Capstone_CYO_MichaelMurray

Michael Murray

May 4, 2019

Executive Summary

The goal of this project is to develop a mathematical model for forecasting Day Ahead Locational Marginal Pricing in the PJM region. PJM Interconnection LLC (PJM) is a regional transmission organization (RTO) in the United States. It is part of the Eastern Interconnection grid operating an electric transmission system serving all or parts of Delaware, Illinois, Indiana, Kentucky, Maryland, Michigan, New Jersey, North Carolina, Ohio, Pennsylvania, Tennessee, Virginia, West Virginia, and the District of Columbia. Locational Marginal Pricing (LMP) represents the cost to buy and sell power at different locations within wholesale electricity markets, usually called Independent System Operators (ISOs). PJM is an ISO. The model should be able to predict tomorrow's average locational marginal pricing in PJM using available historical LMP data. This analysis was performed using the R programming language and publically available data from the PJM portal.

Methods/Analysis

The PJM day ahead locational marginal pricing dataset was used for this analysis and to develop an da lmp forecasting model. The raw dataset was downloaded from pjm.org and the data was processed using Data Wrangling techniques to add columns based on the operational date and time to accomodate time-series and seasonal analysis. The dataset is also reduced in size to limit to make the model run faster for this assignment. The full dataset has hourly day ahead lmp data from June 2010 to May 2019, I am only using data from 2019 for this assignment. The data was divided into two sets, one dataset to create the model using hourly data from 1/1/2019 to 4/18/2019 and one dataset to determine forecasting error using hourly data from 4/18/2019 to 5/3/2019. Time-series analysis using the zoo package is employed to develop the model, the zoo package enables analysis of irregular time series of vectors and factors. The forecast package is used to create the time serices forecast model using the ARIMA function for auto-regressive integrated moving average analysis and calcualting the model error.

Download the PJM DA LMP dataset, created model and test datasets and format the dataset for processing.

```
# Install required packages and libraries
if(!require(zoo)) install.packages("zoo", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 3.5.1
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## Warning: package 'tidyverse' was built under R version 3.5.3
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.0     v forcats 0.3.0
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

```
## Warning: package 'tibble' was built under R version 3.5.1
```

```
## Warning: package 'tidyr' was built under R version 3.5.2
```

```
## Warning: package 'readr' was built under R version 3.5.1
```

```
## Warning: package 'purrr' was built under R version 3.5.1
```

```
## Warning: package 'dplyr' was built under R version 3.5.2
```

```
## Warning: package 'stringr' was built under R version 3.5.1
```

```
## Warning: package 'forcats' was built under R version 3.5.1
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Warning: package 'caret' was built under R version 3.5.1
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(xts)
```

```
## Warning: package 'xts' was built under R version 3.5.1
```

```
##
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```
library(ggplot2)
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 3.5.1
```

```
# Load raw data file containg day-ahead location marginal pricing data for PJM region
rawdata <- read.csv("https://raw.githubusercontent.com/mmurray2073/harvardx_datascience/master/pjm_da_l

# Assign name to dataset
df <- rawdata
```

```r
# Set datetime field to enable hourly time-series analysis for hourly pricing data
df$DATETIME <- paste(df$OPR_DATE,df$OPR_HOUR)
df$DATETIME <- as.POSIXct(df$DATETIME, format = '%m/%d/%Y %H')
df <- data.frame(df$DATETIME, df$PRICE)
names(df) <- c("datetime","price")

# Day of the week
df$day <- as.factor(strftime(df$datetime, format = '%A'))
# Day of the year
df$yearday <- as.factor(strftime(df$datetime, format = '%m%d'))
# Final structure for the study
str(df)
```

```
## 'data.frame':    77597 obs. of  4 variables:
##  $ datetime: POSIXct, format: "2010-06-24 01:00:00" "2010-06-24 02:00:00" ...
##  $ price   : num  30.4 28.1 26.4 25.9 25.8 ...
##  $ day     : Factor w/ 7 levels "Friday","Monday",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ yearday : Factor w/ 366 levels "0101","0102",..: 176 176 176 176 176 176 176 176 176 176 ...
```

```r
# Adjust size of dataset improve model run-time
df <- subset(df, datetime >= strptime('01-01-2019 00:00', format = '%d-%m-%Y %H:%M'))
str(df)
```

```
## 'data.frame':    2951 obs. of  4 variables:
##  $ datetime: POSIXct, format: "2019-01-01 00:00:00" "2019-01-01 01:00:00" ...
##  $ price   : num  20.2 19.4 19.3 19 18.9 ...
##  $ day     : Factor w/ 7 levels "Friday","Monday",..: 6 6 6 6 6 6 6 6 6 6 ...
##  $ yearday : Factor w/ 366 levels "0101","0102",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
# Create test data set
df_test <- subset(df, datetime >= strptime('18-04-2019 00:00', format = '%d-%m-%Y %H:%M'))
df <- subset(df, datetime < strptime('18-04-2019 00:00', format = '%d-%m-%Y %H:%M'))
str(df)
```

```
## 'data.frame':    2566 obs. of  4 variables:
##  $ datetime: POSIXct, format: "2019-01-01 00:00:00" "2019-01-01 01:00:00" ...
##  $ price   : num  20.2 19.4 19.3 19 18.9 ...
##  $ day     : Factor w/ 7 levels "Friday","Monday",..: 6 6 6 6 6 6 6 6 6 6 ...
##  $ yearday : Factor w/ 366 levels "0101","0102",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
str(df_test)
```

```
## 'data.frame':    385 obs. of  4 variables:
##  $ datetime: POSIXct, format: "2019-04-18 00:00:00" "2019-04-18 01:00:00" ...
##  $ price   : num  21.5 19 19.1 18.6 18.2 ...
##  $ day     : Factor w/ 7 levels "Friday","Monday",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ yearday : Factor w/ 366 levels "0101","0102",..: 109 109 109 109 109 109 109 109 109 109 ...
```

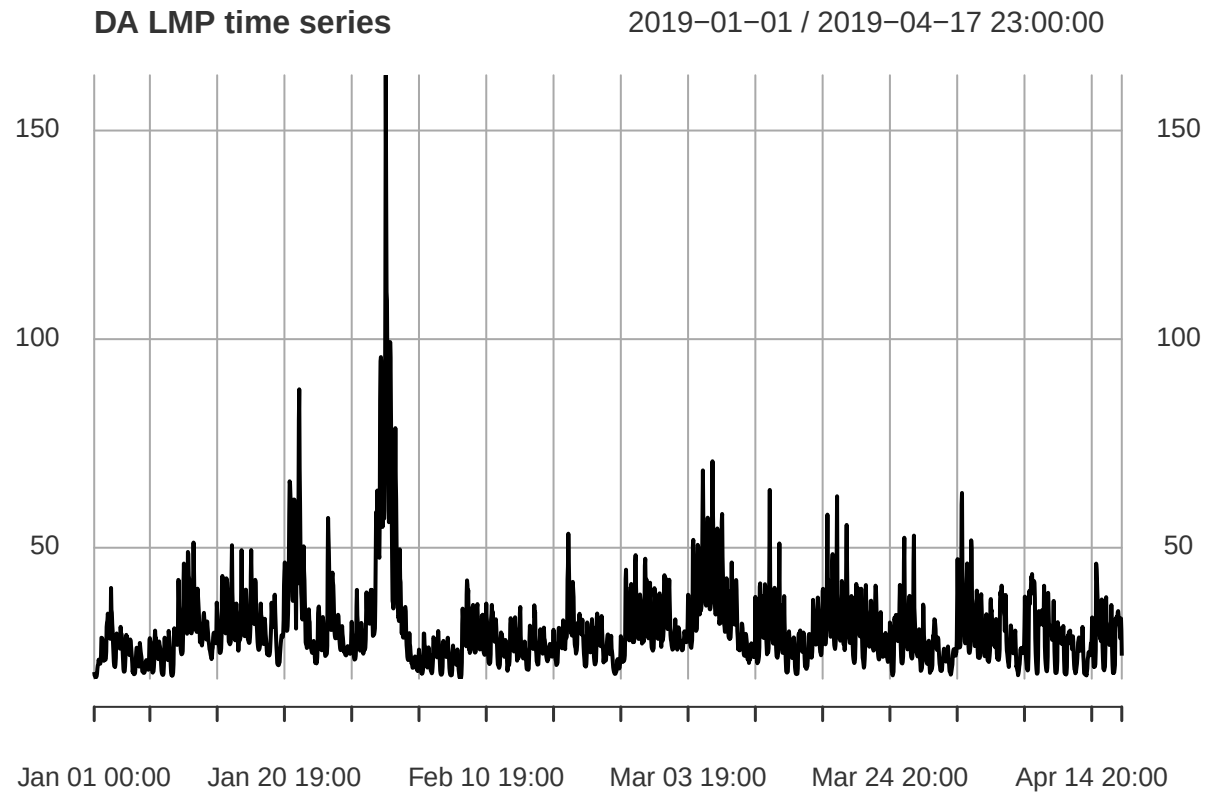
Results

Perform time series analysis of da lmp pricing data.

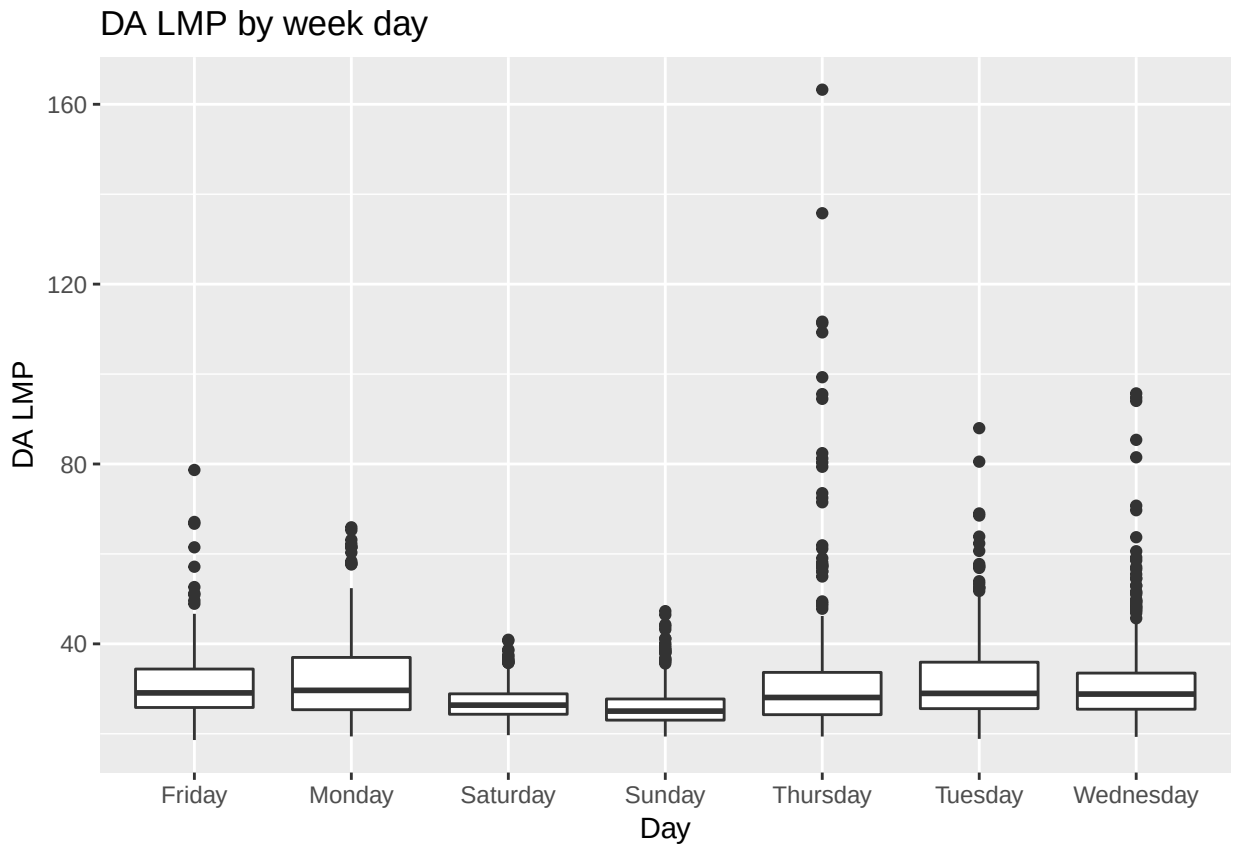
```r
# Create time-series objects
ts <- ts(df$price, frequency = 1)
da_lmp_ts <- xts(df$price, df$datetime)
```

There appears to be seasonality in the data, the pricing variations could also be due to factors like temperature, day of the week (holidays or weekends) and operating hour (on-peak or off-peak). The boxplot shows lower pricing on weekends most likely due to lower demand on weekends.

```
# Plot the time series to visualize pricing across the time period
plot(da_lmp_ts, main = 'DA LMP time series', xlab = 'Date', ylab = 'DA LMP')
```

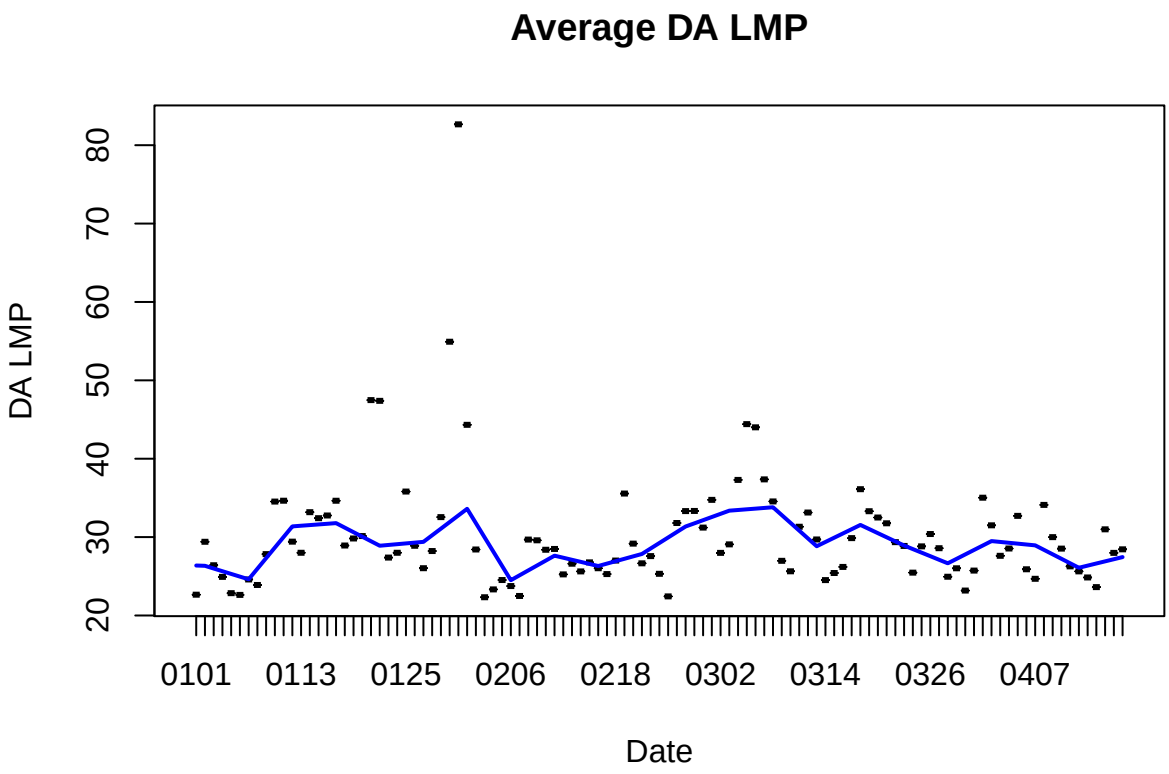



```
# Plot day ahead lmp by day of week
ggplot(df, aes(day, price)) + geom_boxplot() + xlab('Day') + ylab('DA LMP') + ggtitle('DA LMP by week da
```



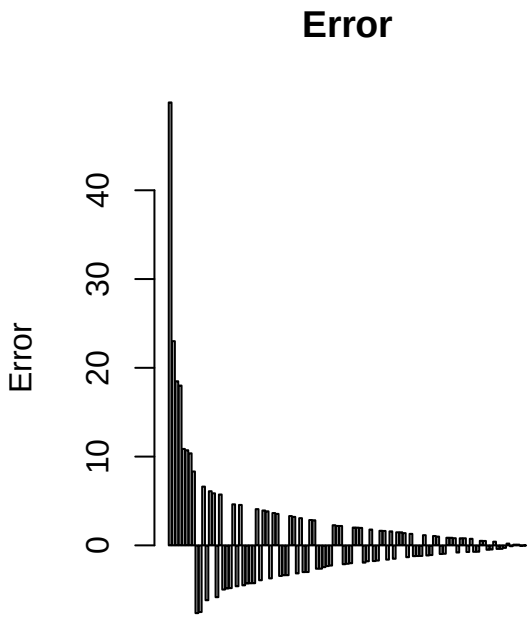
Smooth the curve for time series to highlight patterns or anomolies, seasonality is clear on this plot.

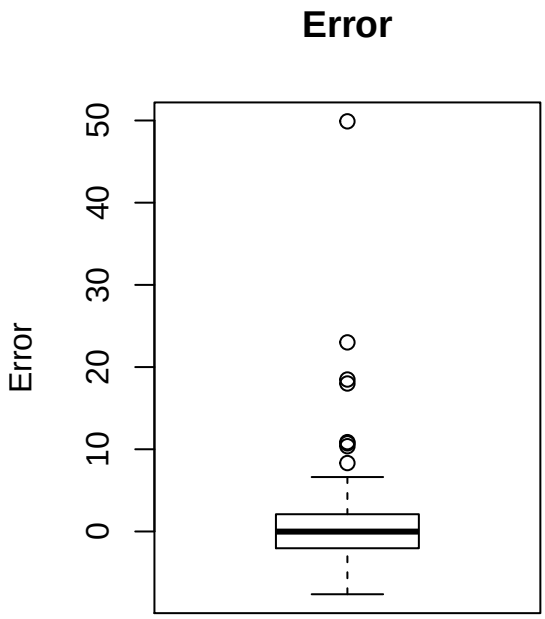
```
# Aggregate an average day ahead lmp per day
avg_da_lmp_per_yearday <- aggregate(price ~ yearday, df, 'mean')
avg_da_lmp_per_yearday$yearday <- factor(avg_da_lmp_per_yearday$yearday)

# Smooth the curve for the time series
smooth_yearday <- rbind(avg_da_lmp_per_yearday, avg_da_lmp_per_yearday, avg_da_lmp_per_yearday, avg_da_l

smooth_yearday <- lowess(smooth_yearday$price, f = 1 / 60)
lts <- length(avg_da_lmp_per_yearday$price)
lts_0 <- 2 * lts + 1
lts_1 <- 3 * lts
smooth_yearday <- smooth_yearday$y[lts_0:lts_1]

# Create plot with smoothed curve
par(mfrow = c(1, 1))
plot(avg_da_lmp_per_yearday$yearday, avg_da_lmp_per_yearday$price, type = 'l', main = 'Average DA LMP',
lines(avg_da_lmp_per_yearday$yearday, smooth_yearday, col = 'blue', lwd = 2)
```



Largest errors are positive.

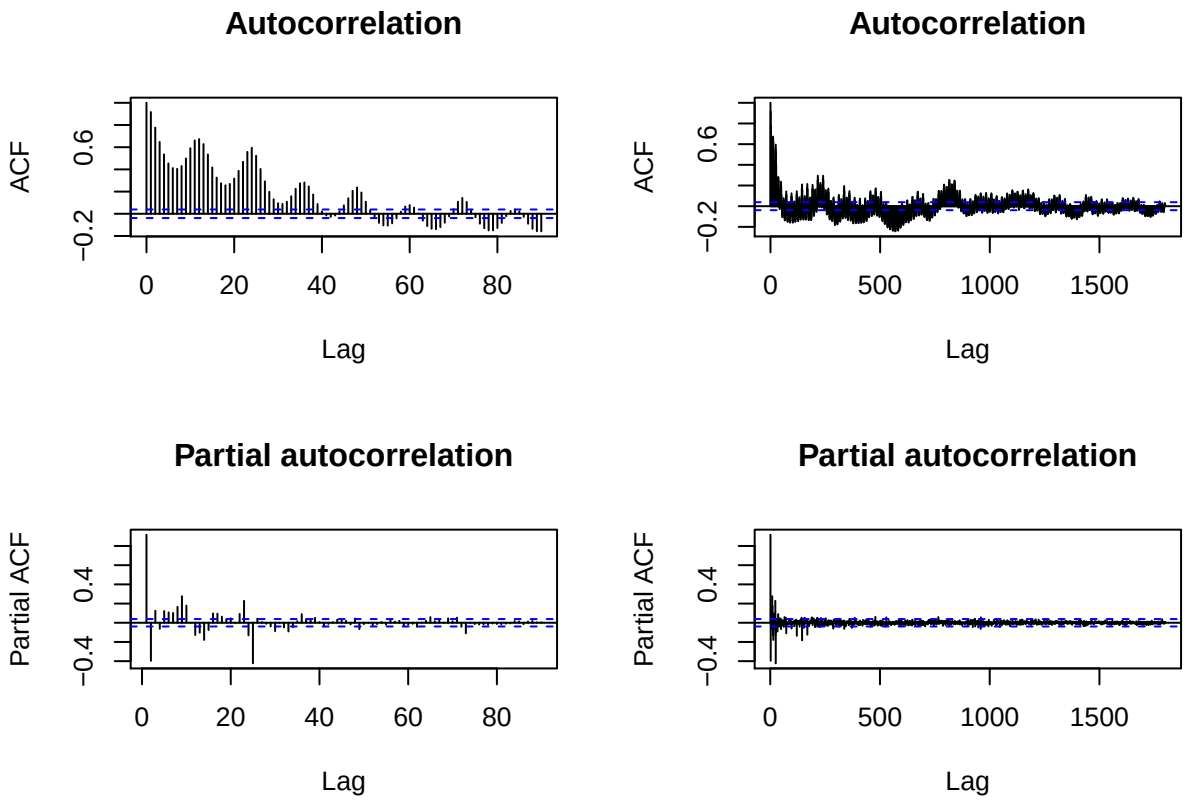
```
# Create bar and box plot showing errors
par(mfrow = c(1, 2))
diff <- avg_da_lmp_per_yearday$price - smooth_yearday
abs_diff <- abs(diff)
barplot(diff[order(-abs_diff)], main = 'Error', ylab = 'Error')
boxplot(diff, main = 'Error', ylab = 'Error')
```





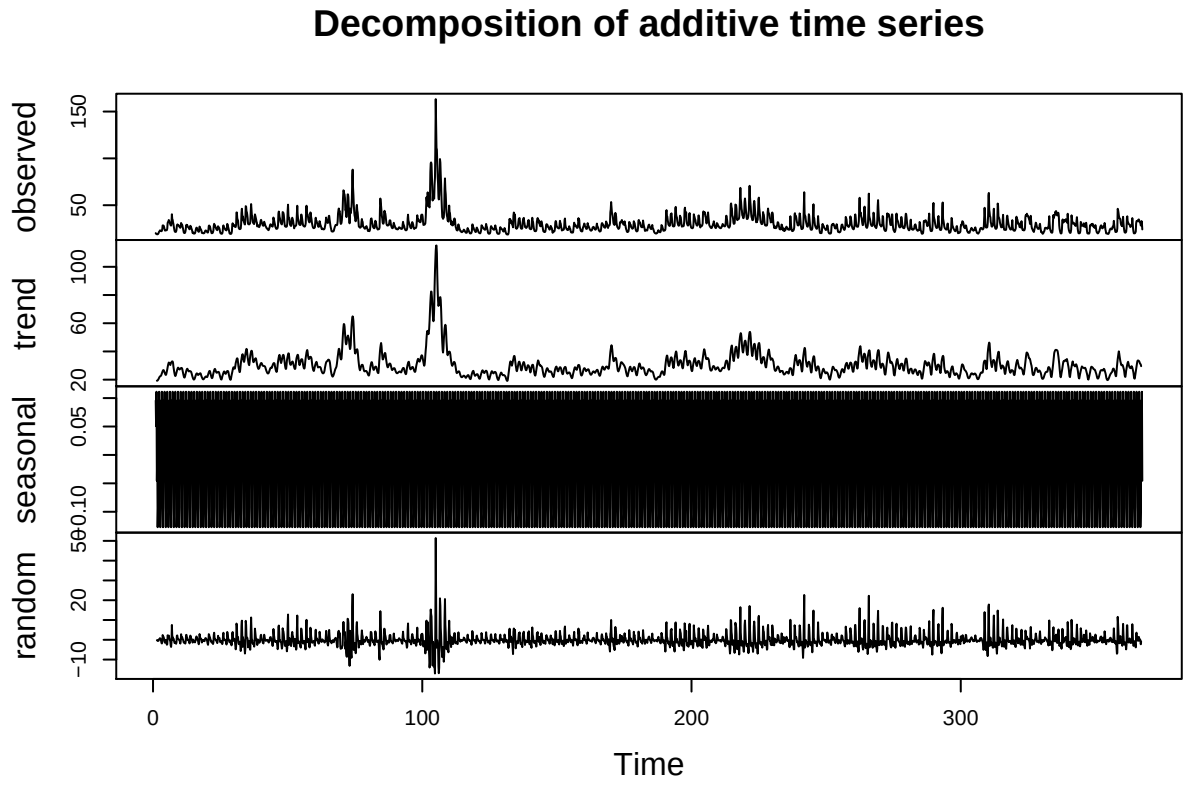
Autocorrelation used to confirm and identify seasonality.

```
# Plot autocorrelation and partial autocorrelation to identify seasonality
par(mfrow = c(2, 2))
acf(df$price, 90, main = 'Autocorrelation')
acf(df$price, 1800, main = 'Autocorrelation')
pacf(df$price, 90, main = 'Partial autocorrelation')
pacf(df$price, 1800, main = 'Partial autocorrelation')
```

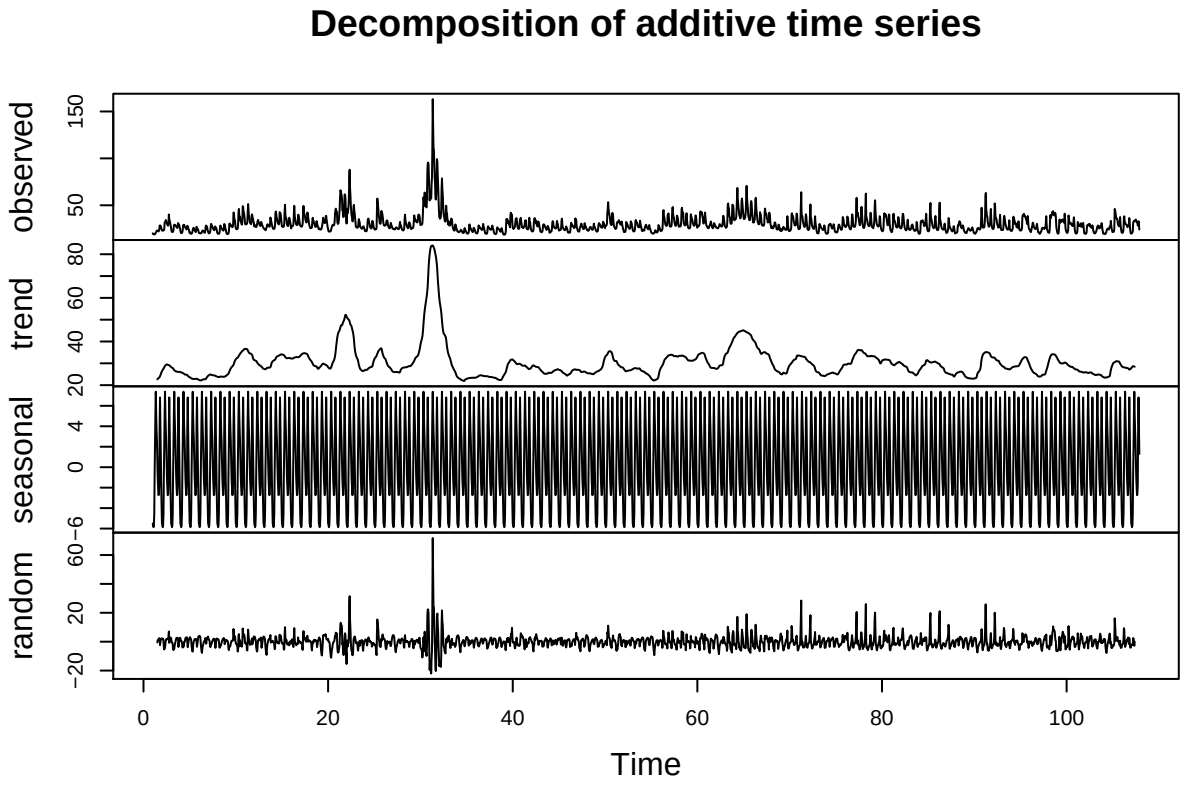


Visualize and correct for weekly seasonality patterns.

```
# Plot decomposition to analyse weekly seasonal patterns
weeklyts <- ts(ts, frequency = 7)
decomp_weeklyts <- decompose(weeklyts)
plot(decomp_weeklyts)
```

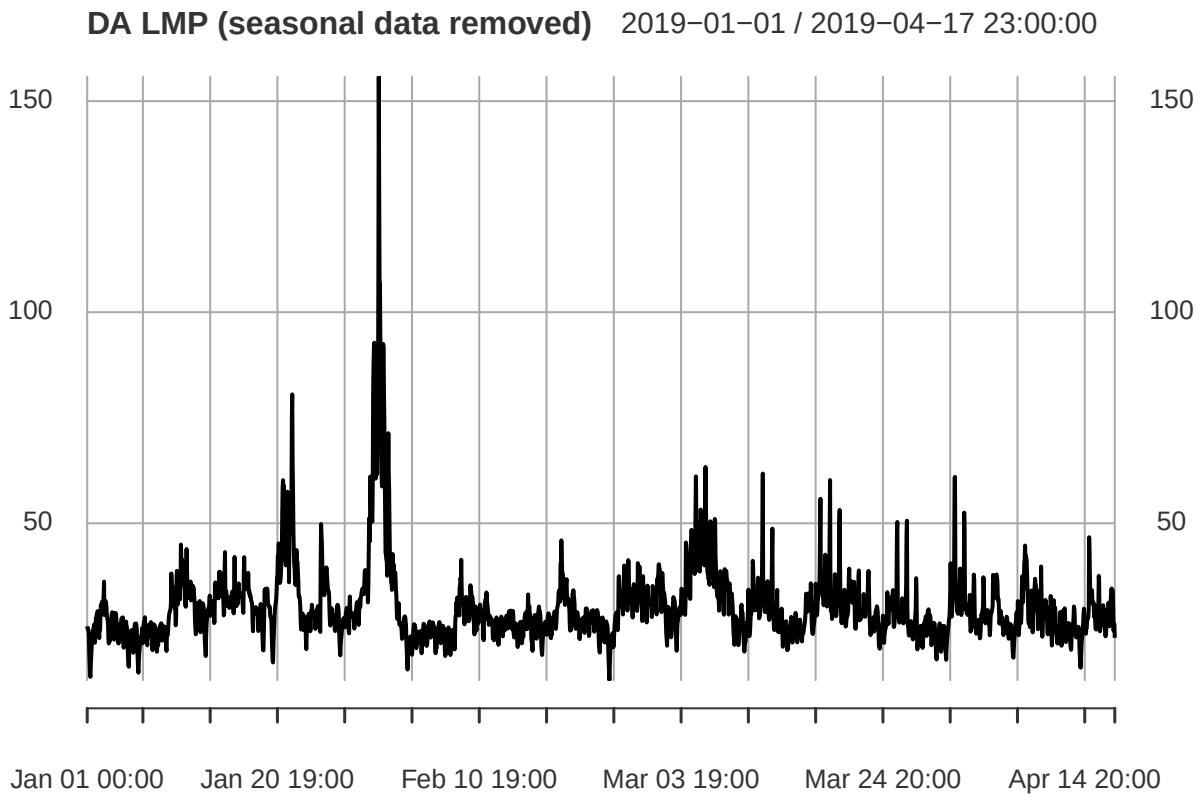



```r
# Adjust for weekly seaonality
df$price_weeklyts <- df$price - as.numeric(decomp_weeklyts$season)

# Plot decomposition to analyse yearly seasonal patterns
yearlyts <- ts(df$price_weeklyts, frequency = 24)
decomp_yearlyts <- decompose(yearlyts)
plot(decomp_yearlyts)
```



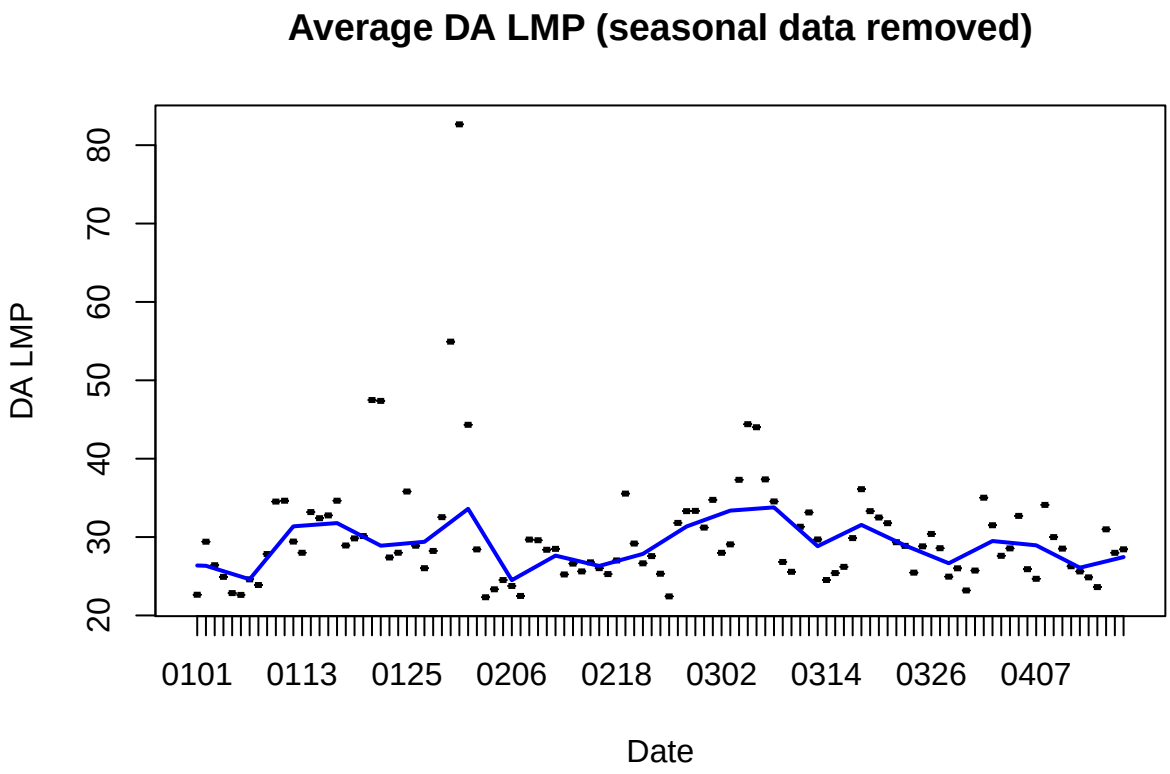
Visualize and correct for yearly seasonality patterns.

```
# Adjust for yearly seasonality
df$price_weeklyts_yearlyts <- df$price_weeklyts - as.numeric(decomp_yearlyts$season)

# Plot day ahead lmp with seasonal data removed
par(mfrow = c(1, 1))
ts_weekly_yearly <- ts(df$price_weeklyts_yearlyts, frequency = 1)
pricets_weekly_yearly <- xts(df$price_weeklyts_yearlyts, df$datetime)
plot(pricets_weekly_yearly, main = 'DA LMP (seasonal data removed)', xlab = 'Date', ylab = 'DA LMP')
```

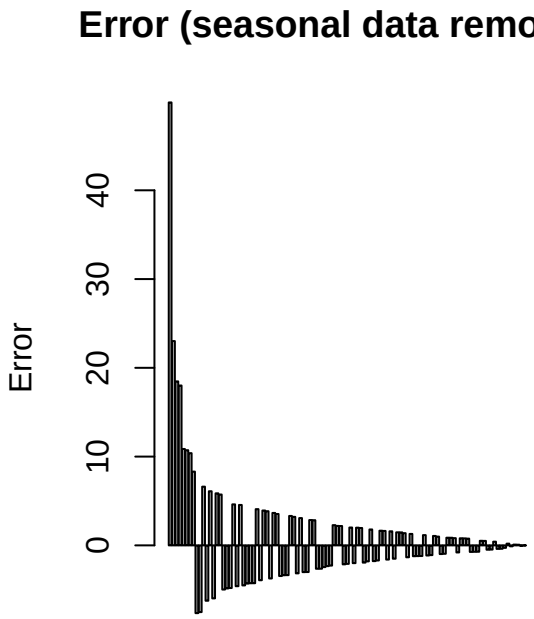


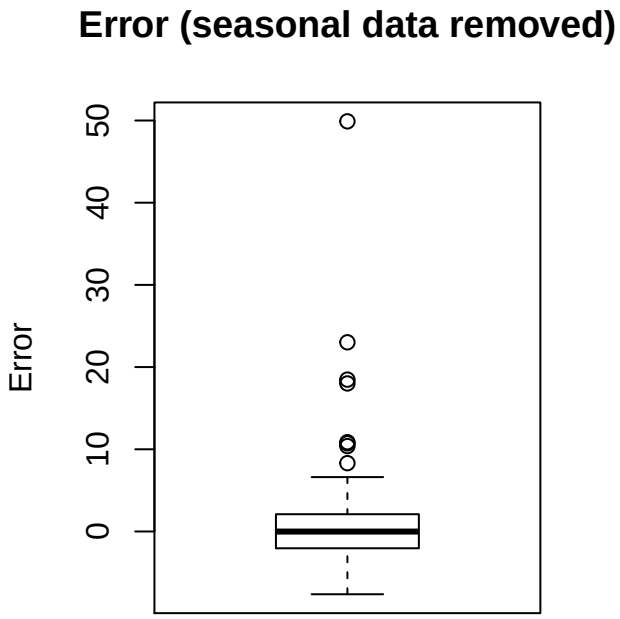
Smooth the data and plot the curve. Autocorrelation shows seasonality.

```
# Aggregating demand by day of the year (average)
avg_lmp_per_yearday <- aggregate(price_weeklyts_yearlyts ~ yearday, df, 'mean')
avg_da_lmp_per_yearday$yearday <- factor(avg_da_lmp_per_yearday$yearday)

# Smooth curve for the time series
smooth_yearday <- rbind(avg_lmp_per_yearday, avg_lmp_per_yearday, avg_lmp_per_yearday, avg_lmp_per_yearc
smooth_yearday <- lowess(smooth_yearday$price_weeklyts_yearlyts, f = 1 / 60)
ltssea <- length(avg_lmp_per_yearday$price_weeklyts_yearlyts)
ltssea_0 <- 2 * ltssea + 1
ltssea_1 <- 3 * ltssea
smooth_yearday <- smooth_yearday$y[ltssea_0:ltssea_1]

# Plotting the seasonally corrected smoothed results
par(mfrow = c(1, 1))
plot(avg_da_lmp_per_yearday$yearday, avg_lmp_per_yearday$price_weeklyts_yearlyts, type = 'l', main = 'A
lines(avg_da_lmp_per_yearday$yearday, smooth_yearday, col = 'blue', lwd = 2)
```

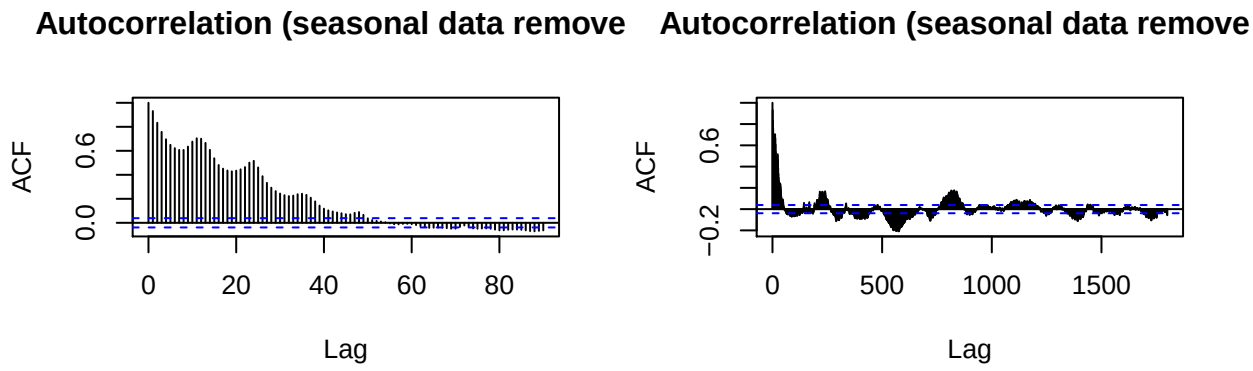


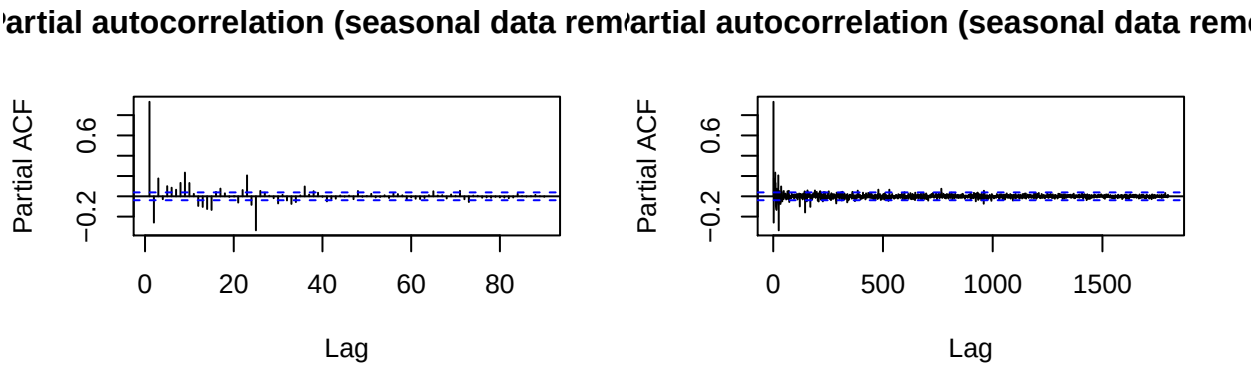

```r
# Create bar and box plot showing errors without seasonality
par(mfrow = c(1, 2))
diff <- avg_lmp_per_yearday$price_weeklyts_yearlyts - smooth_yearday
abs_diff <- abs(diff)
barplot(diff[order(-abs_diff)], main = 'Error (seasonal data removed)', ylab = 'Error')
boxplot(diff, main = 'Error (seasonal data removed)', ylab = 'Error')
```





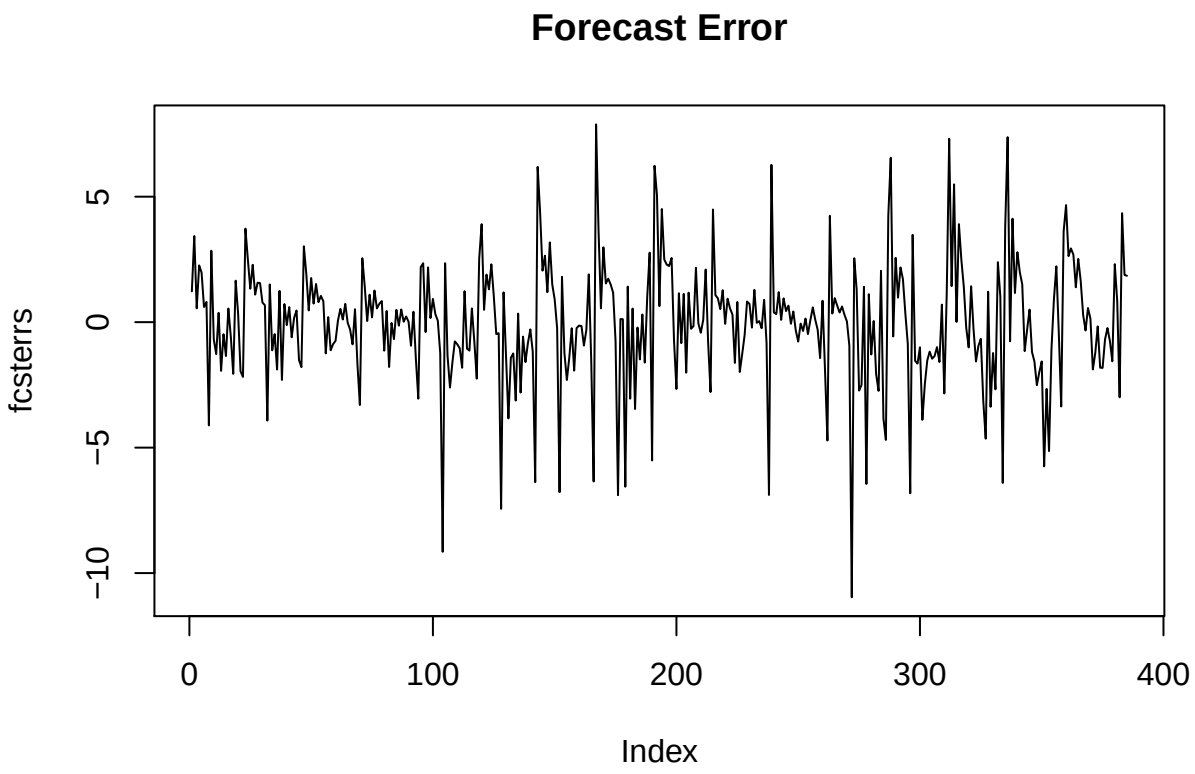

```r
# Autocorrelation and partial correlation on data corrected for seasonality
par(mfrow = c(2, 2))
acf(df$price_weeklyts_yearlyts, 90, main = 'Autocorrelation (seasonal data removed)')
acf(df$price_weeklyts_yearlyts, 1800, main = 'Autocorrelation (seasonal data removed)')
pacf(df$price_weeklyts_yearlyts, 90, main = 'Partial autocorrelation (seasonal data removed)')
pacf(df$price_weeklyts_yearlyts, 1800, main = 'Partial autocorrelation (seasonal data removed)')
```



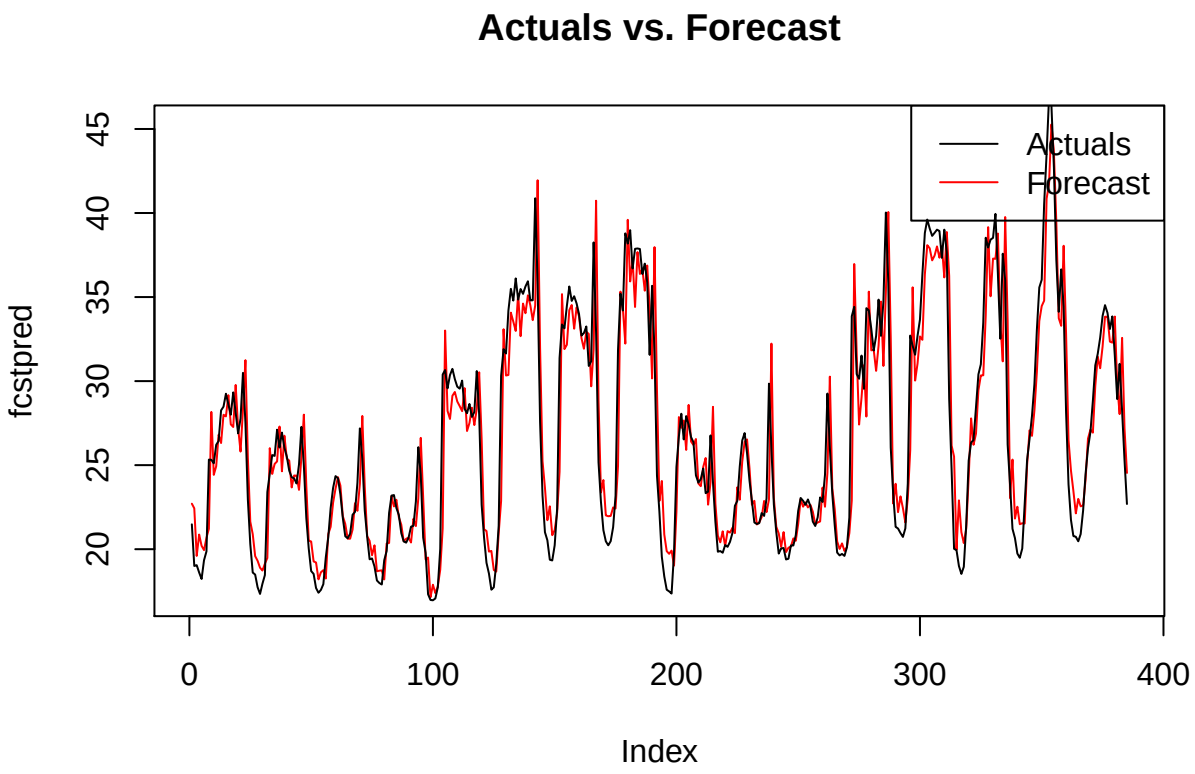


Create day ahead lmp forecast model using auto-regressive integrated moving averages and calculate the forecast error.

```
# Use Arima function to create forecast model
forecast_model <- Arima(ts, order = c(2, 1, 2), list(order = c(1, 1, 1), period = 7))

# Calculate forecast error using test data set
fcstlmpts <- ts
fcstlmpmodel <- forecast_model
fcsterrs <- c()
fcstpred <- c()
fcstperc <- c()
for (i in 1:nrow(df_test)) {
  p <- as.numeric(predict(fcstlmpmodel, newdata = fcstlmpts, n.ahead = 1)$pred)
  fcstpred <- c(fcstpred, p)
  fcsterrs <- c(fcsterrs, p - df_test$price[i])
  fcstperc <- c(fcstperc, (p - df_test$price[i]) / df_test$price[i])
  fcstlmpts <- ts(c(fcstlmpts, df_test$price[i]), frequency = 7)
  fcstlmpmodel <- Arima(fcstlmpts, model = fcstlmpmodel)
}

# Plot forecaset error
par(mfrow = c(1, 1))
plot(fcsterrs, type = 'l', main = 'Forecast Error')
```



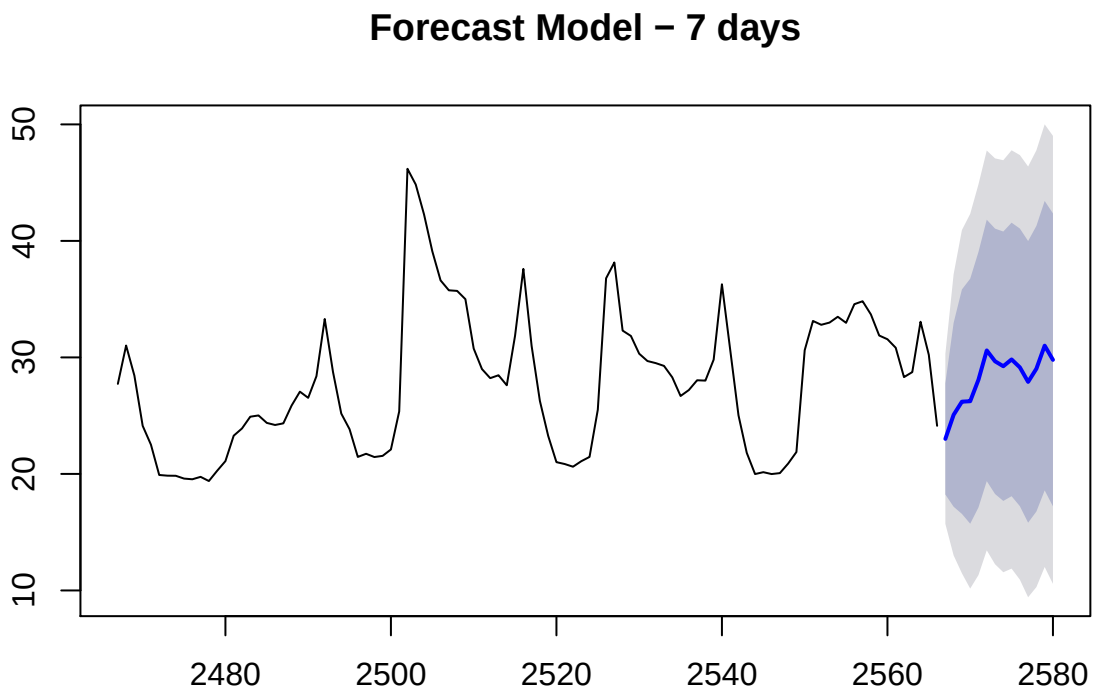

```
# Plot actuals versus forecast
plot(fcstpred, type = 'l', main = 'Actuals vs. Forecast', col = 'red')
lines(df_test$price)
legend('topright', c('Actuals', 'Forecast'), lty = 1, col = c('black', 'red'))
```




```
# Calculate mean error
fcstabserr <- mean(abs(fcsterrs))
fcstpercerr <- mean(abs(fcstperc)) * 100
model_results <- data_frame(method = "Auto-Regressive Integrated Moving Averages (ARIMA)", AbsoluteErro
                            PercentError = fcstpercerr)
model_results %>% knitr::kable()
```

method	AbsoluteError	PercentError
Auto-Regressive Integrated Moving Averages (ARIMA)	1.729739	6.412644

```
# Forecast model plot
plot(forecast(Arima(tail(ts, 100), model = forecast_model)), main = 'Forecast Model - 7 days')
```



Conclusion

Data analysis showed da lmp pricing data is impacted by seasonality and after adjusting for that the forecast model error was calculated at 6.4%. The model could potentially be improved by incorporating actual generation data and weather data for the sites and region being analyzed.