DERM Final Project Proposal

Joseph Keogh

11/19/2020

Introduction

With climate change ever looming societies, industries and governments are looking to move away from fossil fuels and towards more renewable energy. The main benefit of renewable energy is that they do not harm the environment in the process of producing electricity and subsequently power. One large problem with renewable energy is the lack of uniform production of the energy. Wind, solar, and tides have ever changing power outputs. The result of this non-uniform output is a surplus of energy at times, and a lack of energy at others.

The challenge that this produces is where to store the electricity when there is a surplus, and where to draw from when there is a deficit. This is where a Distributed Electrical Resource Management (DERM) system comes into play. The idea is to decentralize electricity storage from the electricity companies down to individuals. An individual for example would use their electric car to store electricity when there is a surplus in the system.

This individual would be providing the service of storing electricity while benefiting monetarily when they sell back the electricity to the provider.

The research question is exploring if it is possible/ reasonable to pursue a DERM system in order to help facilitate the move over to renewable power. To help answer this question we will use modeling and forecasting techniques to create a model for citizens to use to store and sell electricity. We will simulate electric car owners utilizing this system for personal gain

The data and the data-generating process

We are using two datasets: lmp and forecasts mwhours both come from pjm a -PJM is a regional transmission organization (RTO) that coordinates the movement of wholesale electricity in all or parts of 13 states and the District of Columbia (PJM.com). PJM collects the data by monitoring the price that electricity providers are charging users in real time. They are able to do this because they facilitate the transfer of electricity from the power company to the citizens The data is valid because PJM has direct access to the LMP as it is a distributer. Forecasts cannot be validated as no one can know the future load on the electrical system. We will however be using the forecasts to provide added information to our model

Both datasets are collected at 5 minute intervals

The lmp data is queried from the PJM database api at https://api.pjm.com/api/v1/rt_unverified_fivemin_lmps

The forecast data is downloaded manually from https://dataminer2.pjm.com/feed/very_short_load_frcst by selecting the specific data and time we will be looking at

Generate the Data

Retrieve data from Pre loaded database

Connect to the DB

Here we connect to our personal database where we have stored the PJM data

Grab data from DB

Here we download the data from our personal database into dataframes using SQL queries.

Check data output to ensure it worked

Understand Data

In this section we are going to do our best to understand the data that we are working with. This is can also be referred to as 'Data Exploration'.

Format the data

The raw data we have recieved is likely not in a perfect format for analysis. We need to format the data into a more usable schema

Format the forecasts data

```
# one hour in seconds
hour_in_seconds <- 60*60

forecasts_hour <- forecasts %>%

  # only collect forecasts that are one hour in the future
  mutate(time_diff = forecast_datetime - evaluation_datetime) %>%
  subset(time_diff==hour_in_seconds) %>%

  # format the names to match the response for joining
  mutate(datetime = forecast_datetime) %>%
  mutate(pnode_name = forecast_area) %>%
  select(datetime, pnode_name, forecast_load_mw)
```

Check the formatting

```
##              datetime pnode_name forecast_load_mw
## 1 2020-11-19 00:05:00        AEP            14641
## 2 2020-11-19 00:00:00        AEP            14606
## 3 2020-11-18 23:55:00        AEP            14598
## 4 2020-11-18 23:50:00        AEP            14633
## 5 2020-11-18 23:45:00        AEP            14678
```

```
##     datetime                     pnode_name        forecast_load_mw
##  Min.   :2020-10-20 22:00:00   Length:22136       Min.   : 2002
##  1st Qu.:2020-10-26 03:05:00   Class :character   1st Qu.: 4751
##  Median :2020-10-31 08:32:30   Mode  :character   Median : 7959
##  Mean   :2020-11-04 04:30:39                      Mean   : 8067
##  3rd Qu.:2020-11-13 22:55:00                      3rd Qu.:11036
##  Max.   :2020-11-19 07:55:00                      Max.   :16959
```

We choose to simplify the analysis by only looking at forecasts that are made one hour prior to the forecasted time. The goal is to create a one to one mapping of forecasts to real time LMP.

This time interval can easily be changed in further analysis. It is recommeded that other time intervals, and a many to one mapping be explored in further analysis.

Combine forecasts with correct lmp

```
combined_response <- left_join(response, forecasts_hour, id=c("datetime", "pnode_name")) %>% na.omit
```

```
## Joining, by = c("datetime", "pnode_name")
```

Here we do the actual one to one mapping of the forecast data to the LMP data. For the remainder of the proposal we will refer to two main datasets. The first being 'LMP Data' which is the LMP data that was not combined with the forecasts, and 'Combined Data' which is the LMP data matched with the Forecast data.

Inspect the combined data

The combined data has 21.730446% of the original LMP data

The combined data has 2.6786313% of the original Forecasts data

The combined data has 99.2771955% of the Combined data

```
##                 datetime pnode_id pnode_name total_lmp_rt
## 25306 2020-10-20 22:00:00  8445784        AEP        21.35
## 25307 2020-10-20 22:00:00  8445784        AEP        21.35
## 25308 2020-10-20 22:05:00  8445784        AEP        22.05
## 25309 2020-10-20 22:05:00  8445784        AEP        22.05
## 25310 2020-10-20 22:10:00  8445784        AEP        21.25
##       forecast_load_mw
## 25306            13454
## 25307            12877
## 25308            13477
## 25309            12732
## 25310            13417
```

```
##     datetime                        pnode_id          pnode_name
##  Min.   :2020-10-20 22:00:00   Min.   :  8445784   Length:21976
##  1st Qu.:2020-10-26 03:35:00   1st Qu.: 26930724   Class :character
##  Median :2020-10-31 21:10:00   Median : 74553062   Mode  :character
##  Mean   :2020-11-04 05:33:27   Mean   : 70407001
##  3rd Qu.:2020-11-13 23:05:00   3rd Qu.:118029338
```

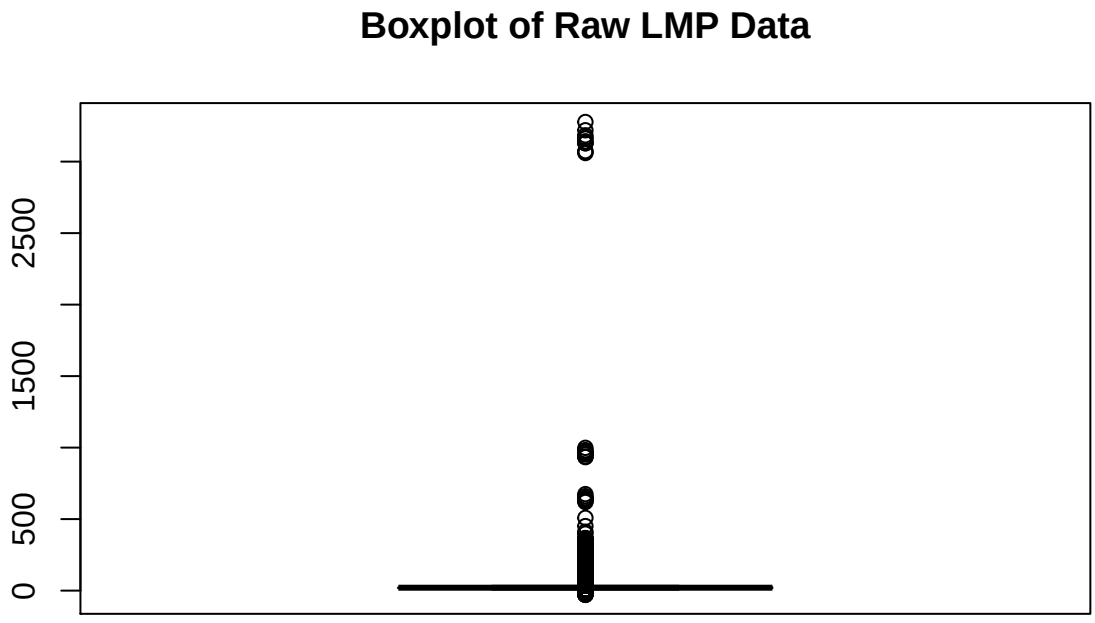
```
##  Max.   :2020-11-19 07:55:00   Max.   :124076095
##   total_lmp_rt    forecast_load_mw
##  Min.   :-14.26   Min.   : 2002
##  1st Qu.: 15.74   1st Qu.: 4751
##  Median : 18.37   Median : 7960
##  Mean   : 20.46   Mean   : 8068
##  3rd Qu.: 21.40   3rd Qu.:11036
##  Max.   :367.42   Max.   :16959
```

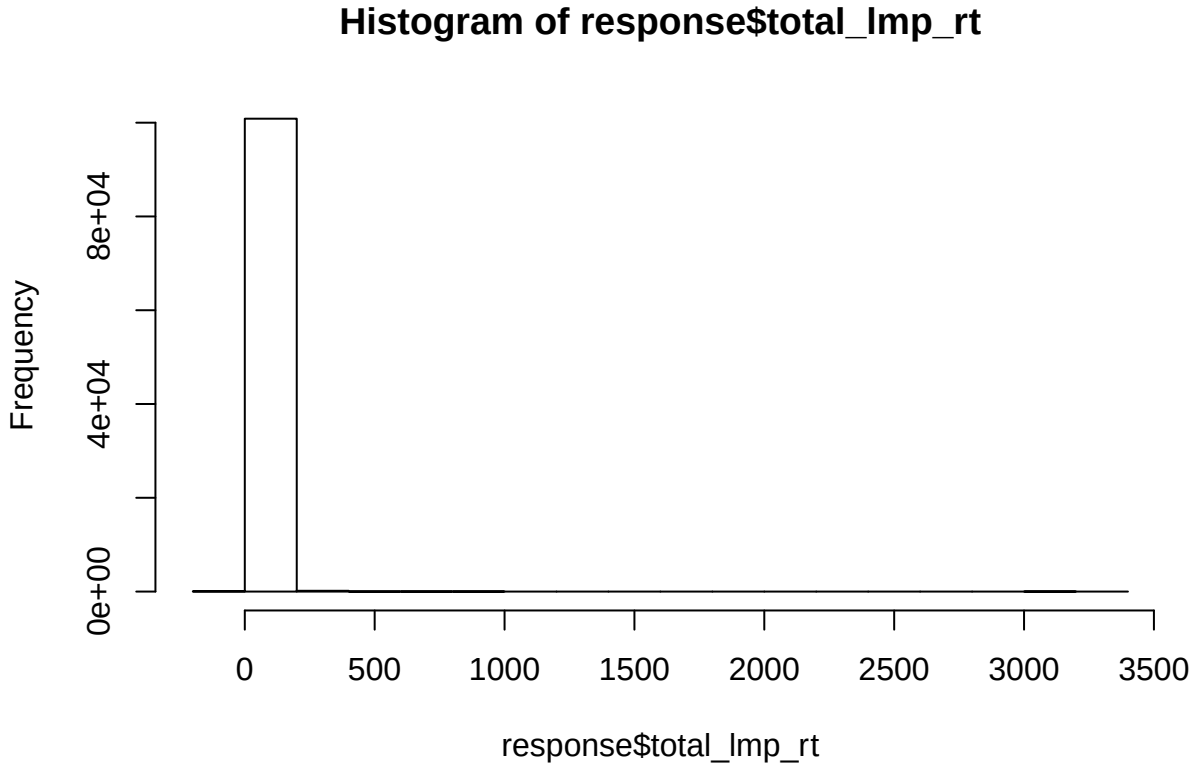
The time difference we are looking at

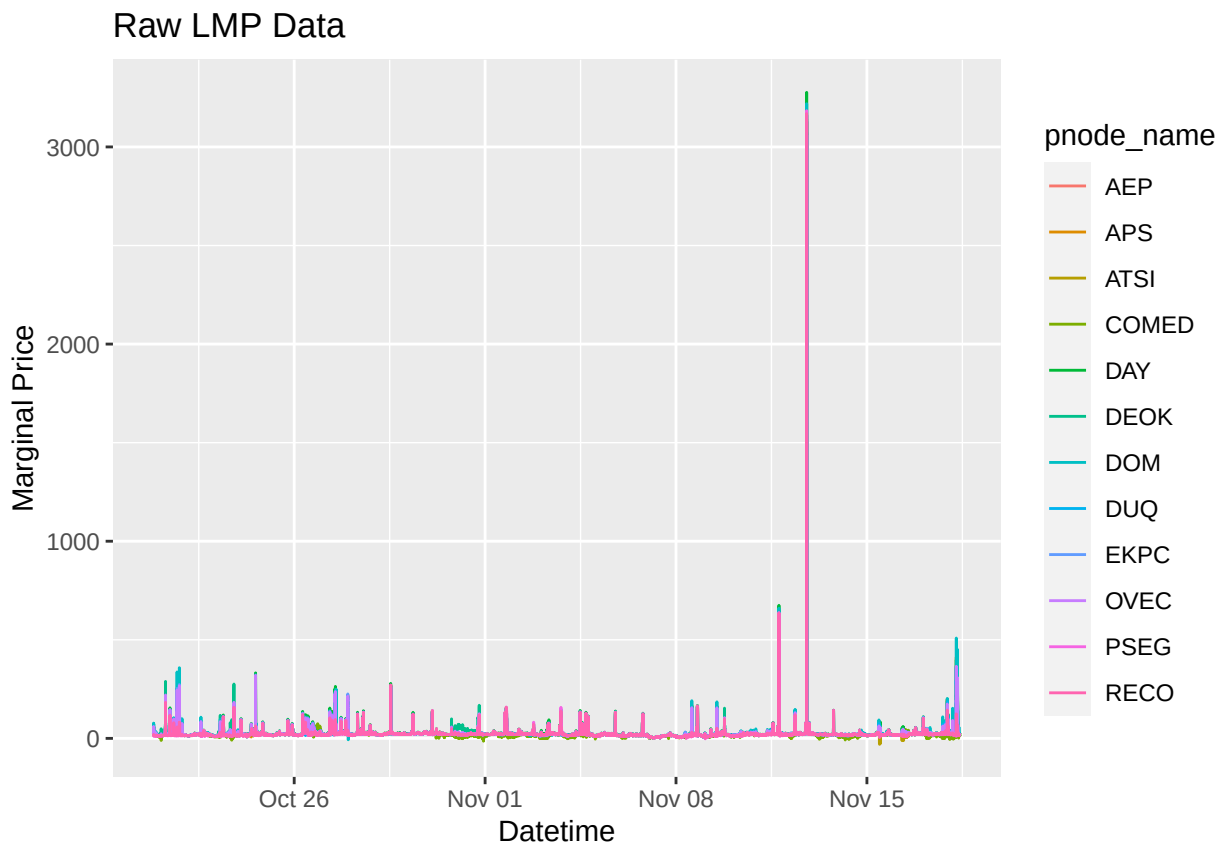
The LMP only data is over a span of 29.59375 days

The Combined data is over a span of 29.4548611 days

Visualize the LMP Data

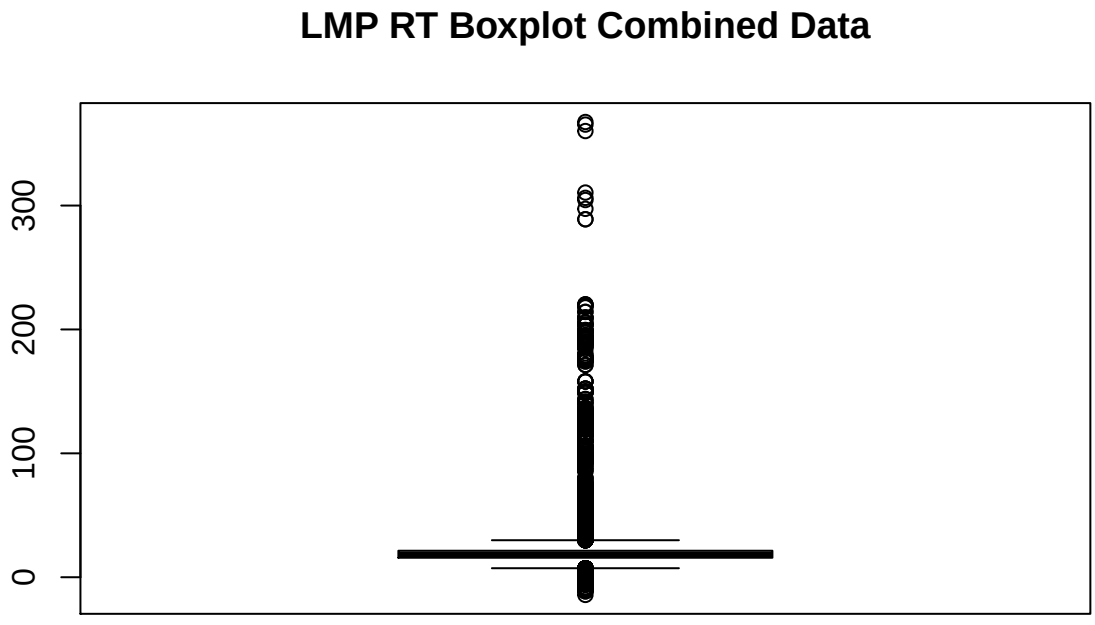


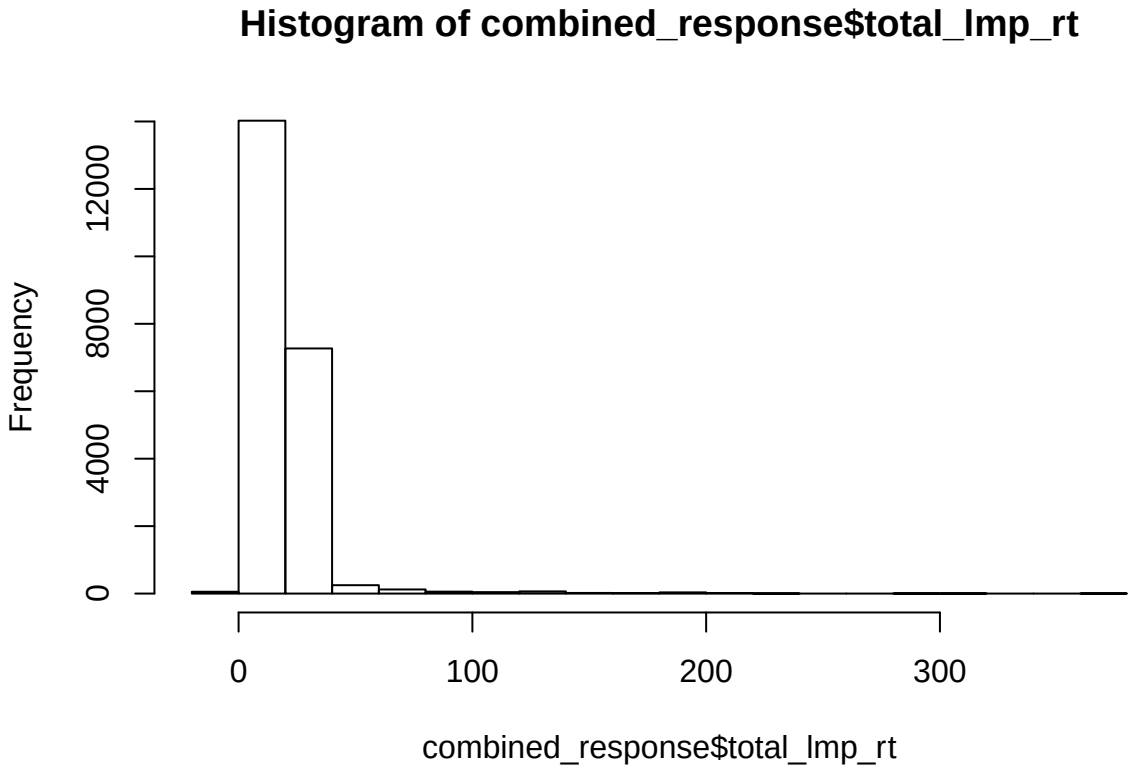


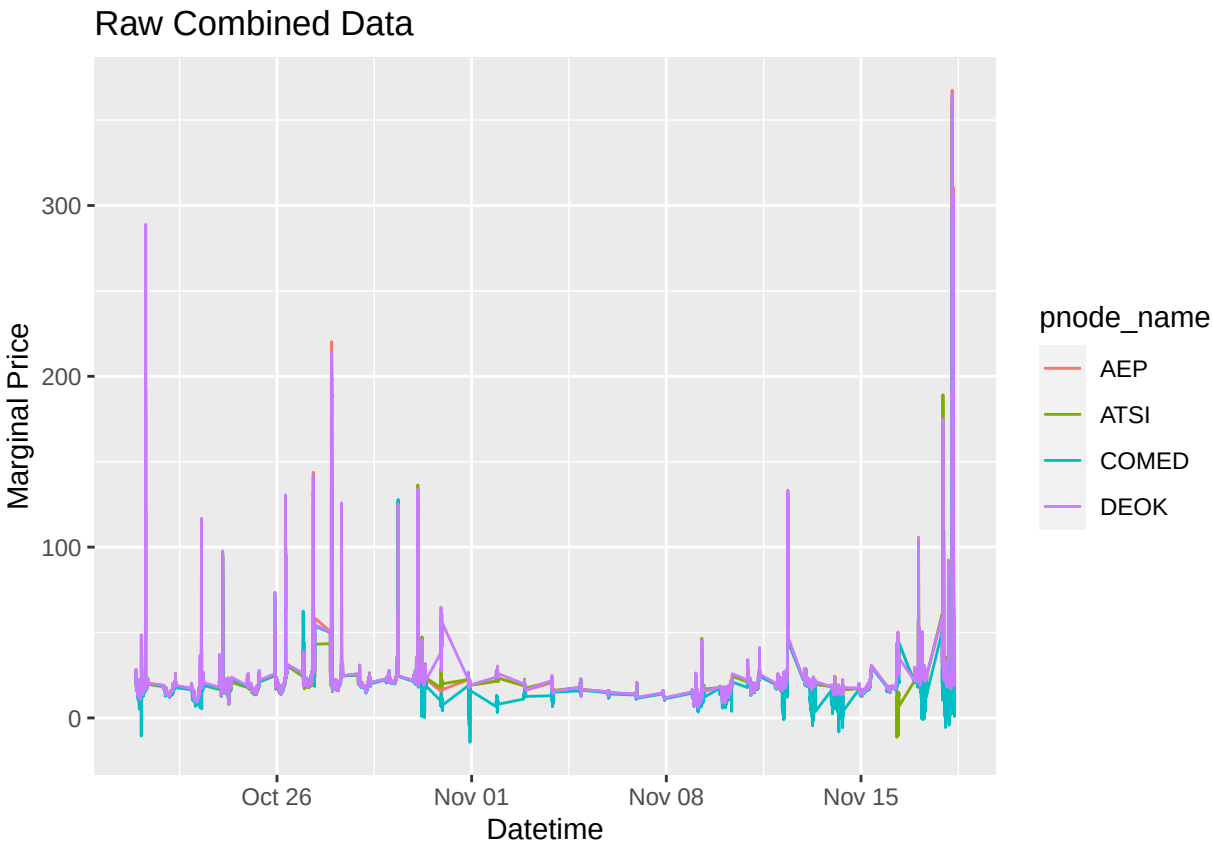


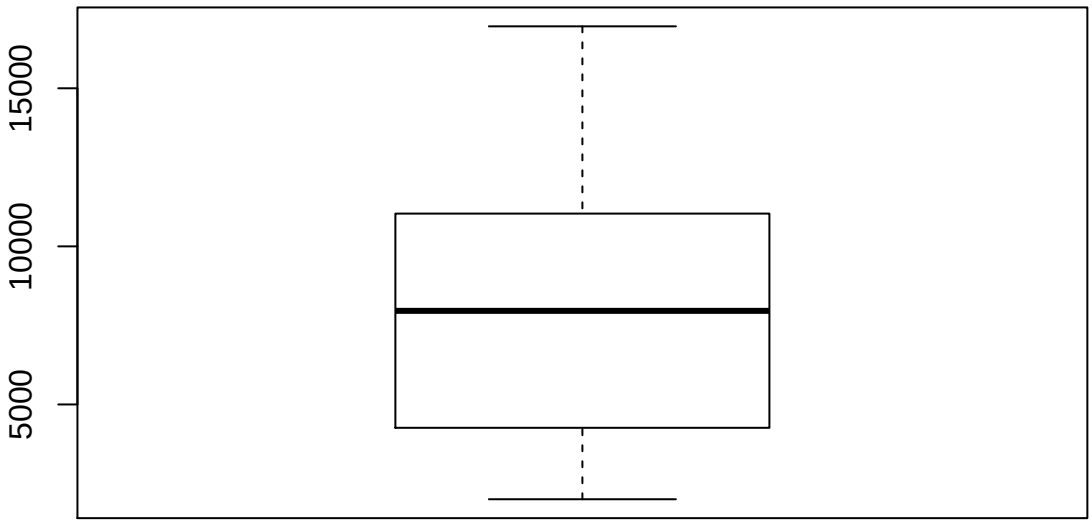
Severe outliers in the data

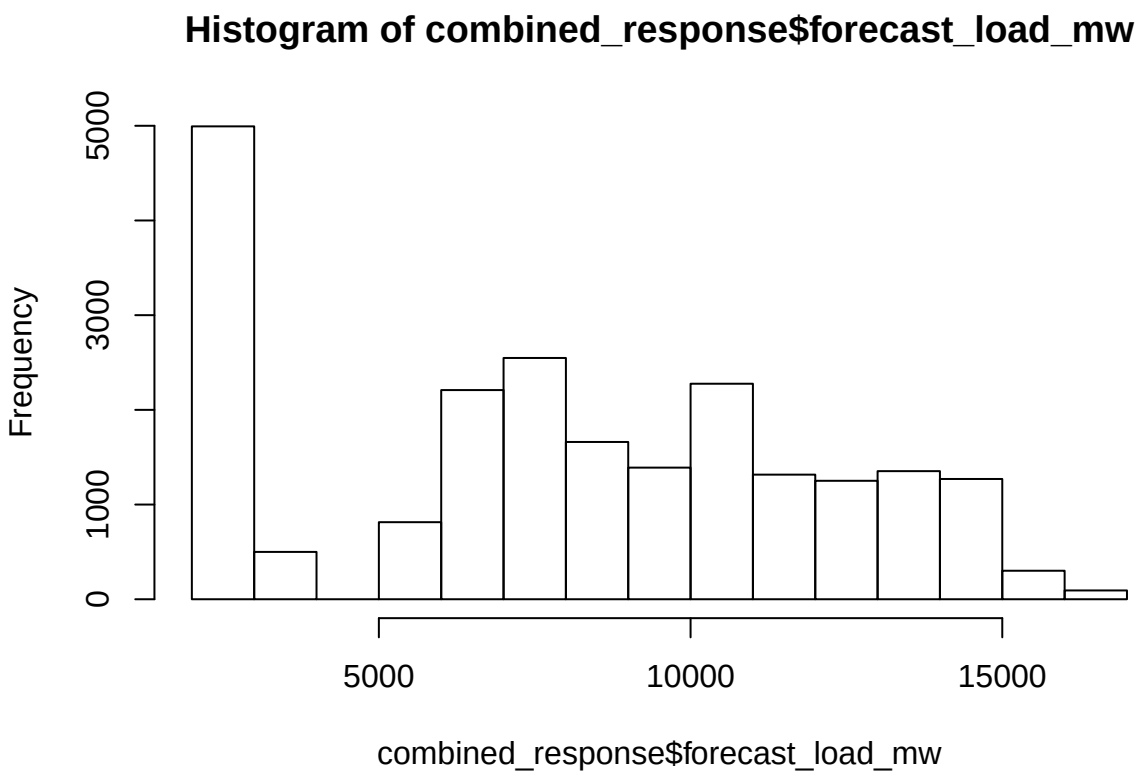
Visualize the combined data

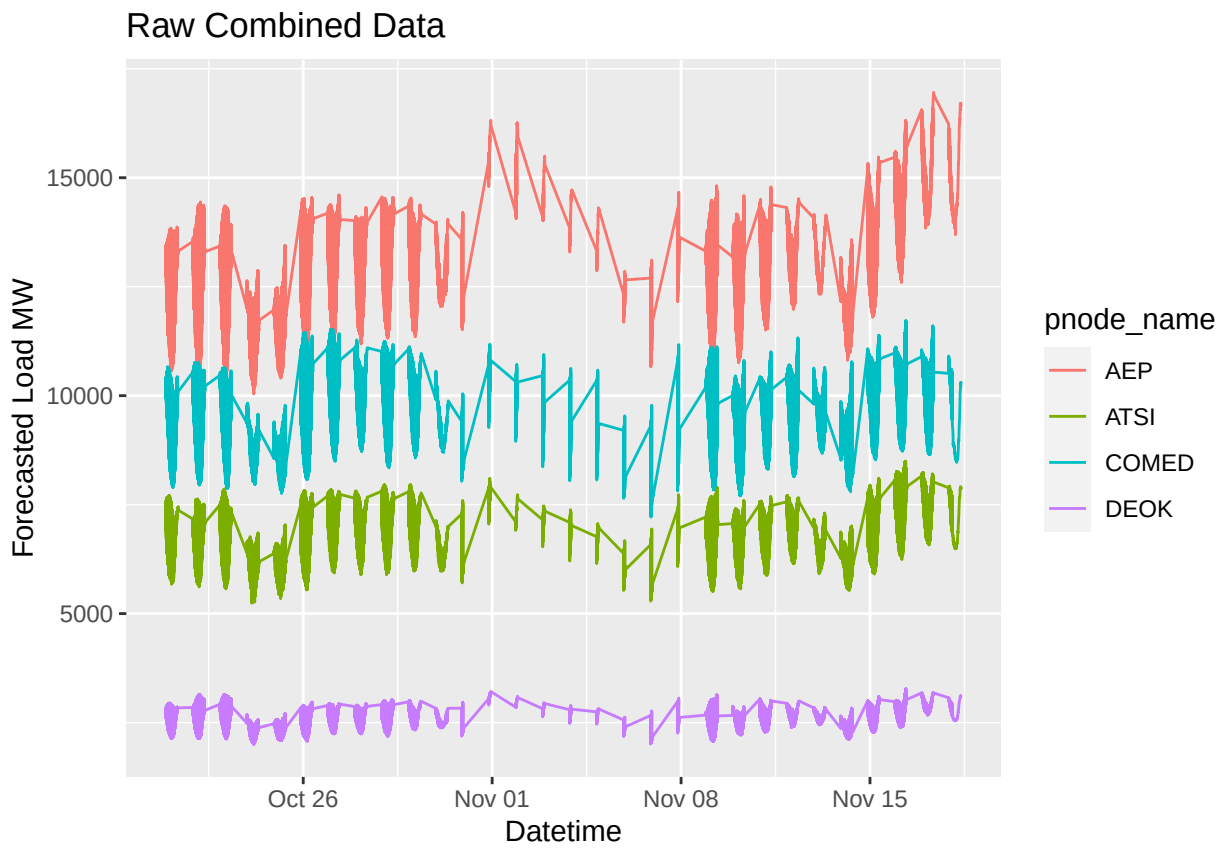





Raw Combined Data
Marginal Price
300
200
100
0
Oct 26
Nov 01
Nov 08
Nov 15
Datetime
pnode_name
AEP
ATSI
COMED
DEOK


Forecasts Load MW Boxplot Combined Data
15000
10000
5000


Histogram of combined_response$forecast_load_mw
Frequency
0
1000
3000
5000
5000
10000
15000
combined_response$forecast_load_mw

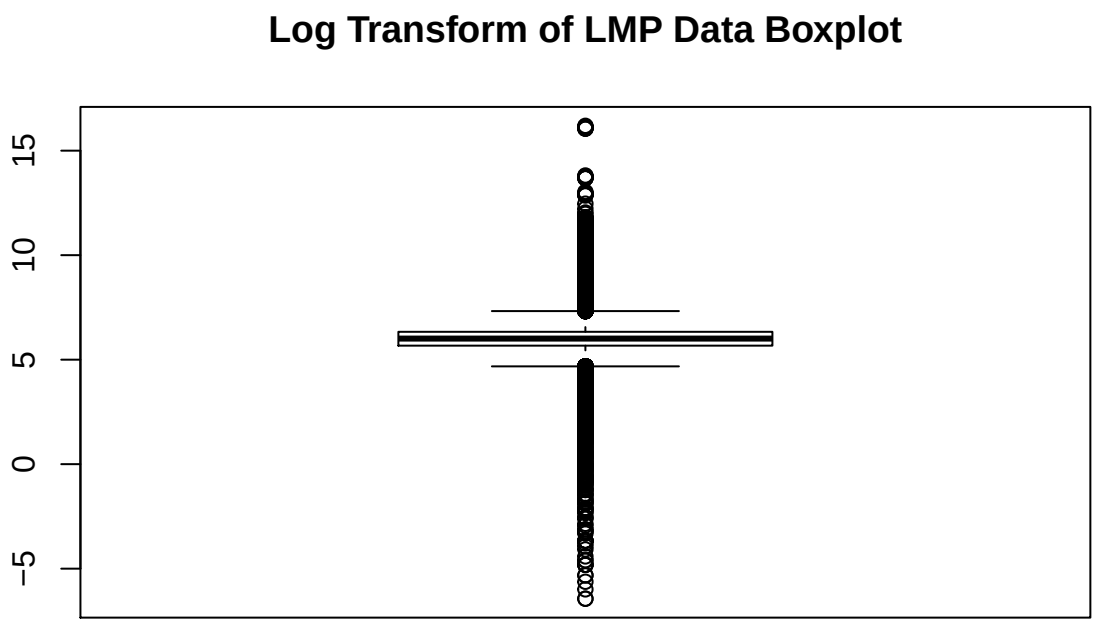


Very skewed LMP data, likely due to a few large outliers

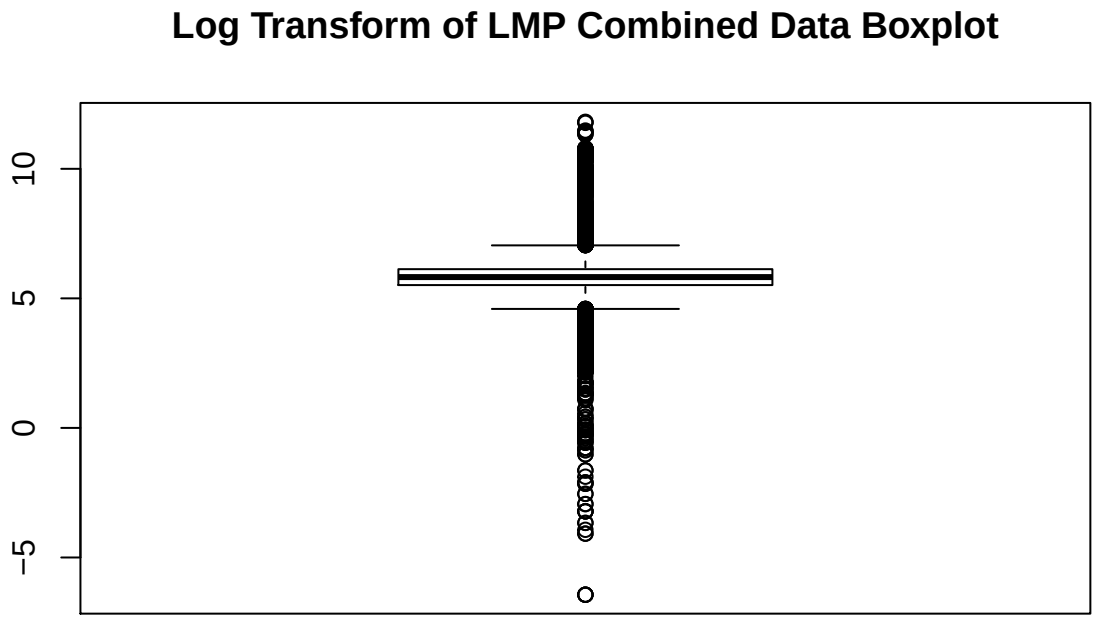
Forecast MW data seems to be more uniformly distributed, with periods of no data during the day

Can we make the data look better?

Log transform of LMP data



Log transform of combined data



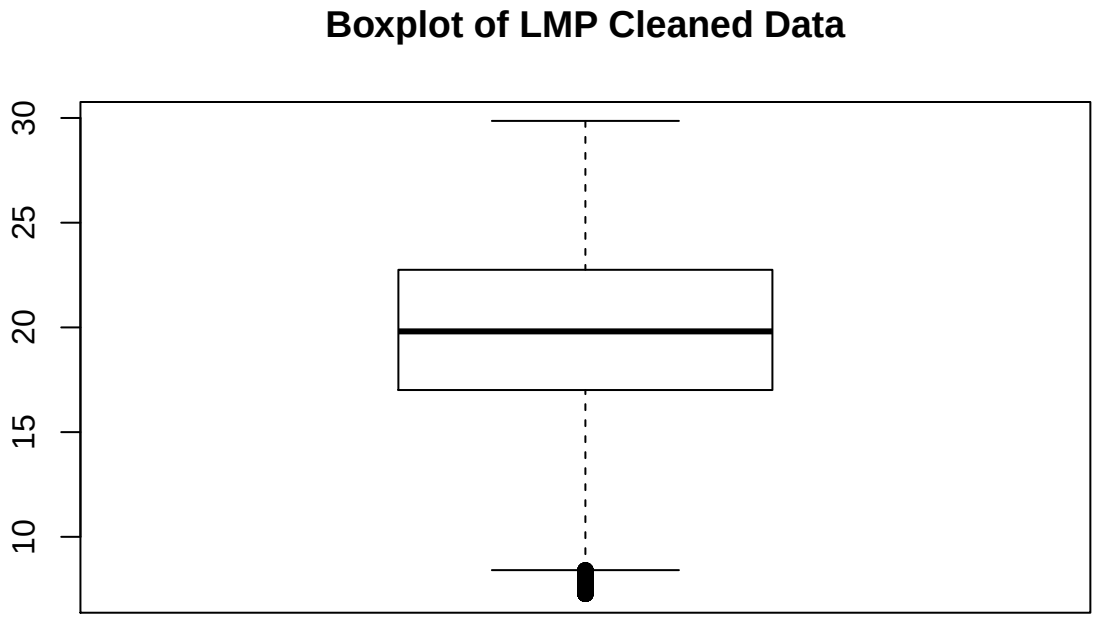
Log transform results in data looking slightly more normally distributed, but still not ideal as outliers still have a large impact on the distribution

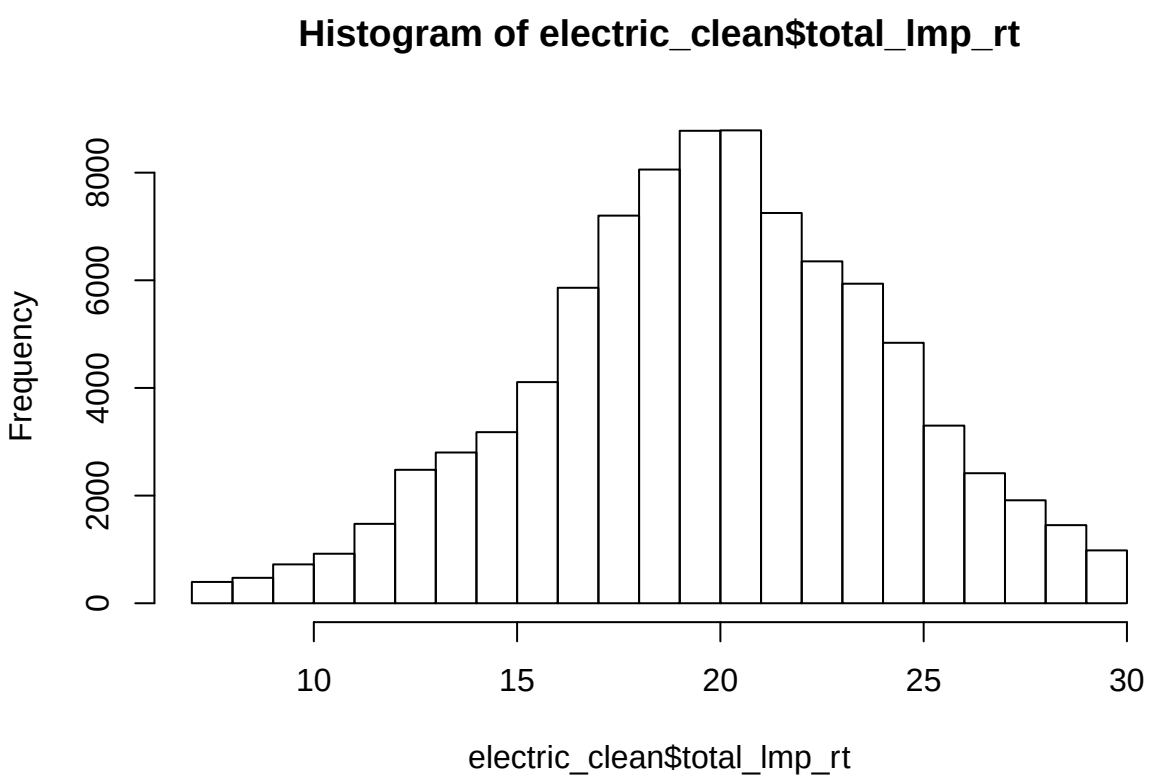
Decide on cleaned data

```
extrm_high <- boxplot(combined_response$total_lmp_rt, plot=FALSE)$stats[5]
extrm_low <- boxplot(combined_response$total_lmp_rt, plot=FALSE)$stats[1]

combined_clean <- combined_response %>%
  filter(total_lmp_rt < extrm_high) %>%
  filter(total_lmp_rt > extrm_low) %>%
  mutate(date = date(datetime)) %>%
  mutate(time = time(datetime)) %>%
  na.omit()

electric_clean <- response %>%
  filter(total_lmp_rt < extrm_high) %>%
  filter(total_lmp_rt > extrm_low) %>%
  mutate(date = date(datetime)) %>%
  mutate(time = time(datetime)) %>%
  na.omit()
```

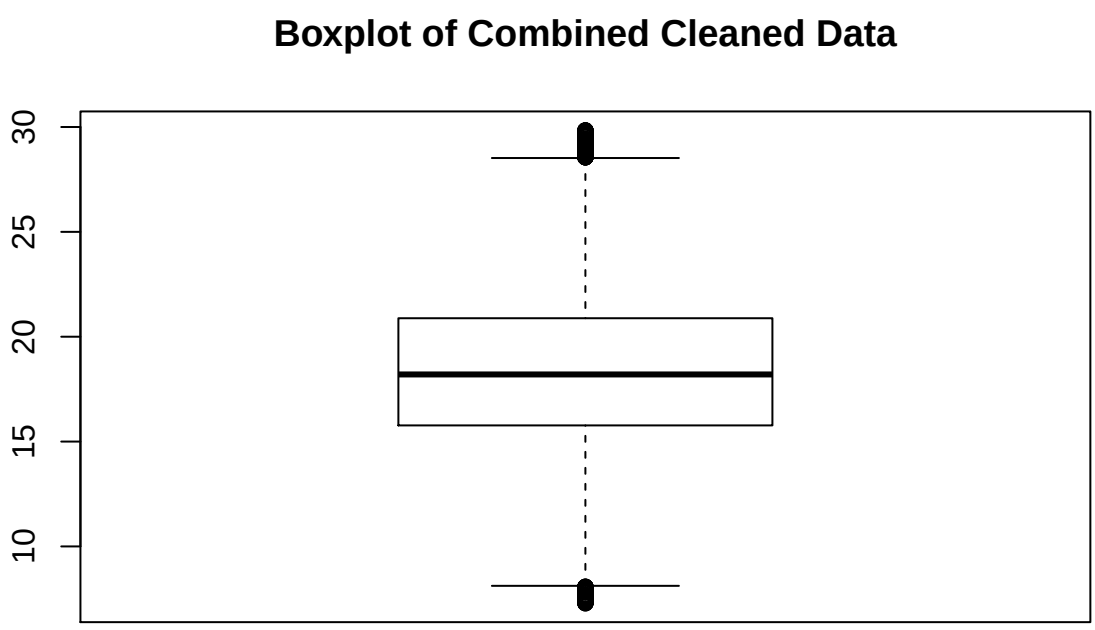
Inspect cleaned data

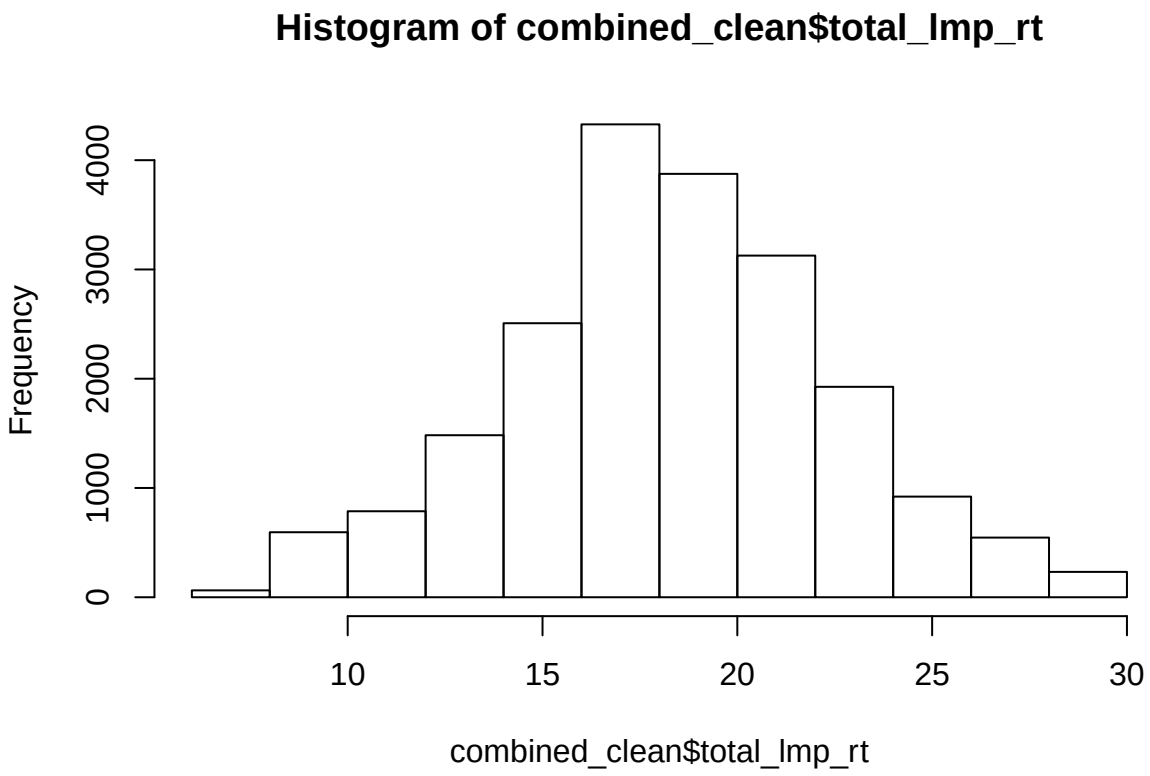
```
##     datetime                       pnode_id          pnode_name
##  Min.   :2020-10-20 20:00:00   Min.   :5.130e+04   Length:89679
##  1st Qu.:2020-10-28 11:50:00   1st Qu.:8.395e+06   Class :character
##  Median :2020-11-04 20:05:00   Median :3.451e+07   Mode  :character
##  Mean   :2020-11-04 16:43:50   Mean   :2.569e+08
##  3rd Qu.:2020-11-12 05:35:00   3rd Qu.:1.160e+08
##  Max.   :2020-11-19 09:15:00   Max.   :1.710e+09
##   total_lmp_rt        date                time
##  Min.   : 7.28   Min.   :2020-10-20   Min.   :    1
##  1st Qu.:17.01   1st Qu.:2020-10-28   1st Qu.:22421
##  Median :19.81   Median :2020-11-04   Median :44840
##  Mean   :19.74   Mean   :2020-11-04   Mean   :44840
##  3rd Qu.:22.75   3rd Qu.:2020-11-12   3rd Qu.:67260
##  Max.   :29.86   Max.   :2020-11-19   Max.   :89679
```






```
##     datetime                       pnode_id          pnode_name
##  Min.   :2020-10-20 22:00:00   Min.   :  8445784   Length:20391
##  1st Qu.:2020-10-26 01:20:00   1st Qu.:  8445784   Class :character
##  Median :2020-10-31 08:50:00   Median :116013753   Mode  :character
##  Mean   :2020-11-04 01:44:02   Mean   : 70823902
##  3rd Qu.:2020-11-13 07:20:00   3rd Qu.:124076095
##  Max.   :2020-11-19 07:55:00   Max.   :124076095
##   total_lmp_rt   forecast_load_mw      date                 time
##  Min.   : 7.29   Min.   : 2002    Min.   :2020-10-20   Min.   :    1
##  1st Qu.:15.77   1st Qu.: 3166    1st Qu.:2020-10-26   1st Qu.: 5098
##  Median :18.20   Median : 7904    Median :2020-10-31   Median :10196
##  Mean   :18.31   Mean   : 8039    Mean   :2020-11-03   Mean   :10196
##  3rd Qu.:20.88   3rd Qu.:11052    3rd Qu.:2020-11-13   3rd Qu.:15294
##  Max.   :29.84   Max.   :16959    Max.   :2020-11-19   Max.   :20391
```


Boxplot of Combined Cleaned Data
30
25
20
15
10



The LMP cleaned data has 0.8867695% of the raw LMP data

The LMP cleaned data has 0.9278759% of the raw LMP data

Separate the data into separate nodes and averaged values for the nodes

```
# combined separated by node
electric_combined_nodes <- combined_clean

# combined averages of the nodes
electric_combined_average <- combined_clean %>%
  group_by(datetime) %>%
  summarize(total_lmp_rt = mean(total_lmp_rt), forecast_load_mw = mean(forecast_load_mw))
```

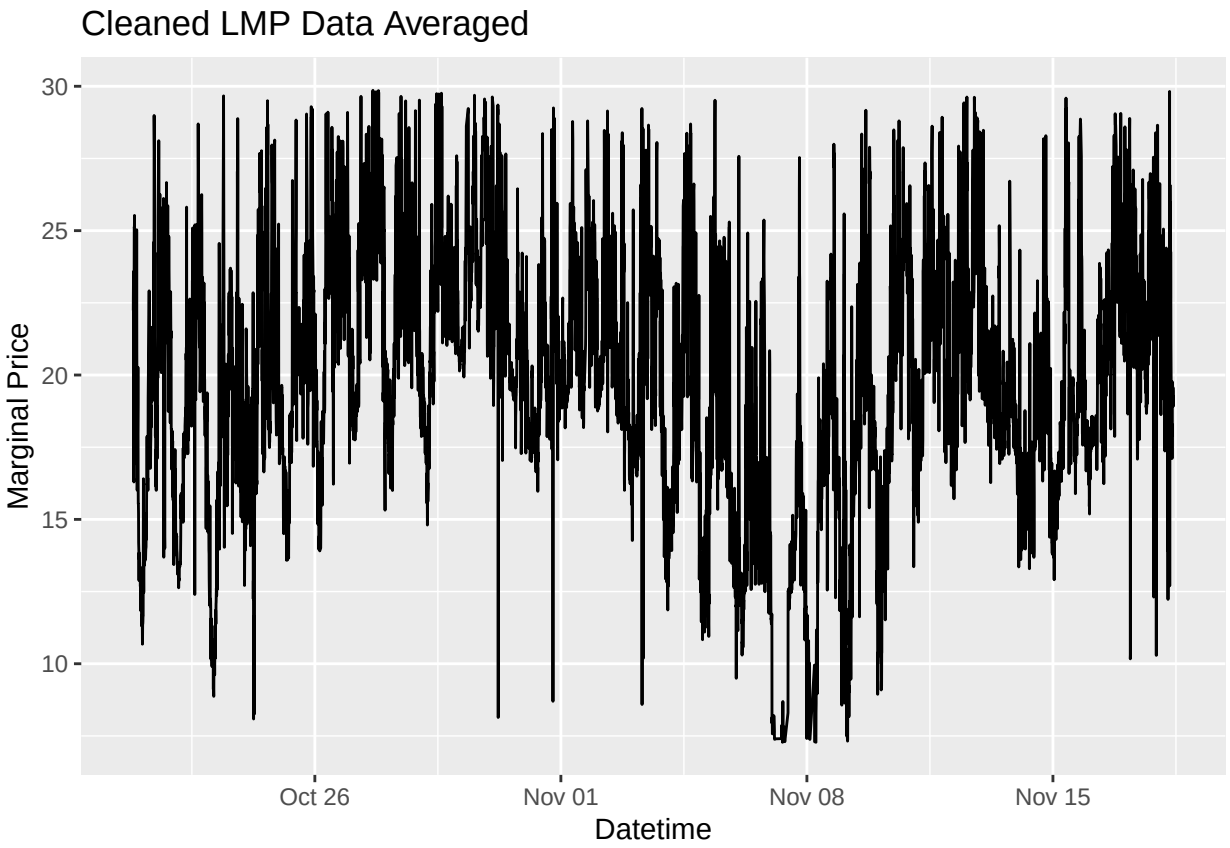
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

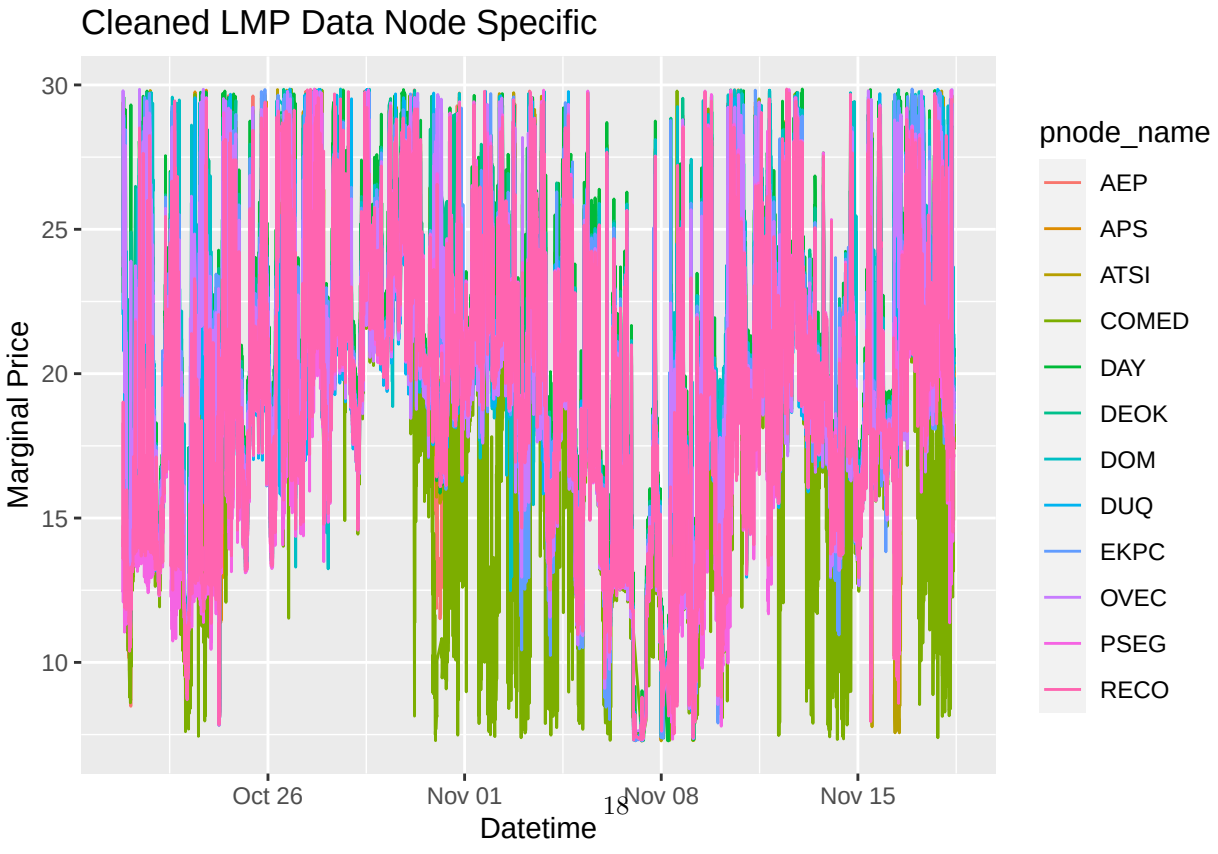
```
# separated by node
electric_nodes <- electric_clean

# averages of the nodes
electric_average <- electric_clean %>%
  group_by(datetime) %>%
  summarize(total_lmp_rt = mean(total_lmp_rt))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

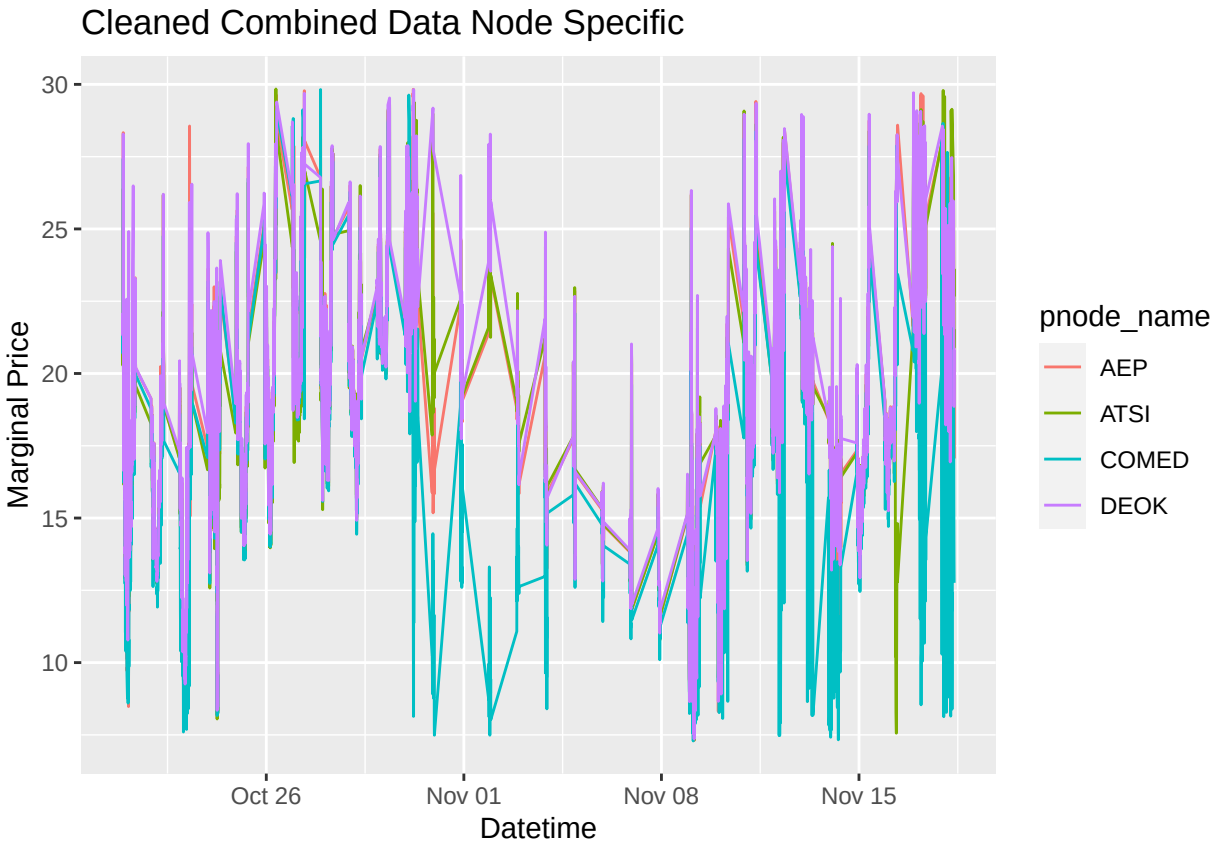
Visualize Cleaned Data

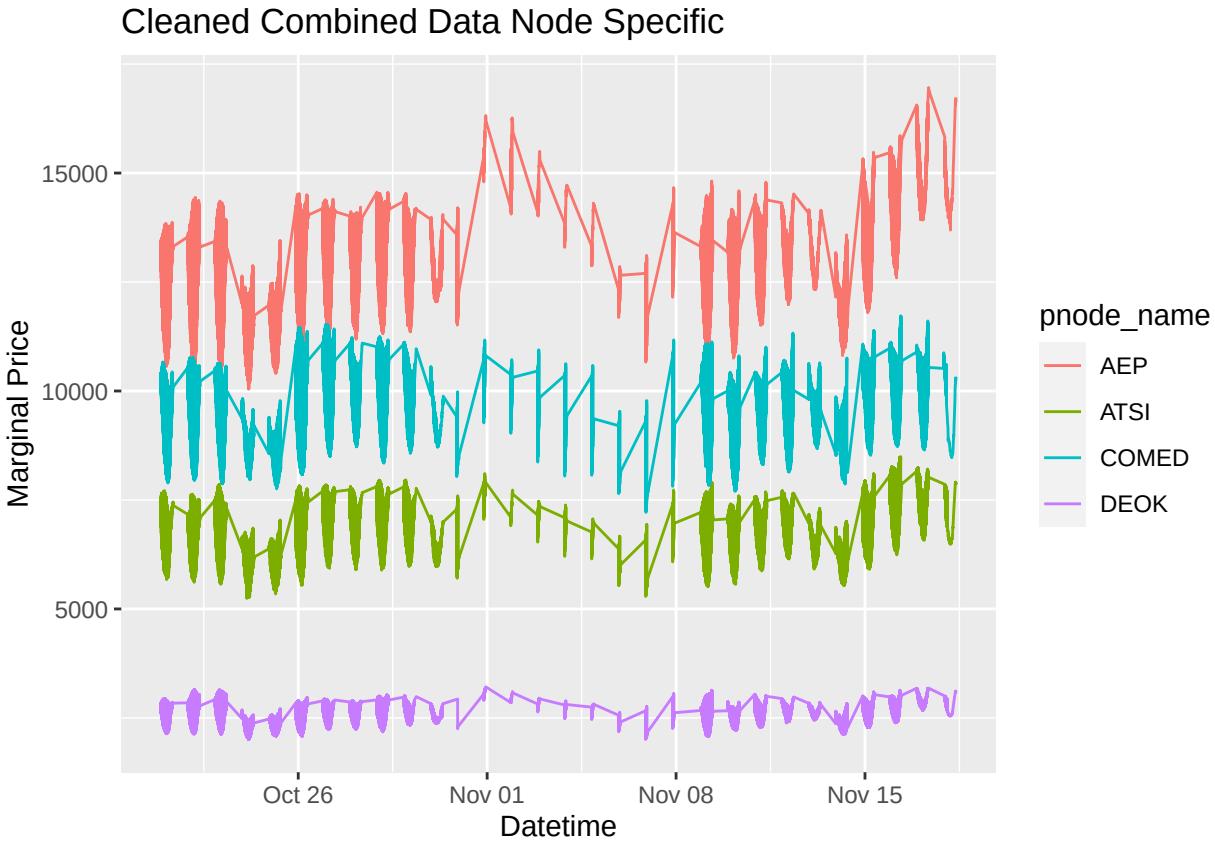
Plot the LMP Data

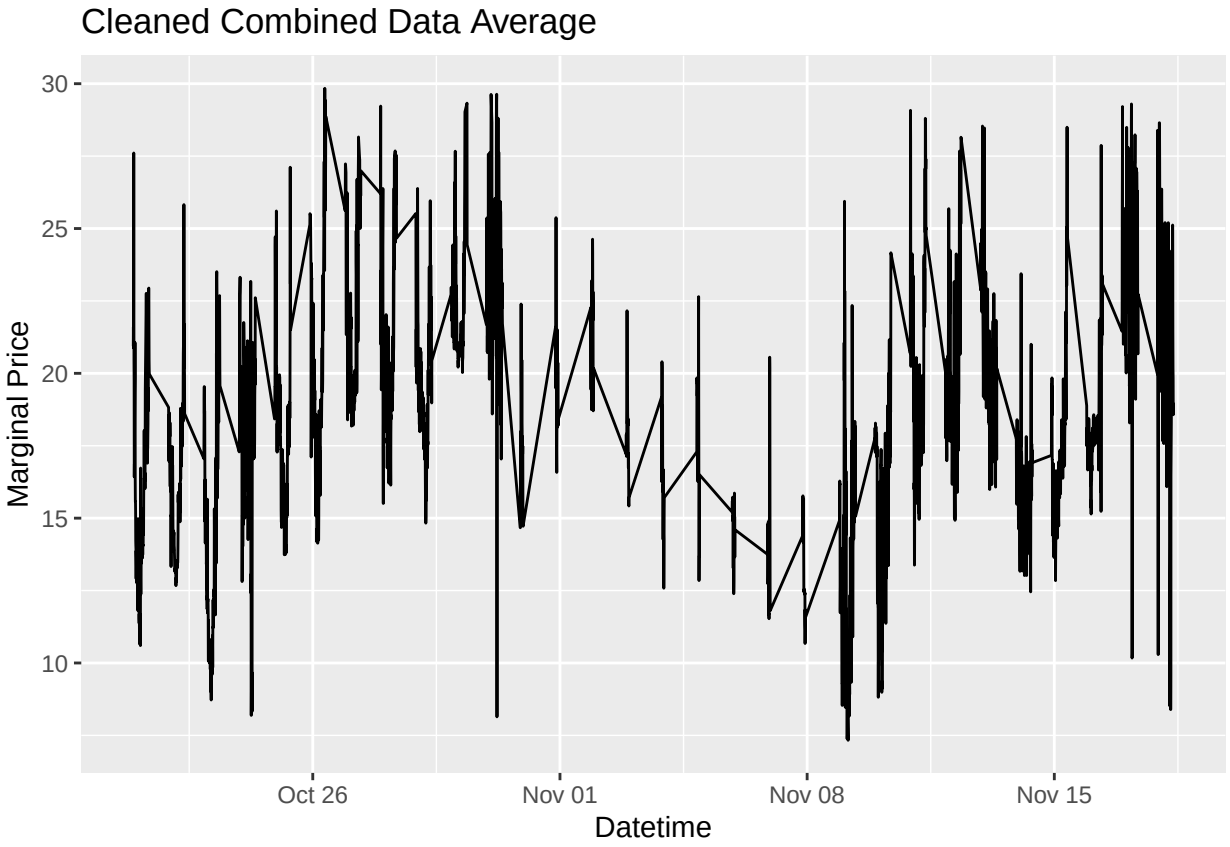


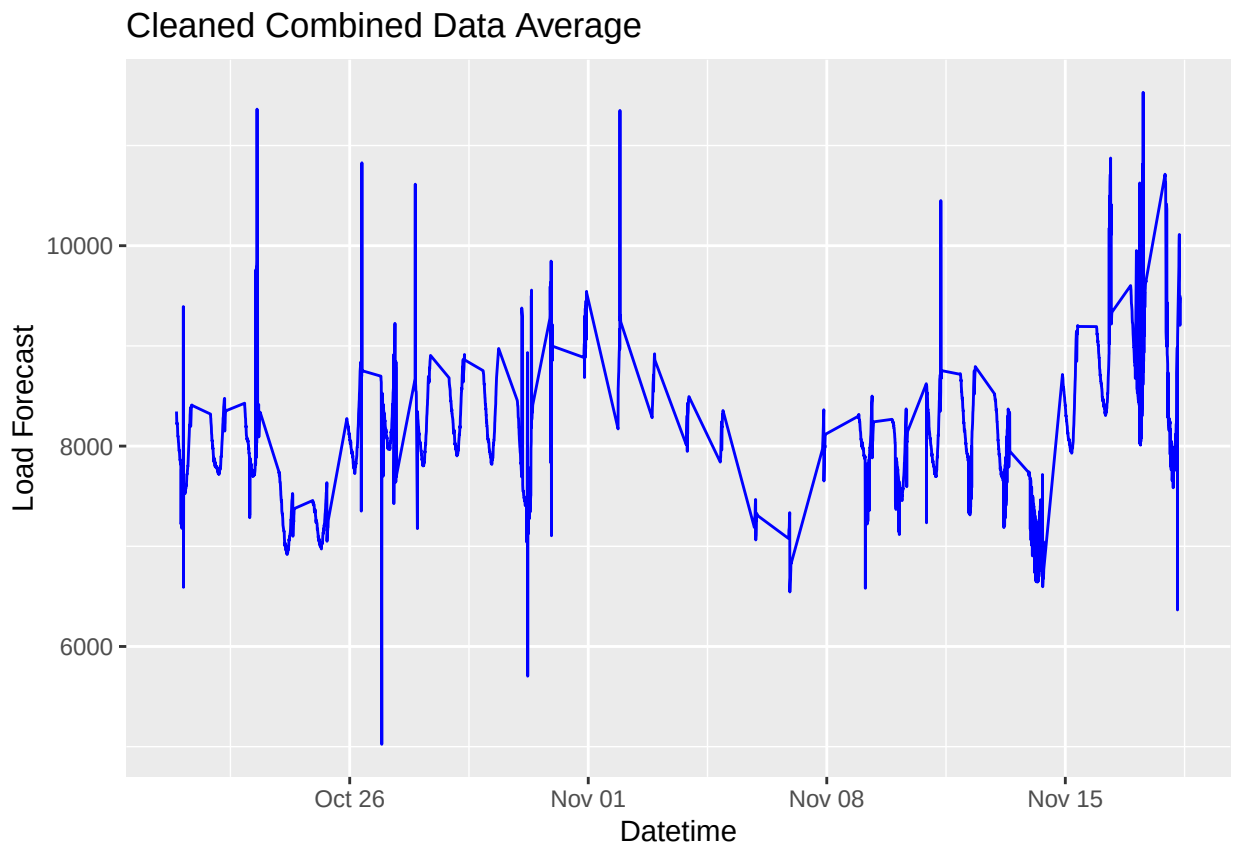


Plot the Combined Data






Cleaned Combined Data Average
Marginal Price
30
25
20
15
10
Oct 26
Nov 01
Nov 08
Nov 15
Datetime



Decide on averaged or node specific data for analysis

We will first do analysis on just the LMP data without the forecasts. This dataset is larger and has less holes in it.

The matching of the forecasts and the LMP data creates holes in the LMP data as the forecasts do not exists throughout the day

We will also start with the averaged data, not the node specific data

Create testing and training data

```
# the index to split the data on
separationIndex <- nrow(electric_average)-(60/5)

# split the data
electric_train <- electric_average[1:separationIndex-1,]
electric_test <- electric_average[separationIndex:nrow(electric_average),]
```

We have choosen to use one hour of testing data. This is obtained by taking 60min / 5min as 5 minutes is the frequency of the data.

This has been choosen as the application of forecasting prices for an electric vehicle will not involve long term predictions, only predictions that will impact the next hour or so of charging.

Check the training and testing data

The training data has 99.8368474% of the cleaned LMP averaged data The testing data has 0.1631526% of the cleaned LMP averaged data

Create Mostly Hand Made model

The first model created will not take into account the forecasts being provided by pjm. The data we will use first will the raw lmp data that has been cleaned to remove outliers. This data is more complete than when we combine the forecasts with this cleaned data set.

Remove trend

Create linear model and see statistical validity

```
lm <- lm(total_lmp_rt ~ datetime, electric_train)
```

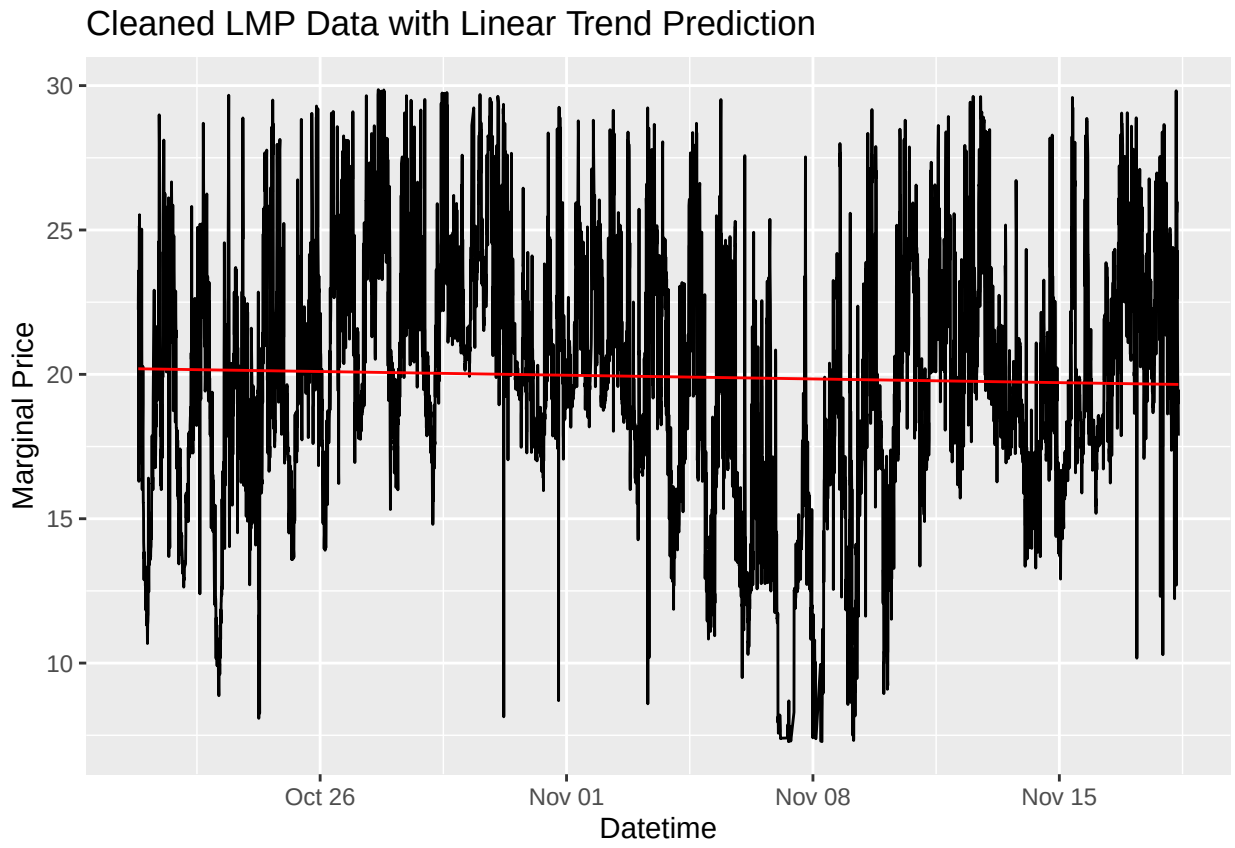
Check the model

```
summary(lm)
```

```
##
## Call:
## lm(formula = total_lmp_rt ~ datetime, data = electric_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.5741  -2.5627   0.0267   2.9426  10.1679
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.604e+02  1.007e+02   3.580 0.000346 ***
## datetime    -2.122e-07  6.275e-08  -3.382 0.000724 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.157 on 7953 degrees of freedom
## Multiple R-squared:  0.001436,   Adjusted R-squared:  0.00131
## F-statistic: 11.44 on 1 and 7953 DF,  p-value: 0.0007239
```

The linear trend model is significant so it will be considered for further analysis

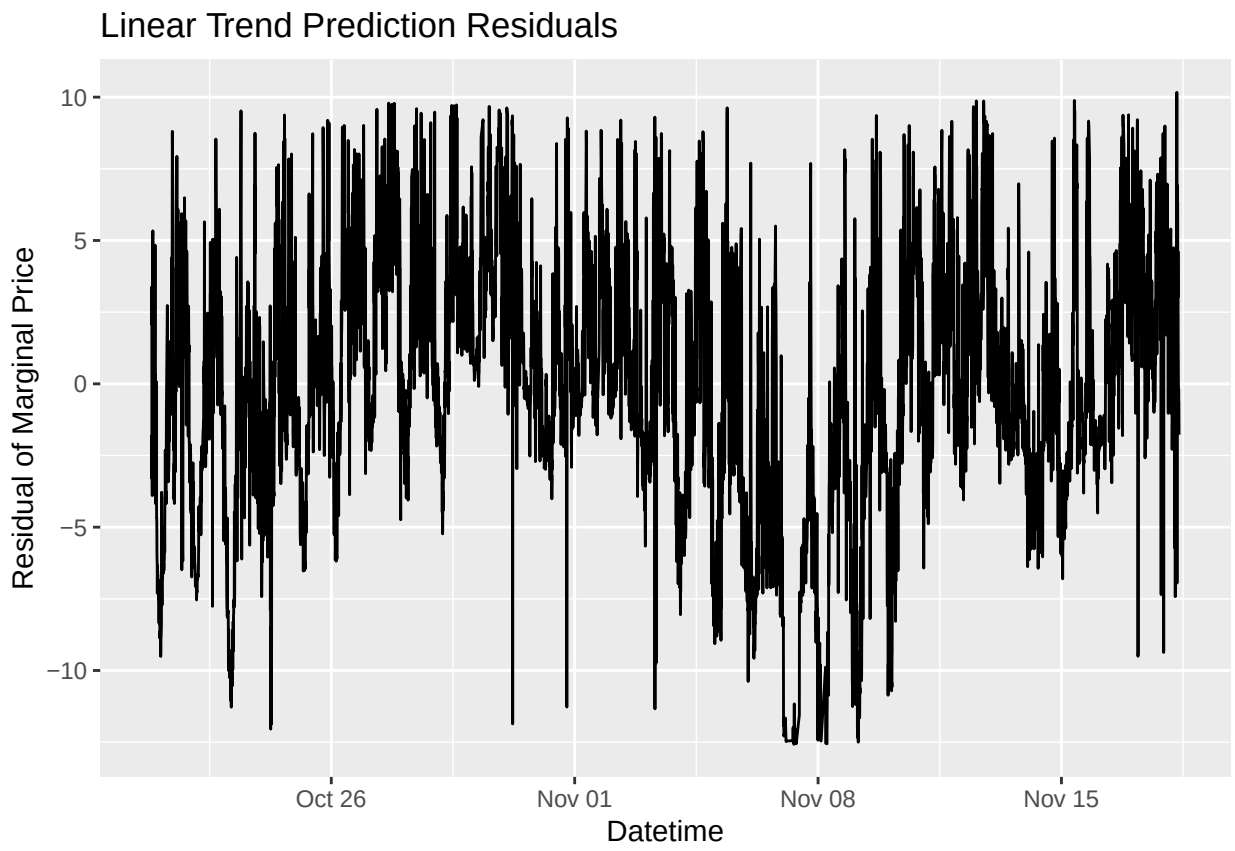
Plot the trendline



Use prediction in the future find residuals and save in dataframe

```
electric_train$lm_res <- lm$residuals
electric_train$lm_pred <- lm$fitted.values
```

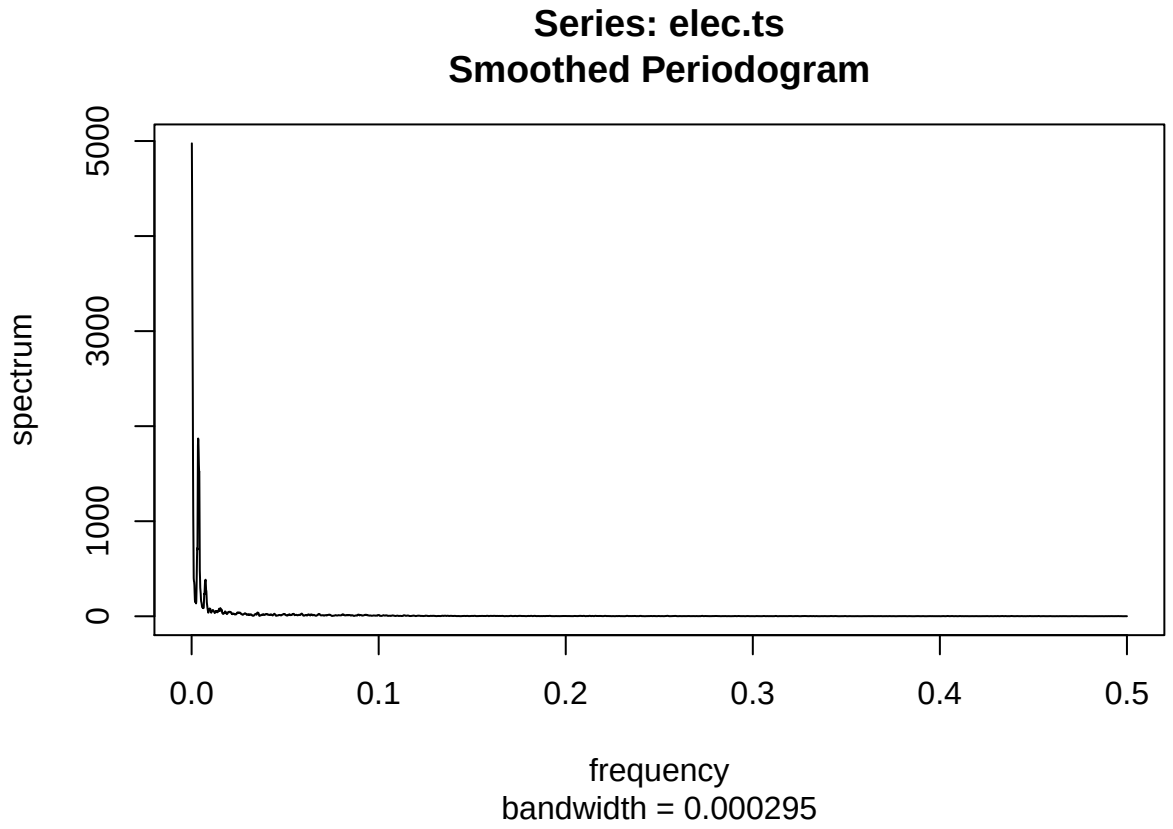
Plot the residuals



Remove sinusoidal movement

Find any sinusoidal movement

```
# create time series
elec.ts <- ts(electric_train$lm_res)

# find frequencies of high influence
pgram <- spec.pgram(elec.ts, spans=9, demean=T, log='no')
```




```
# sort the frequencies based on influence
sorted.spec <- sort(pgram$spec, decreasing=T, index.return=T)

# convert to periods
sorted.omegas <- pgram$freq[sorted.spec$ix]
sorted.Ts <- 1/pgram$freq[sorted.spec$ix]

# the cutoff for influential
pgram.cutoff <- 10

# the sampling rate per day
print('sampling rate')
```

```
## [1] "sampling rate"
```

```
nrow(electric_train)/as.numeric(max(electric_train$datetime)-min(electric_train$datetime))
```

```
## [1] 269.2174
```

```
# the top periods
print('top periods')
```

```
## [1] "top periods"
```

```
sorted.Ts[1:pgram.cutoff]
```

```
##  [1] 8000.0000 4000.0000 2666.6667 2000.0000 1600.0000  285.7143  275.8621
##  [8]  296.2963  266.6667  258.0645
```

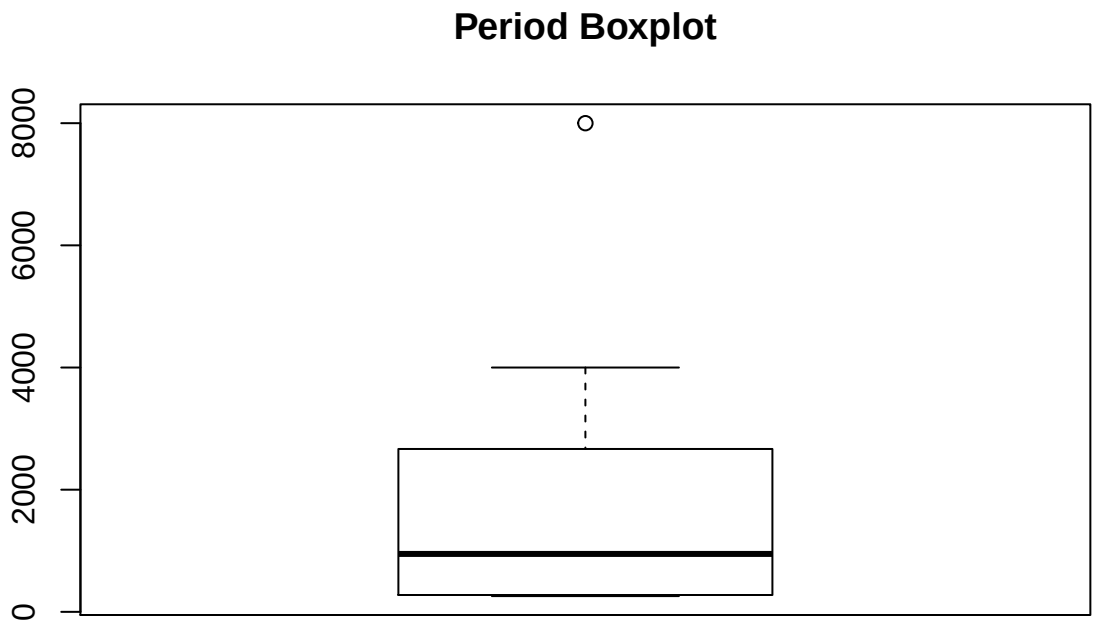
```
# top frequencies
## to double check that this makes sense based on periodogram
print('top frequencies')
```

```
## [1] "top frequencies"
```

```
sorted.omegas[1:pgram.cutoff]
```

```
##  [1] 0.000125 0.000250 0.000375 0.000500 0.000625 0.003500 0.003625
##  [8] 0.003375 0.003750 0.003875
```

```
# visual
pgram.box <- boxplot(sorted.Ts[1:pgram.cutoff], main="Period Boxplot")
```



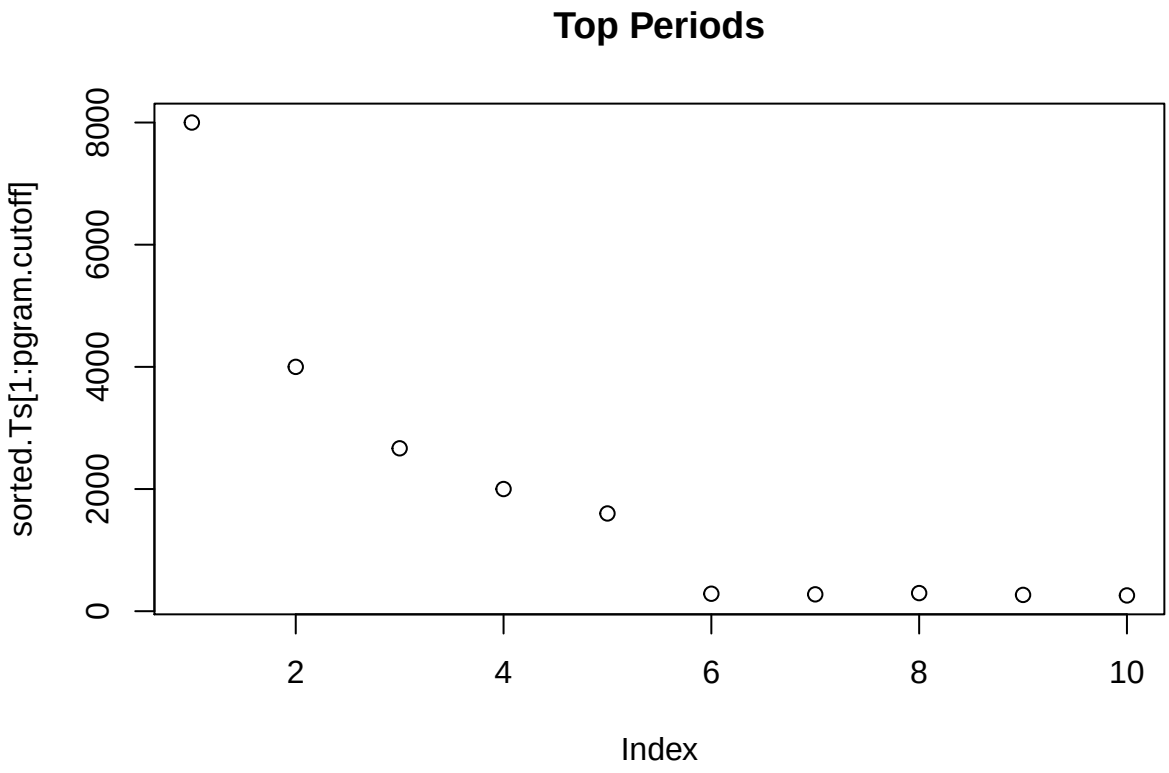

```
# the average influential period
print('mean of top periods')
```

```
## [1] "mean of top periods"
```

```
pgram.box.mean <- pgram.box$stats[3]
print(pgram.box.mean)
```

```
## [1] 948.1481
```

```
# plot top periods
plot(sorted.Ts[1:pgram.cutoff], main = "Top Periods")
```

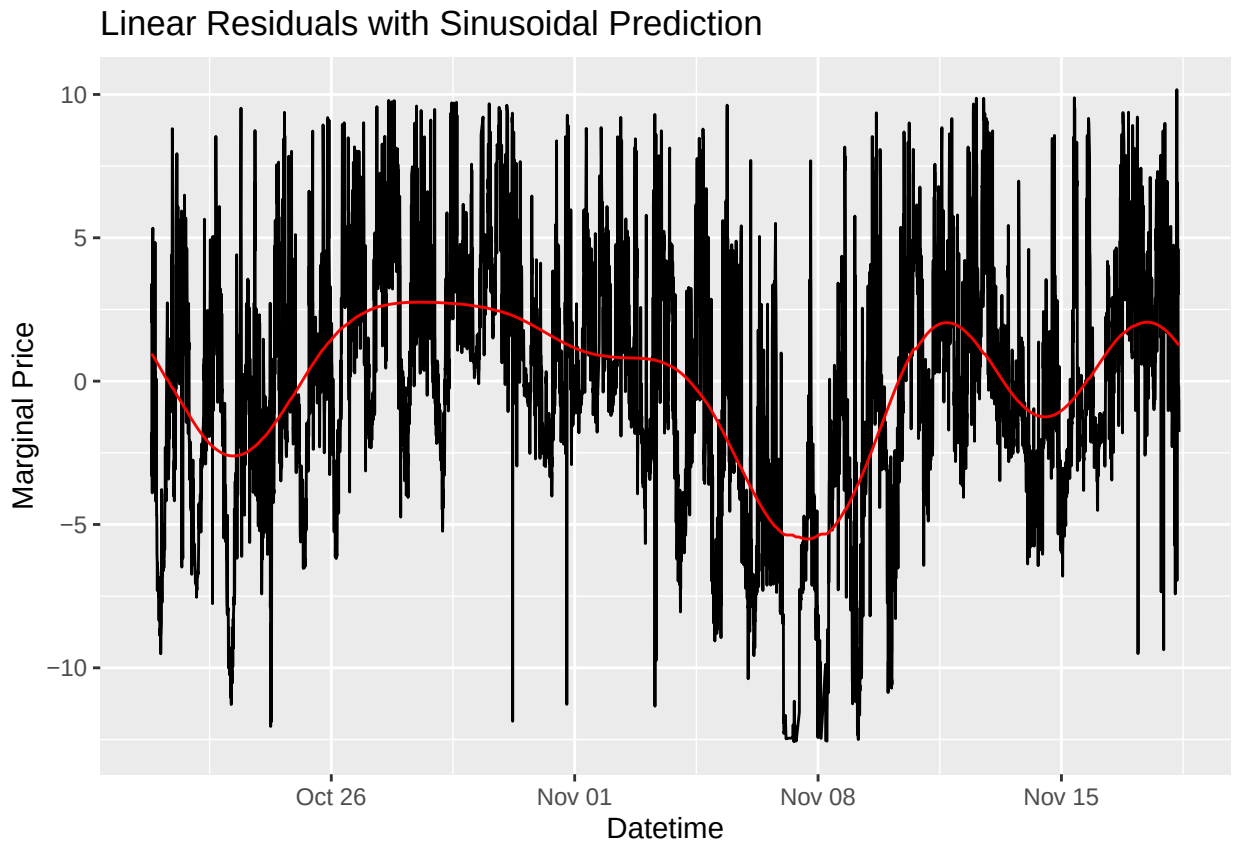



```
### create a model for the seasonality
# assign potential periods to variables
p1 <- sorted.Ts[1]
p2 <- sorted.Ts[2]
p3 <- sorted.Ts[3]
p4 <- sorted.Ts[4]
p5 <- sorted.Ts[5]
p6 <- sorted.Ts[6]

# create time variable
time<-c(1:length(elec.ts))

# model
sin_mov <- lm(elec.ts ~
                  sin(2*pi*time/p1)
                  + cos(2*pi*time/p1)
```

```
                  + sin(2*pi*time/p2)
                  + cos(2*pi*time/p2)
                  + sin(2*pi*time/p3)
                  + cos(2*pi*time/p3)
                  + sin(2*pi*time/p4)
                  + cos(2*pi*time/p4)
                  + sin(2*pi*time/p5)
                  + cos(2*pi*time/p5)
                  )
summary(sin_mov)
```

```
##
## Call:
## lm(formula = elec.ts ~ sin(2 * pi * time/p1) + cos(2 * pi * time/p1) +
##     sin(2 * pi * time/p2) + cos(2 * pi * time/p2) + sin(2 * pi *
##     time/p3) + cos(2 * pi * time/p3) + sin(2 * pi * time/p4) +
##     cos(2 * pi * time/p4) + sin(2 * pi * time/p5) + cos(2 * pi *
##     time/p5))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.1506  -2.3257  -0.0943   2.2848  13.1815
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           0.006206   0.039426   0.157   0.8749
## sin(2 * pi * time/p1)  1.444549   0.055591  25.985  < 2e-16 ***
## cos(2 * pi * time/p1)  0.339198   0.055923   6.065 1.38e-09 ***
## sin(2 * pi * time/p2) -1.870549   0.055591 -33.648  < 2e-16 ***
## cos(2 * pi * time/p2) -0.783087   0.055922 -14.003  < 2e-16 ***
## sin(2 * pi * time/p3)  0.107889   0.055592   1.941   0.0523 .
## cos(2 * pi * time/p3) -0.252112   0.055922  -4.508 6.63e-06 ***
## sin(2 * pi * time/p4) -0.456966   0.055593  -8.220 2.36e-16 ***
## cos(2 * pi * time/p4)  1.539175   0.055921  27.524  < 2e-16 ***
## sin(2 * pi * time/p5) -0.918971   0.055594 -16.530  < 2e-16 ***
## cos(2 * pi * time/p5)  0.113665   0.055920   2.033   0.0421 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.516 on 7944 degrees of freedom
## Multiple R-squared:  0.2855, Adjusted R-squared:  0.2846
## F-statistic: 317.5 on 10 and 7944 DF,  p-value: < 2.2e-16
```

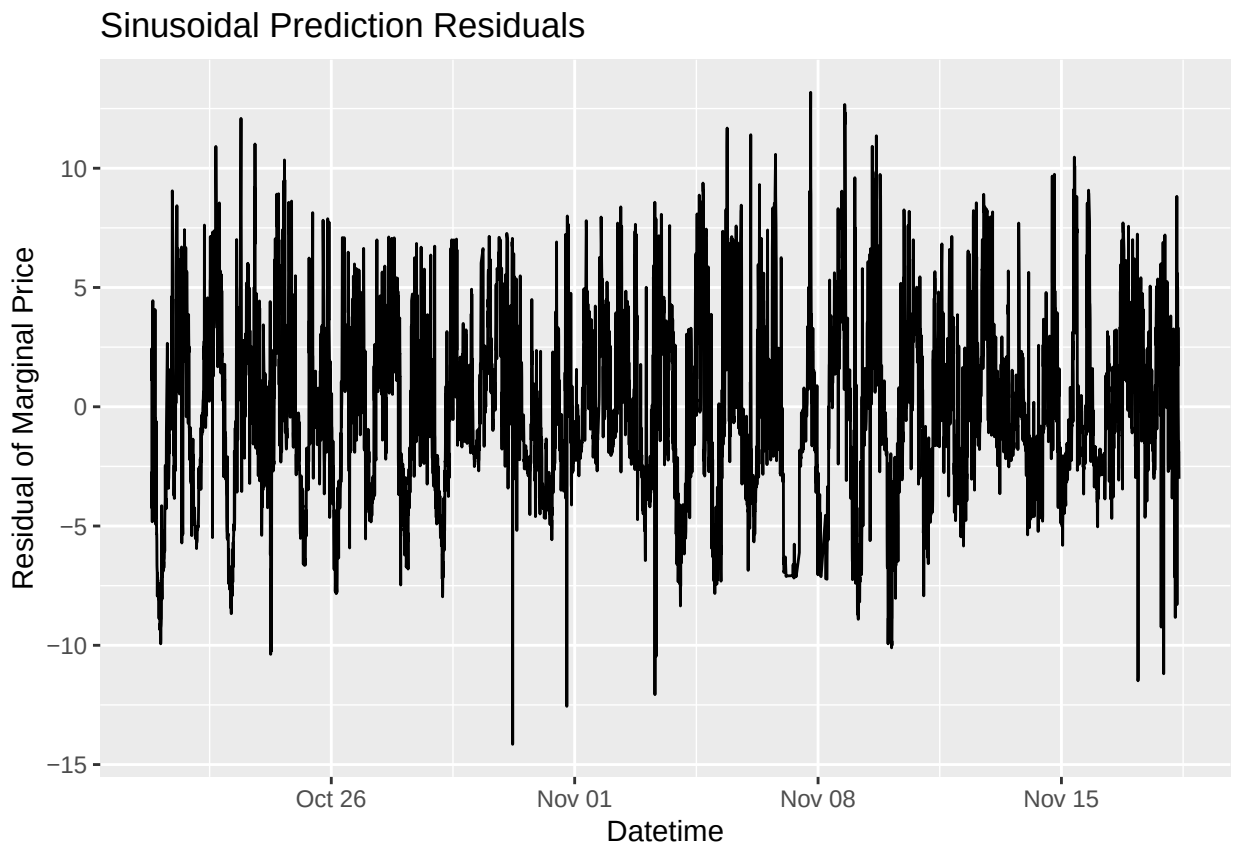
Plot sinusoidal movement



Use model to store residuals

```
electric_train$sin_res <- sin_mov$residuals
electric_train$sin_pred <- sin_mov$fitted.values
```

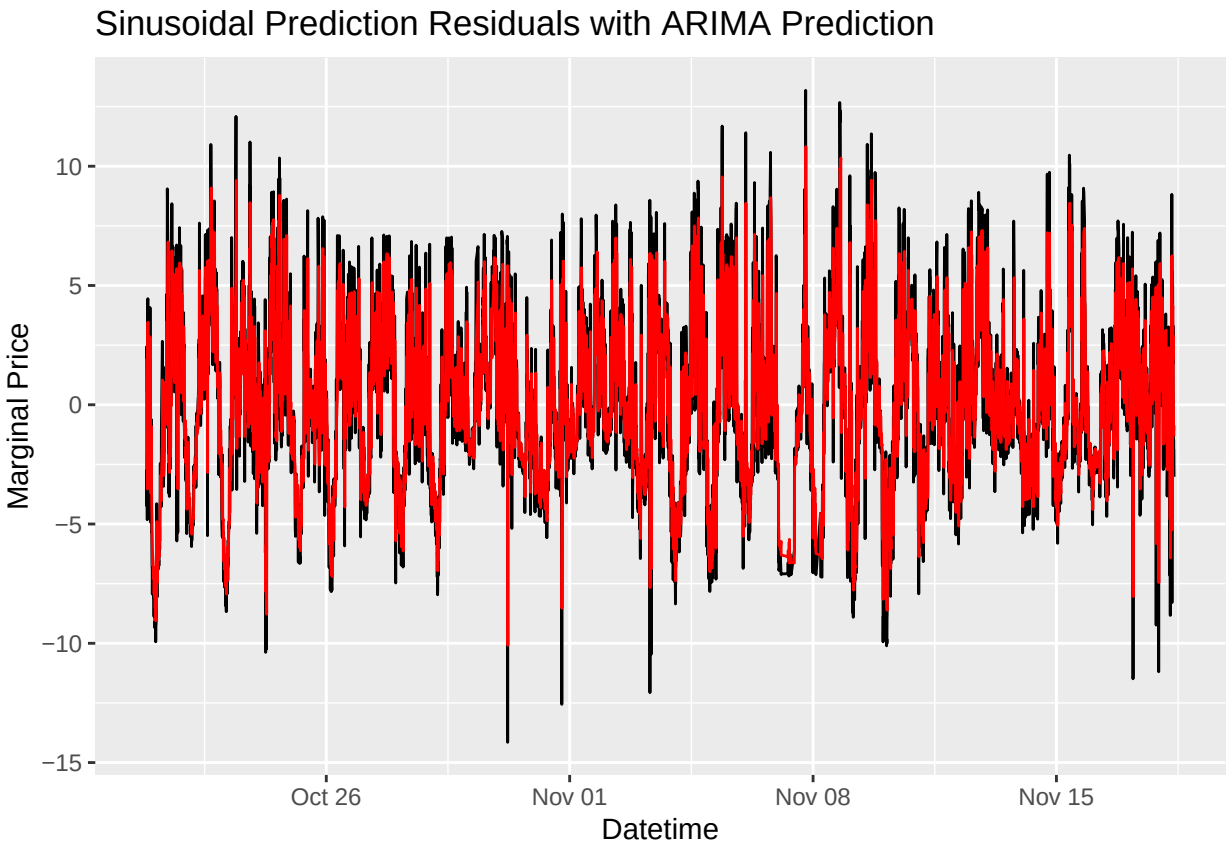
Plot residuals



Model Residuals

```
auto <- auto.arima(electric_train$sin_res, approximation = FALSE)
summary(auto)
```

```
## Series: electric_train$sin_res
## ARIMA(2,0,2) with zero mean
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.6234  -0.6309  -0.9262  0.0491
## s.e.  0.0267   0.0258   0.0293  0.0184
##
## sigma^2 estimated as 3.01:  log likelihood=-15668.8
## AIC=31347.6   AICc=31347.61   BIC=31382.51
##
## Training set error measures:
##                        ME     RMSE      MAE      MPE     MAPE     MASE
## Training set -0.0005331767 1.734366 1.130085 4.505082 246.6963 1.016556
##                      ACF1
## Training set 6.042803e-05
```



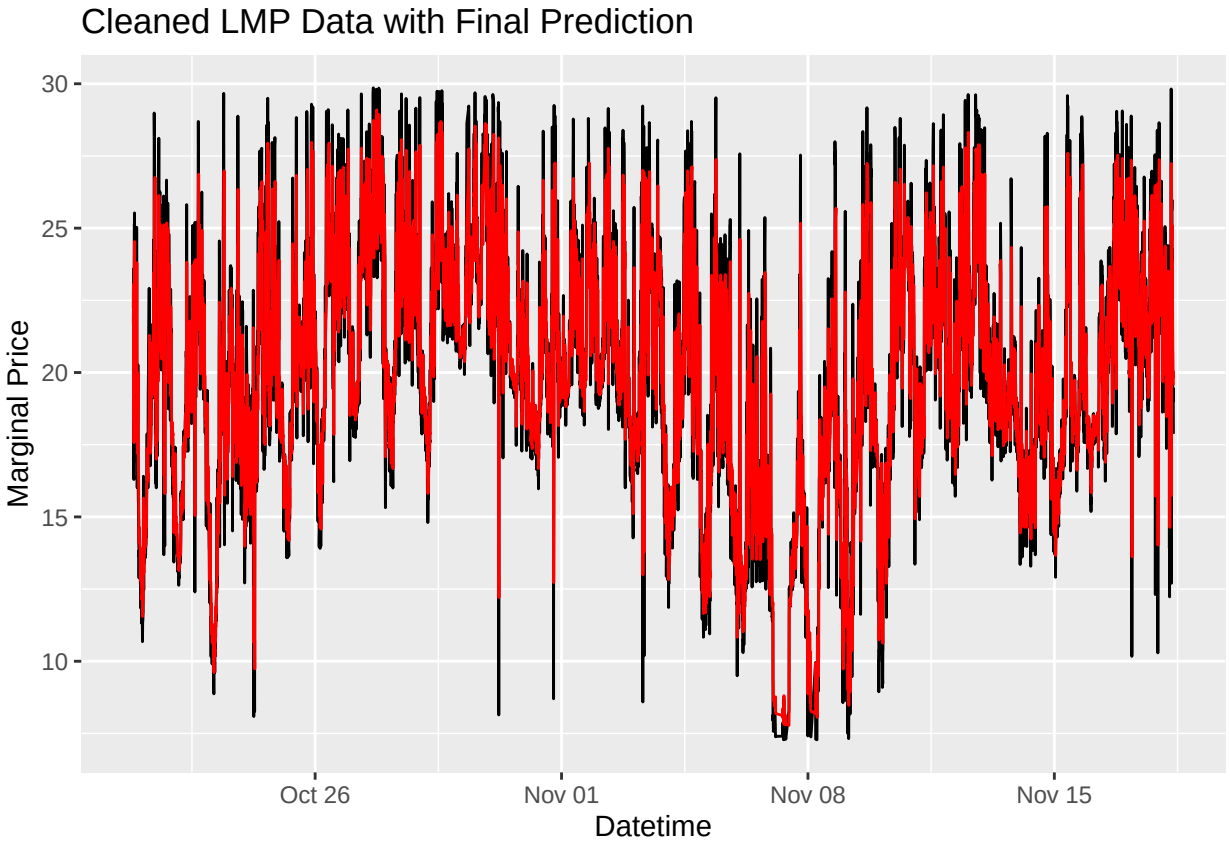
Save residual prediction

```
electric_train$ar_pred <- auto$fitted
```

Combine all models

```
# store the values
electric_train$model_final <- electric_train$ar_pred + electric_train$lm_pred + electric_train$sin_pred
```

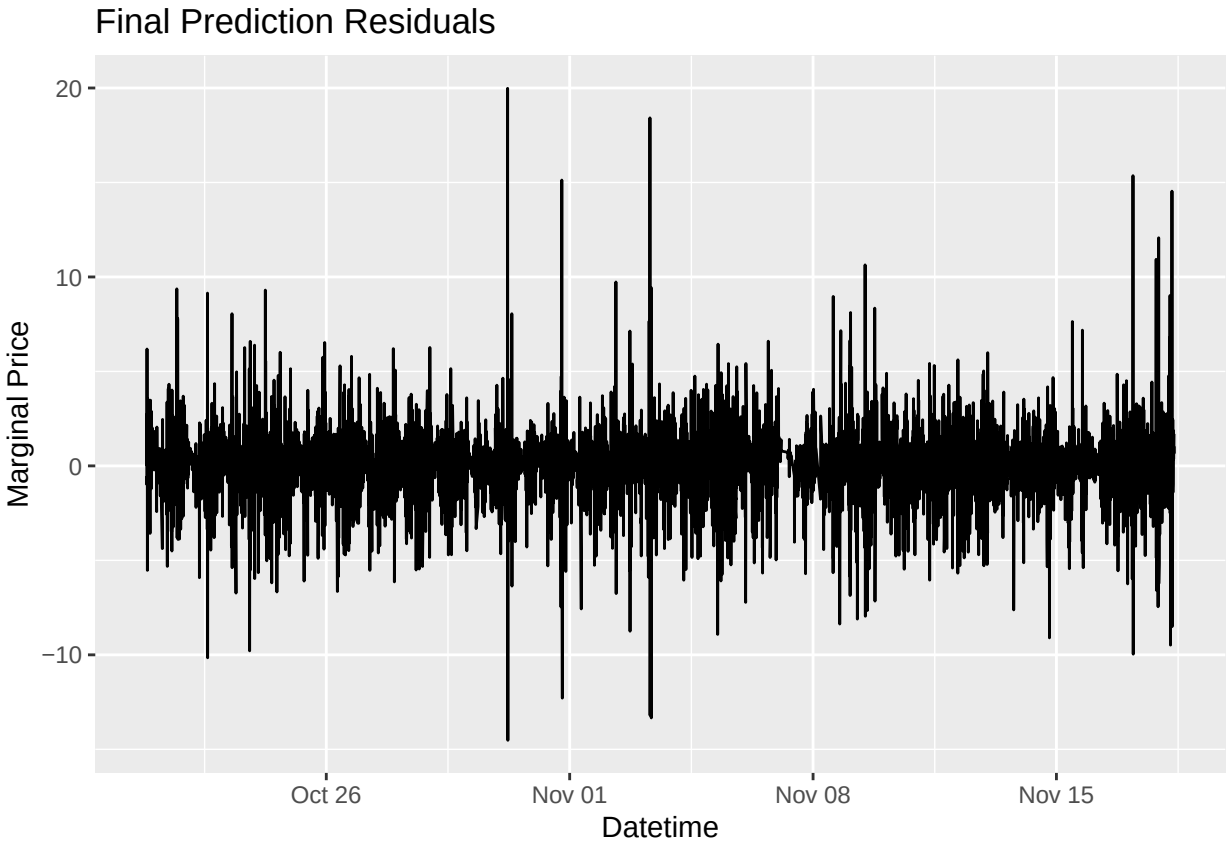
Plot the final model predictions

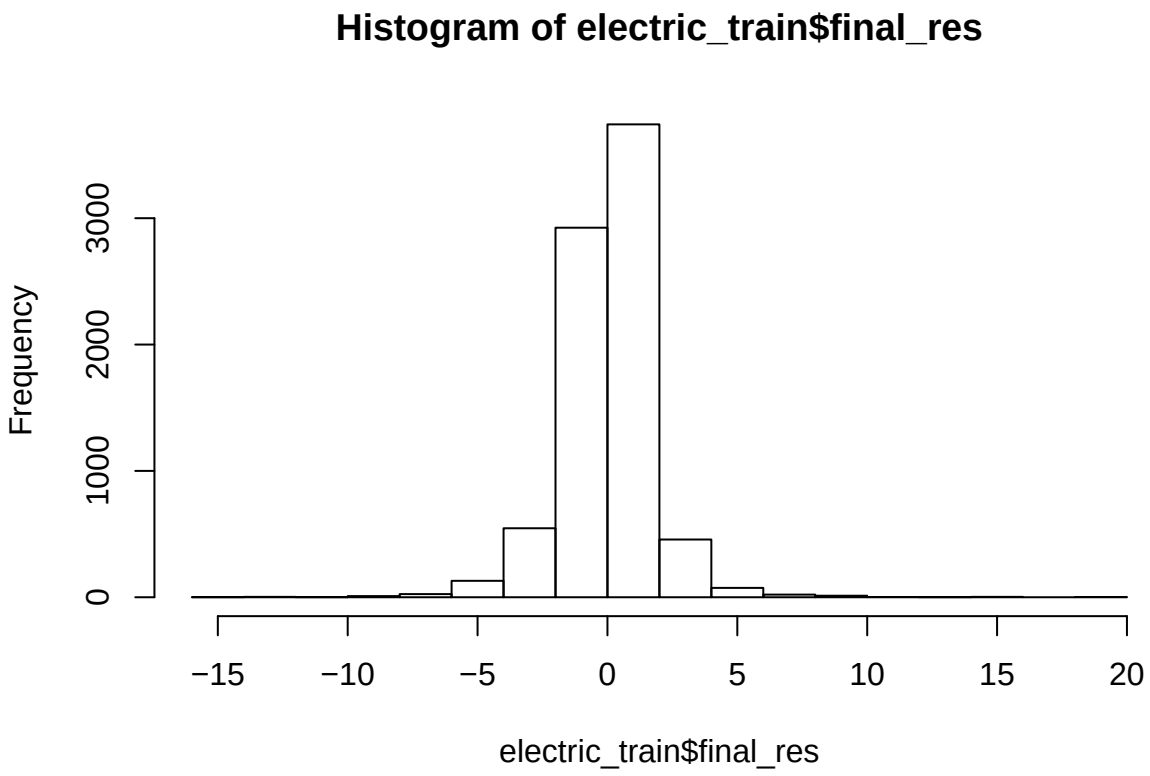


Model validity and statistics

Plot the residuals

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```

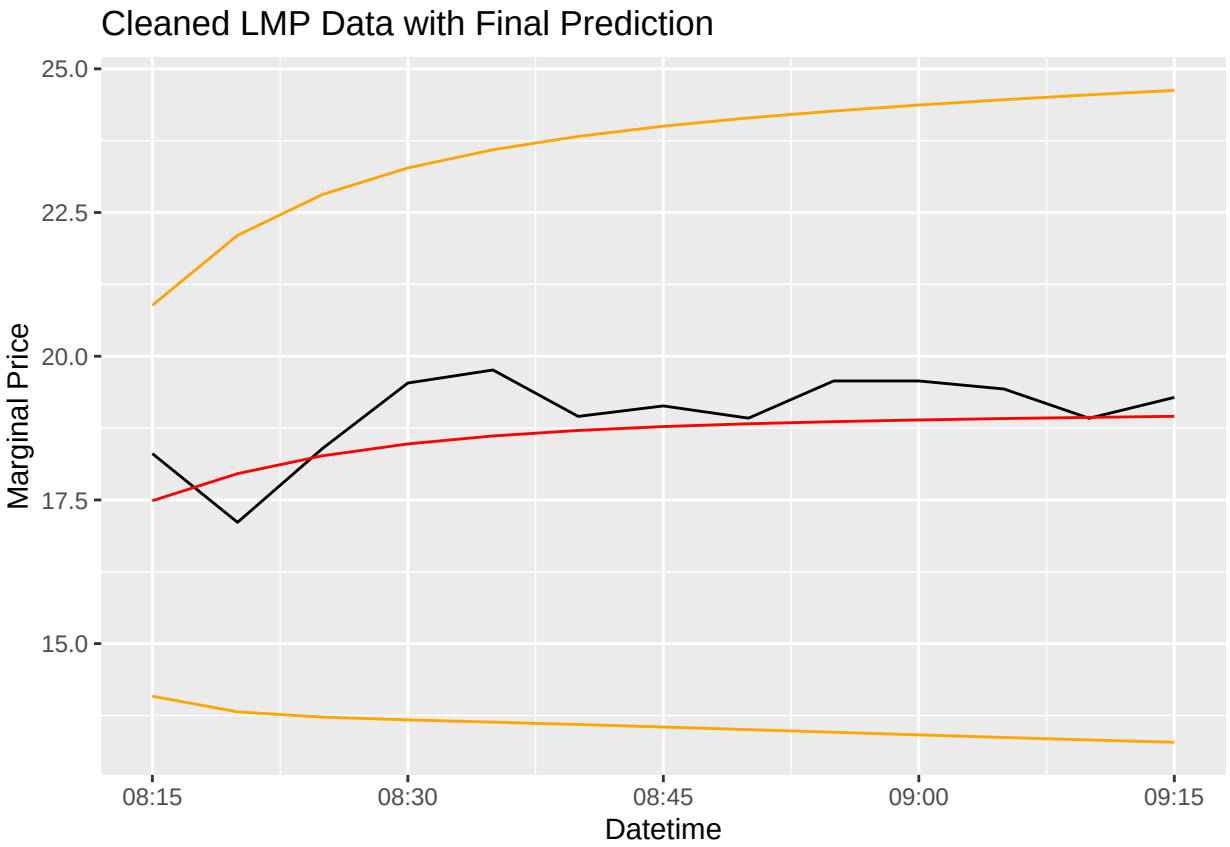

Final Prediction Residuals
20
10
0
−10
Marginal Price
Oct 26
Nov 01
Nov 08
Nov 15
Datetime



Forecast values

```
# create dataframe for forecasted values
future <- electric_test %>%
  select(datetime)

# create predictions
E_Y.pred.lmp <- predict(lm, newdata=future)
e_t.pred.lmp <- forecast(auto, h=nrow(future))
lmp.forecast <- E_Y.pred.lmp + e_t.pred.lmp$mean
```

Plot Forecasted Values



From the graph we can see that our model completes the test within our confidence interval 90%

Explore using nodes as parameters

Create Training and testing data

The same technique is used to split the data into training and testing, using one hour as the testing data ## Create the Model

```
# Use as factor
electric_nodes_train$pnode_name <- as.factor(electric_nodes_train$pnode_name)
# Create the Model
lm_nodes <- lm(total_lmp_rt ~ pnode_name + datetime, data=electric_nodes_train)
```

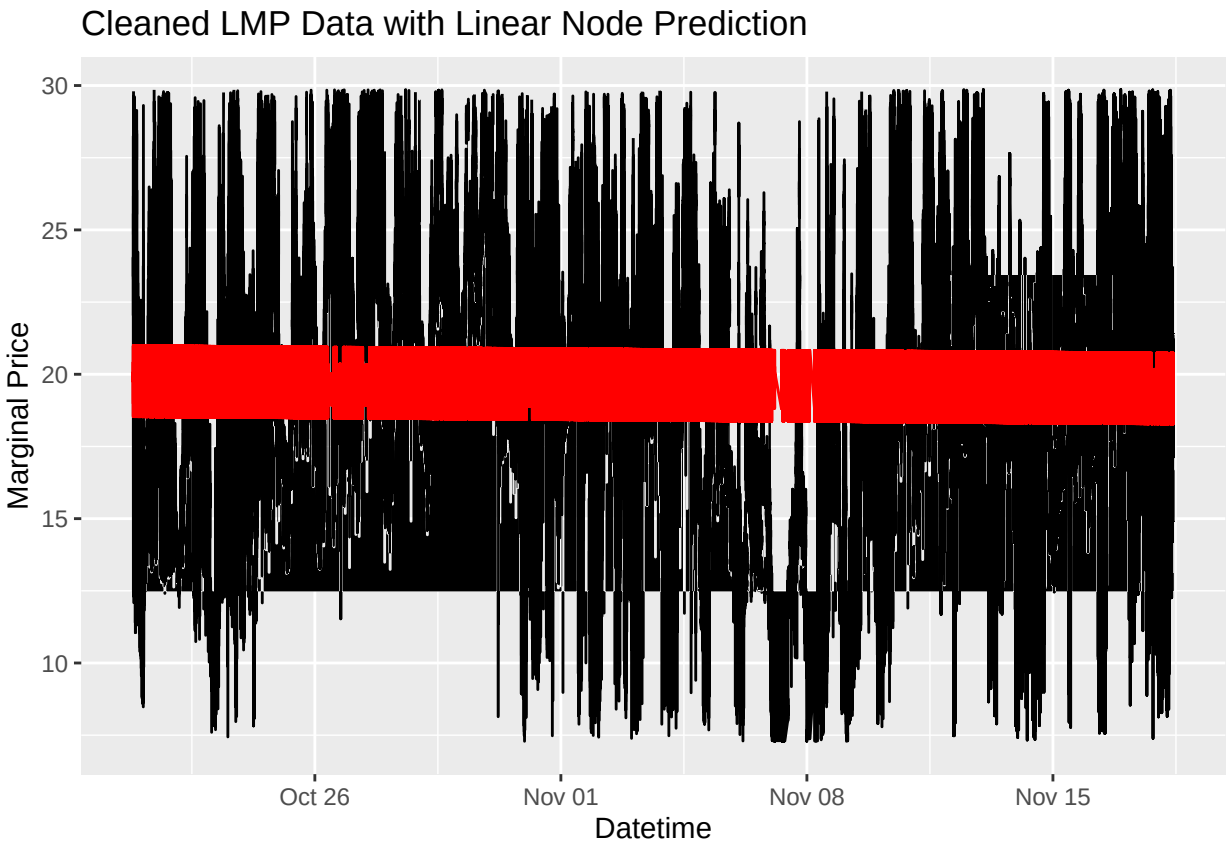
Check the Model

```
##
## Call:
## lm(formula = total_lmp_rt ~ pnode_name + datetime, data = electric_nodes_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5482  -2.7067   0.0568   2.9750  11.4250
```

```
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      1.787e+02  3.109e+01   5.747 9.14e-09 ***
## pnode_nameAPS    9.327e-02  6.994e-02   1.334 0.182364
## pnode_nameATSI   4.941e-02  7.009e-02   0.705 0.480912
## pnode_nameCOMED -1.451e+00  7.099e-02 -20.443  < 2e-16 ***
## pnode_nameDAY    1.001e+00  7.062e-02  14.176  < 2e-16 ***
## pnode_nameDEOK   4.384e-01  7.086e-02   6.187 6.16e-10 ***
## pnode_nameDOM    3.904e-01  7.034e-02   5.550 2.86e-08 ***
## pnode_nameDUQ   -2.546e-01  6.965e-02  -3.655 0.000257 ***
## pnode_nameEKPC   2.618e-01  7.023e-02   3.728 0.000193 ***
## pnode_nameOVEC   8.998e-02  7.017e-02   1.282 0.199742
## pnode_namePSEG  -1.007e+00  6.962e-02 -14.468  < 2e-16 ***
## pnode_nameRECO  -9.206e-01  6.967e-02 -13.213  < 2e-16 ***
## datetime        -9.897e-08  1.938e-08  -5.108 3.26e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.288 on 89653 degrees of freedom
## Multiple R-squared:  0.02351,    Adjusted R-squared:  0.02338
## F-statistic: 179.9 on 12 and 89653 DF,  p-value: < 2.2e-16
```

We can see that since the model is signficant and that there are pnodes that are signficiant, this shows that adding pnodes as a predictor will help us in creating our model

Plot the linear model



Explore Modeling with the forecasts

Create Training and Testing Data

```
# the index to split the data on
separationIndex <- nrow(electric_combined_average)-(60/5)

# split the data
electric_combined_train <- electric_combined_average[1:separationIndex-1,]
electric_combined_test <- electric_combined_average[separationIndex:nrow(electric_combined_average),]
```

Create the Model

```
# Create the Model
lm_forecast <- lm(total_lmp_rt ~ forecast_load_mw + datetime, data=electric_combined_train)
```

Check the Model

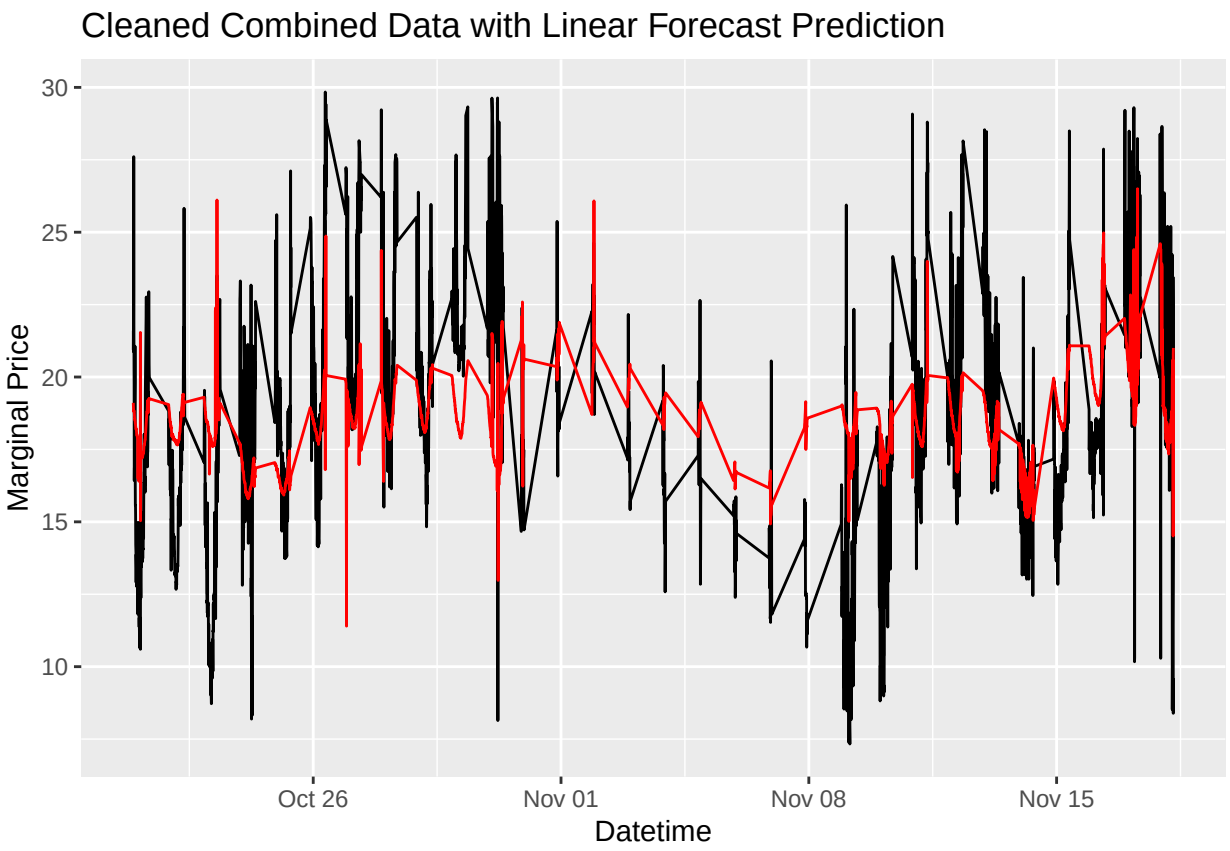
```
##
```

```
## Call:
## lm(formula = total_lmp_rt ~ forecast_load_mw + datetime, data = electric_combined_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.0819  -2.3051  -0.0344   2.3415  15.5449
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       9.615e+01  1.370e+02   0.702    0.483
## forecast_load_mw  2.340e-03  1.125e-04  20.810   <2e-16 ***
## datetime         -6.018e-08  8.550e-08  -0.704    0.482
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.735 on 2829 degrees of freedom
## Multiple R-squared:  0.1383, Adjusted R-squared:  0.1377
## F-statistic:   227 on 2 and 2829 DF,  p-value: < 2.2e-16
```

When adding forecasts, the datetime is no longer a significant parameter, this is likely due to the holes in the dataset

We will remove datatime from this linear model

Visualize the Model



Explore modeling with both node and forecast

Create testing and training data

The same technique is used to separate training and testing data, one hour is used as testing data.

Create Model

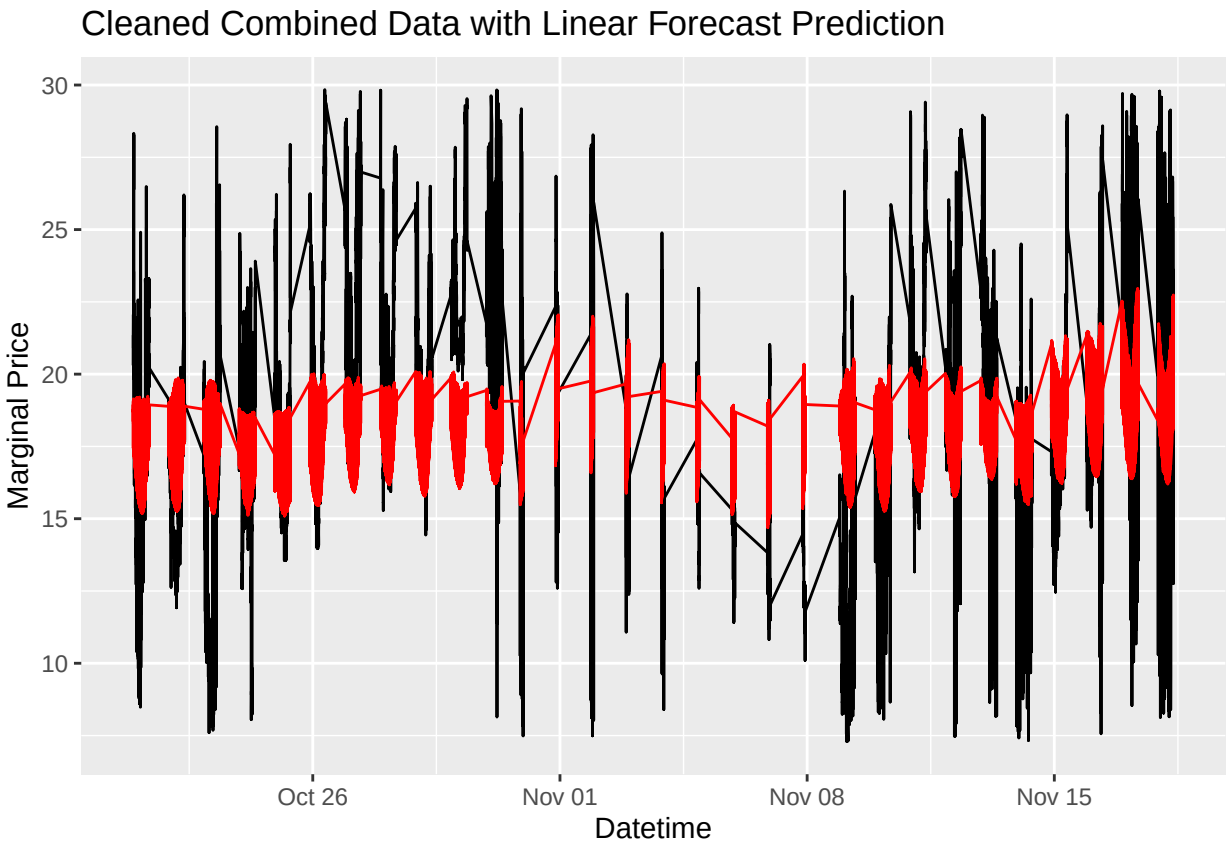
```
lm_node_forecast <- lm(total_lmp_rt ~ forecast_load_mw + pnode_name + datetime, data=electric_combined_
```

Check Model

```
##
## Call:
## lm(formula = total_lmp_rt ~ forecast_load_mw + pnode_name + datetime,
##     data = electric_combined_nodes_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.8512  -2.3738  -0.0579   2.5731  12.9628
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -2.467e+02  5.357e+01  -4.605 4.15e-06 ***
## forecast_load_mw  1.084e-03  3.067e-05  35.347  < 2e-16 ***
## pnode_nameATSI    6.636e+00  2.067e-01  32.100  < 2e-16 ***
## pnode_nameCOMED   2.429e+00  1.310e-01  18.538  < 2e-16 ***
## pnode_nameDEOK    1.166e+01  3.296e-01  35.359  < 2e-16 ***
## datetime          1.565e-07  3.344e-08   4.680 2.88e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.955 on 20372 degrees of freedom
## Multiple R-squared:  0.08329,    Adjusted R-squared:  0.08307
## F-statistic: 370.2 on 5 and 20372 DF,  p-value: < 2.2e-16
```

In this model, all parameters are significant

Plot Model



Choose the best linear model to use on our final model

```
AIC(lm)
```

```
## [1] 45248.23
```

```
AIC(lm_nodes)
```

```
## [1] 515531
```

```
AIC(lm_forecast)
```

```
## [1] 15503.99
```

```
AIC(lm_node_forecast)
```

```
## [1] 113879.2
```

The linear model with the smallest AIC is the model that only uses the forecasted values

While this would tell us we should use the forecasted model, we are going to continue with our original model of just datetime, as the data is much nicer and easier to work with.

We recommend the other models be explored in more detail in the future.

Simulation

Here we are going to see if we can make a profit off of our model. We will use the following logic to design our simulation:

We will simulate time passing by iterating through our testing data (five minute intervals)

Each five minute interval the logic will look at the current price of electricity and the forecasted price of the next five minutes

Electricity will be bought or sold according to the relation between the current and next forecasted price

We will keep track of the money and electricity gained and lost

```
# create the simulation dataframe
sim <- electric_test %>%
  mutate(forecast = lmp.forecast)

# Parameters to keep track of
no_model_cash <- 0
no_model_units <- 0
model_cash <- 0
model_units <- 0

# iterate through the simulation
for (d in sim$id[1:nrow(sim)-1]){

  # the values we are interested in
  current_lmp <- subset(sim, id==d)$total_lmp_rt
  forecast_next_lmp <- subset(sim, id==d+1)$forecast

  # pay for current electricity
  if (forecast_next_lmp > current_lmp){
    model_cash <- model_cash - current_lmp
    model_units <- model_units + 1
  }
  # sell back electricity
  else{
    model_cash <- model_cash + current_lmp
    model_units <- model_units - 1
  }

  # baseline for always buying electricity
  no_model_cash <- no_model_cash - current_lmp
  no_model_units <- no_model_units + 1
}
```

```
## Warning: Unknown or uninitialised column: `id`.
```

Our baseline car owner gained $0 for 0 units of electricity Our model using car owner gained $0 for 0 units of electricity

Our simulation above works on a very simple rule, only looking five minutes ahead: if the forecasted price in the next five minutes is higher than it is now, buy electricity, if the price is not higher, sell the electricity you currenlty have

This is far from a perfect analysis as there is no goal built into the simulation. However it is believed that this shows the proof of concept for the model based electricity forecasting to buy and sell electricity.

Conclusion

Model Valilidity

Our final model using the strictly linear trend model, sinusoidal movement, and arima gives us a fairly accurate model to forecast off of. We recieved a MSE score of 0.4139235 (how many dollars our model is off on average squared).

Next Steps

To extend this initial research it is advised to:

A person that has experience trading equities should be added onto the project. They will help develop an algorithm for when to buy and sell the electricity.

Look closer into techniques to utilize the forecasted load on the electrical system. This may include aquiring a more complete dataset from PJM. Also, multiple forecasts being taken into account, possible in the form of bayesian forecasting theory should be utilized.

A goal oriented simulation should be contucted. This would be closer to the reality of an electric car filling up overnight. The goal would be to obtain X units of electricity while gaining the most amount of money (losing the least amount of money) possible.

Possible Problems

A large problem that could be introduced while implemented this Distributed Electrical Resource Management System is the creation of a stock market for electricity. This could introduce the "Efficient Market Hypothesis" where all information becomes null and void as everyone has access to the same information.

The model proposed has only been validated for a short period of time. For the DERM model to be effective it will have to be implemented all year, as that is the driving seasonality to energy production, since the weather works on a yearly timeline. A model that is able to forecast much further in time is needed to create an year long efficient DERM system.

For a DERM system to be better understood, it is recommended that a similar approach to the one in this proposal be taken. The problem is less on how efficient can we forecast the price of electricity but rather, what parameters need to be met to ensure an efficient DERM system. Parameters such as: Decentralized battery storage, latency times, and the push and pull capabilities of the current infrastructure.