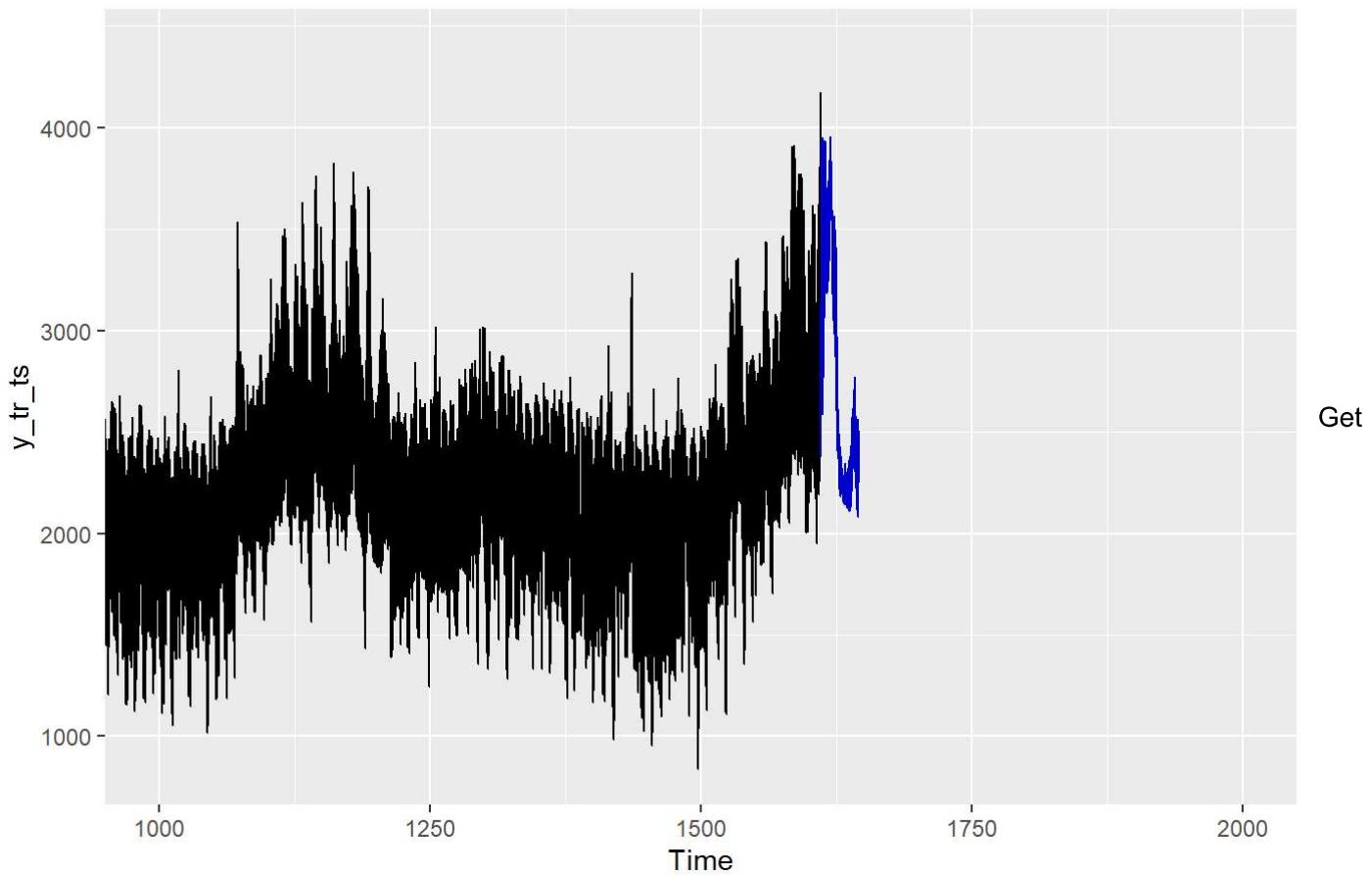
Figure 2.20: Neural Net Residual Plot for Training Period

```
#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")
```

Figure 2.21: Forecasts for Neural Network Model



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)

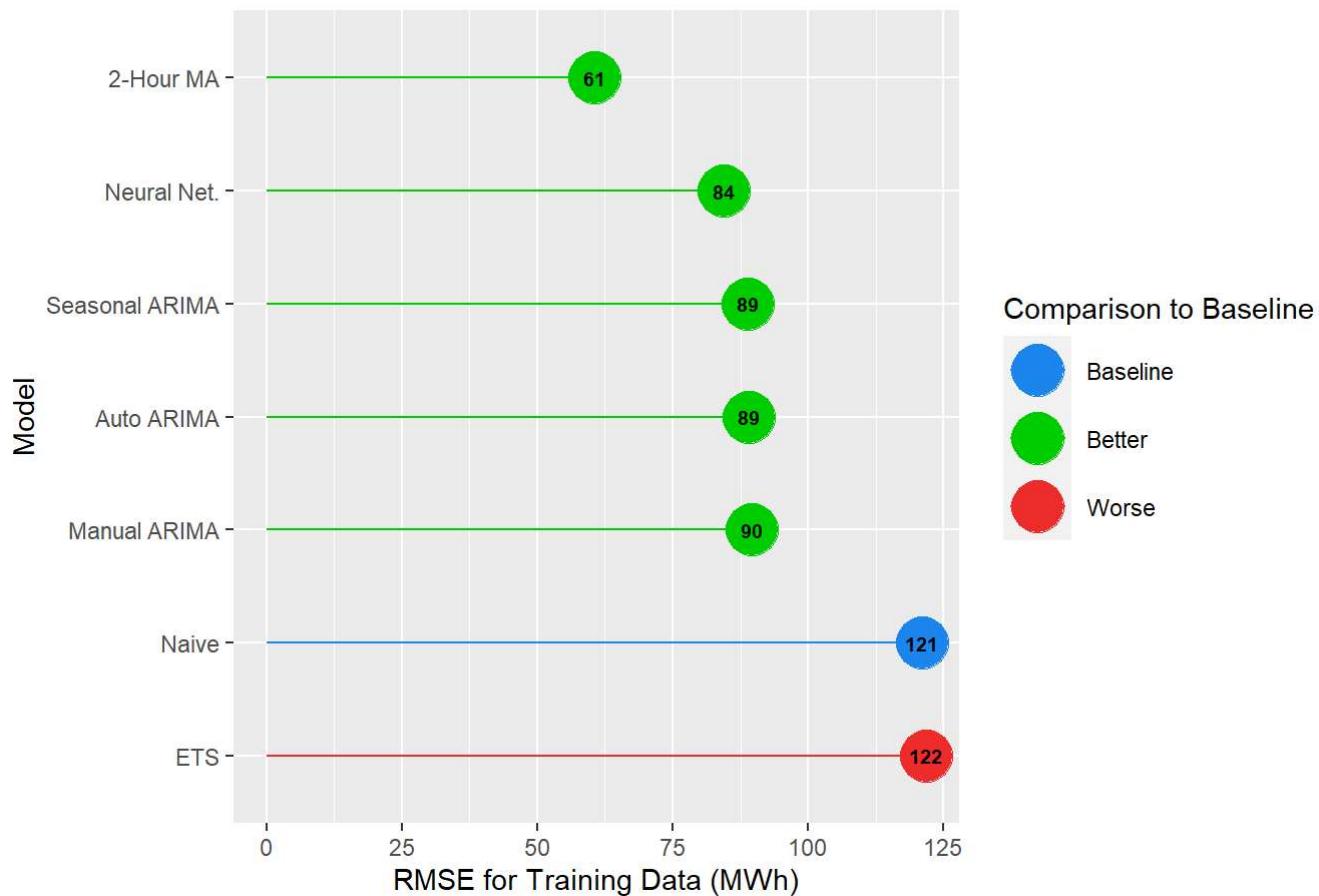
Results_df <- Results_df %>% mutate(BaselineComp_tr = ifelse(Tr_RMSE == Naive_rmse_tr, "Baseline", ifelse(Tr_RMSE < Naive_rmse_tr, "Better", "Worse")))
Results_df <- Results_df %>% mutate(BaselineComp_val = ifelse(Val_RMSE == Naive_rmse_val, "Baseline", ifelse(Val_RMSE < Naive_rmse_val, "Better", "Worse")))

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")
```

Figure 3.1: Model Results for Training Data



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 Data")
+ xlab("Model") + geom_text(aes(label = round(Val_RMSE)), color = "black", size = 2.5, fontface = "bold")
```

Figure 3.2: Model Results for Validation Data

