Final Project Report

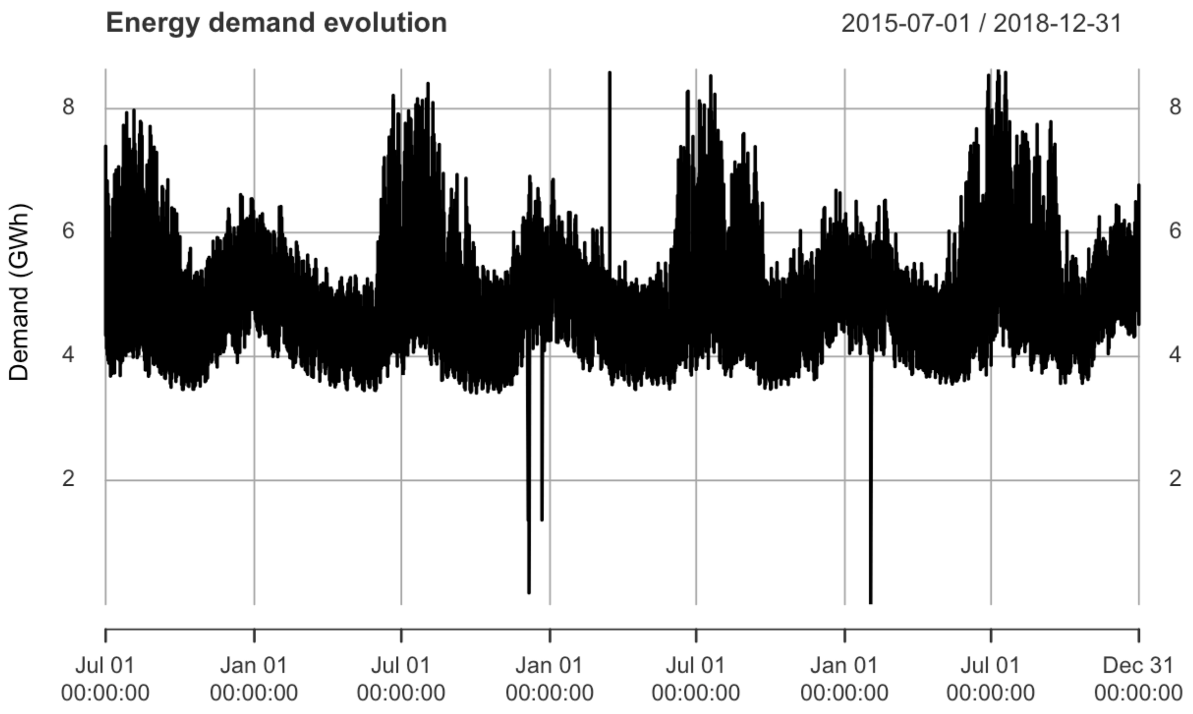
Background

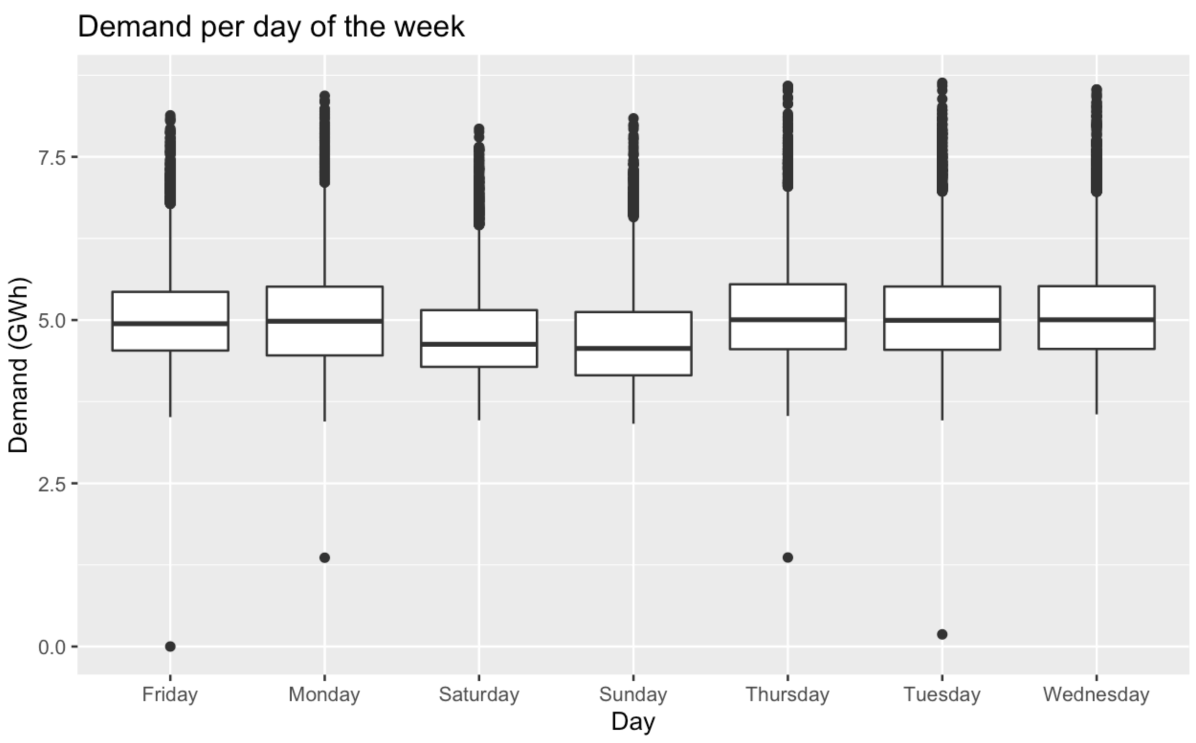
This project aims to calculate and predict the electric energy demand from the Xcel Energy grid in Colorado on Sunday (May 1st) from 5 pm to 6 pm. The data on energy demand in Colorado is available from the US Energy Information Administration (EIA) at /www.eia.gov/opendata. Look for EIA. For reference, the data is located at the following link: (https://www.eia.gov/opendata/qb.php?category=3389994&sdid=EBA.PSCO-ALL.D.HL). This analysis aims to develop a mathematical model for energy demand forecasting in Spain. The model should predict future demand by having data on the daily demand the days before. The research uses R programming language, and all the data and code needed to reproduce the study are provided within this document to ensure fully reproducible research. The data is in CSV format, with values of the hourly energy demand per day in MegaWatts per hour (MWh) between 2015 and the present. The data is converted into GegaWatts per hour (GWh) and divided into two sets, one to create the model containing data from 2015 and 2019 and the other to calculate forecasting-error from the 2019 data till the present day. Additionally, the data-set is scrubbed and cleaned to remove all missing data and any extreme outliers as they can affect the results of the analysis. Following the scrubbing and filtering of the raw data, the next step is to ensure the data is transformed into the data structures and formats needed to complete our analysis. Accordingly, the initial data frame is formatted, dates are converted to POSIX format, new columns are added to contain information about the day of the week and the day of the year, and the data set is divided to create the needed test set mentioned earlier.

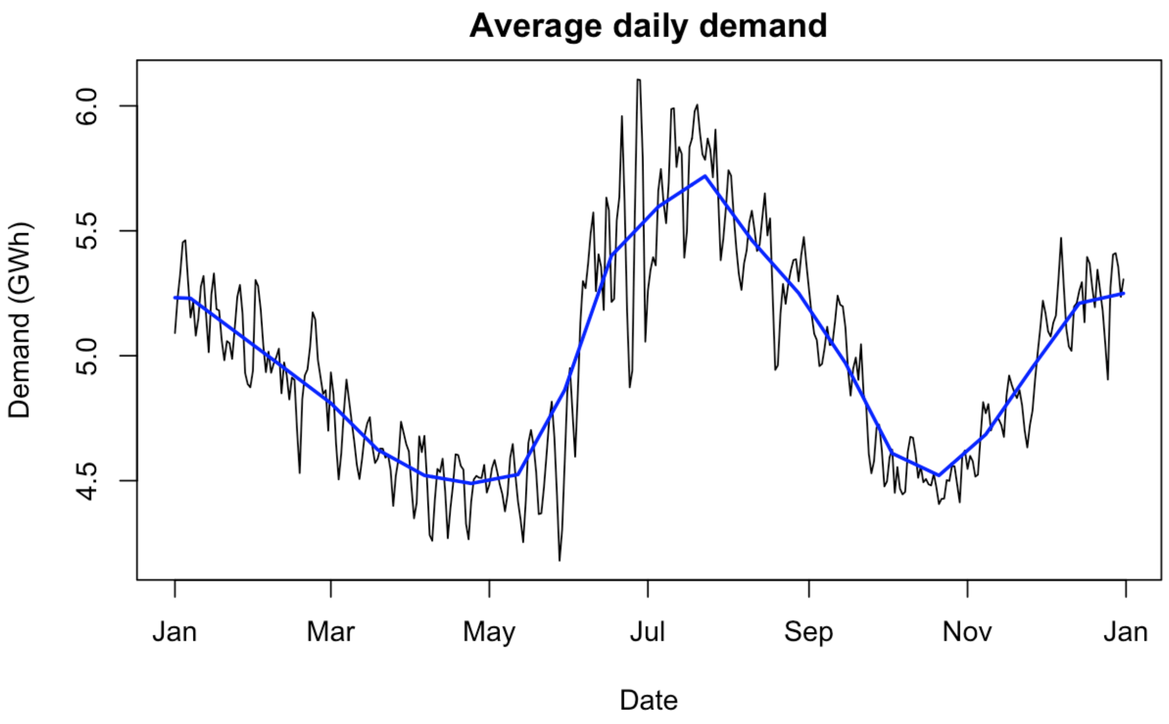
Analysis

The code snippet and figures below utilize a time-series plot to visualize the demand across time. Although the plot indicates a seasonal dependency, other aspects may affect the results, such as the temperature, holidays, weekends, day-light savings, and socio-economic influences to name a few. Additionally, weekends are correlated with a demand decrease when compared to the rest of the weekdays. Finally, winter and summer indicate a higher demand, while holidays are associated with lower demand.

```r
demandts <- xts(df$demand, df$date)
plot(demandts, main = 'Energy demand evolution', xlab = 'Date', ylab = 'Demand (GWh)')

ggplot(df, aes(day, demand)) + geom_boxplot() + xlab('Day') + ylab('Demand (GWh)') + ggtitle('Demand per day of the week')

avg_demand_per_yearday <- aggregate(demand ~ yearday, df, 'mean')
# Computing the smooth curve for the time series. Data is replicated before computing the curve in order to achieve continuity
smooth_yearday <- rbind(avg_demand_per_yearday, avg_demand_per_yearday, avg_demand_per_yearday, avg_demand_per_yearday, avg_demand_per_yearday)
smooth_yearday <- lowess(smooth_yearday$demand, f = 1 / 45)
l <- length(avg_demand_per_yearday$demand)
l0 <- 2 * l + 1
l1 <- 3 * l
smooth_yearday <- smooth_yearday$y[l0:l1]
# Plotting the result
par(mfrow = c(1, 1))
# Setting year to 2016 to allow existence of 29th February
dates <- as.Date(paste(levels(df$yearday), '2016'), format = '%m%d%Y')
plot(dates, avg_demand_per_yearday$demand, type = 'l', main = 'Average daily demand', xlab = 'Date', ylab = 'Demand (GWh)')
lines(dates, smooth_yearday, col = 'blue', lwd = 2)
```

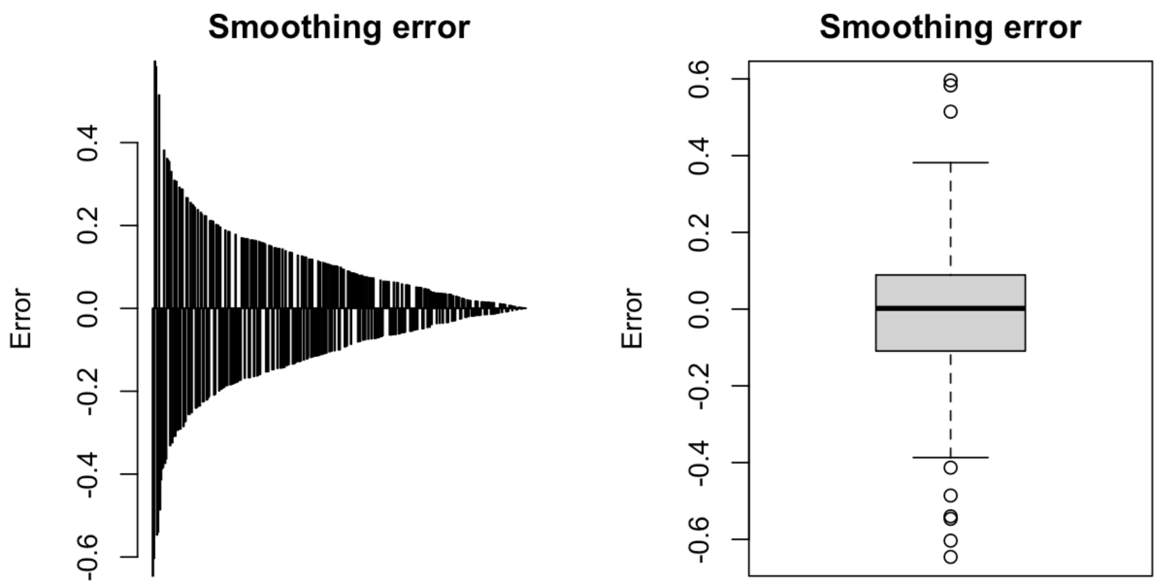



Demand per day of the week
Demand (GWh)
7.5
5.0
2.5
0.0
Friday
Monday
Saturday
Sunday
Thursday
Tuesday
Wednesday
Day

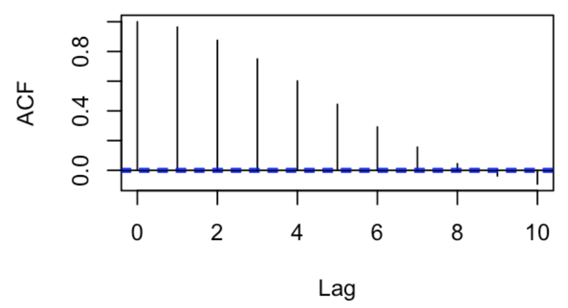

Average daily demand
Demand (GWh)
6.0
5.5
5.0
4.5
Jan
Mar
May
Jul
Sep
Nov
Jan
Date

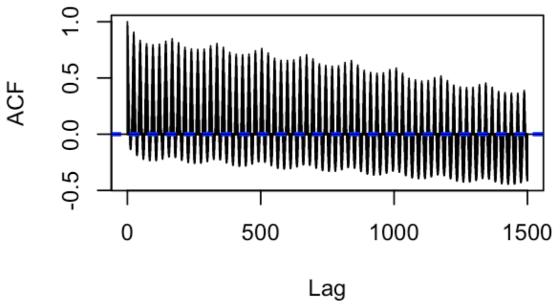
Next, we look at the generated errors, which seem to be centered around zero. The autocorrelation function (ACF), which shows a highly autocorrelated seasonal non-stationary process with yearly and weekly cycles. At the same time, the ACF highlights orders of dependence for Auto-Regression Moving Average (ARMA) or Auto-Regression (AR) processes, and the Partial-autocorrelation (PACF) shows the weekly and yearly seasonality with enhanced clarity, with the correlation decreasing when the lag increases. (see figure below).

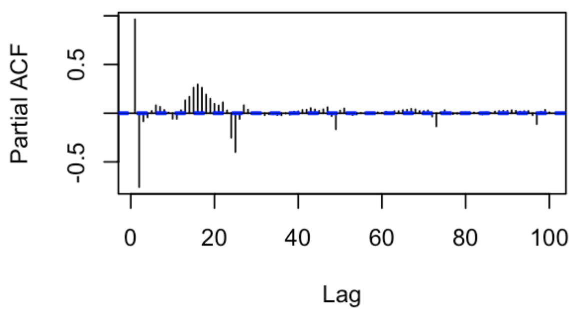
```
par(mfrow = c(1, 2))
diff <- avg_demand_per_yearday$demand - smooth_yearday
abs_diff <- abs(diff)
barplot(diff[order(-abs_diff)], main = 'Smoothing error', ylab = 'Error')
boxplot(diff, main = 'Smoothing error', ylab = 'Error')
```

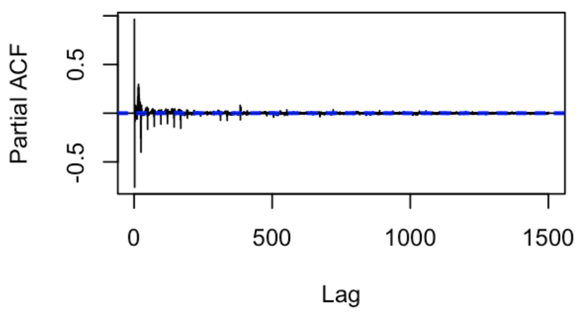


```
head(strftime(dates[order(-abs_diff)], format = '%B %d'), 33)

par(mfrow = c(2, 2))
acf(df$demand, 10, na.action = na.pass, main = 'Autocorrelation')
acf(df$demand, 1500, na.action = na.pass, main = 'Autocorrelation')
pacf(df$demand, 100, na.action = na.pass, main = 'Partial autocorrelation')
pacf(df$demand, 1500, na.action = na.pass, main = 'Partial autocorrelation')
```

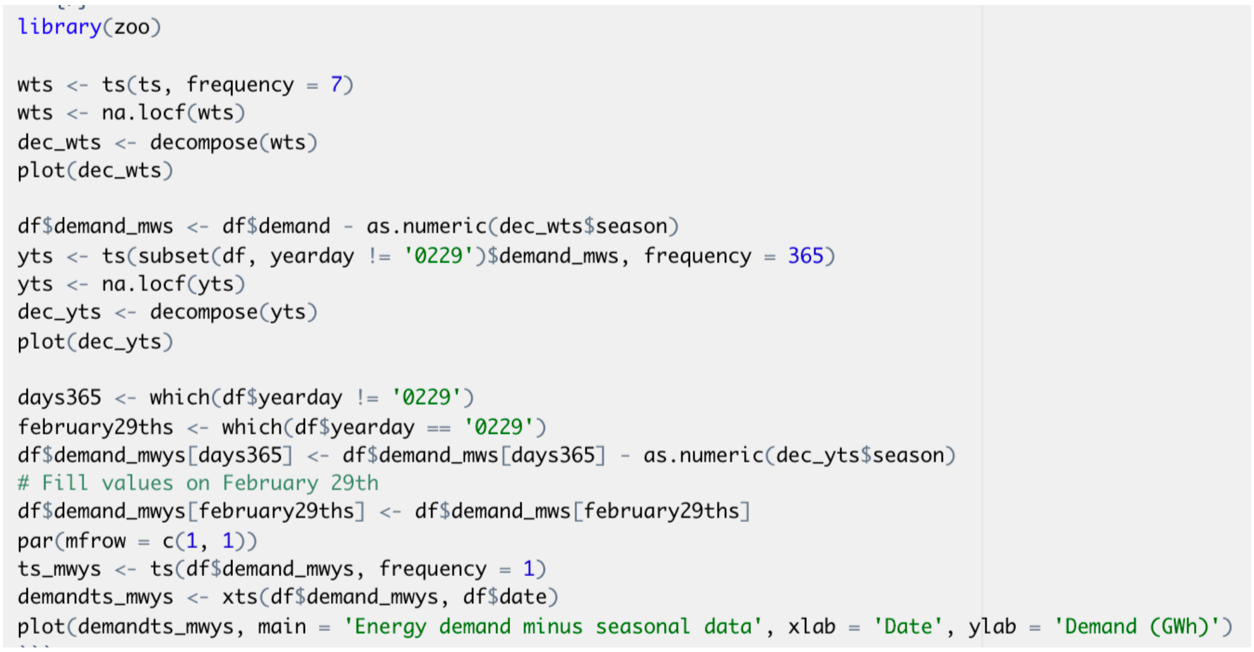




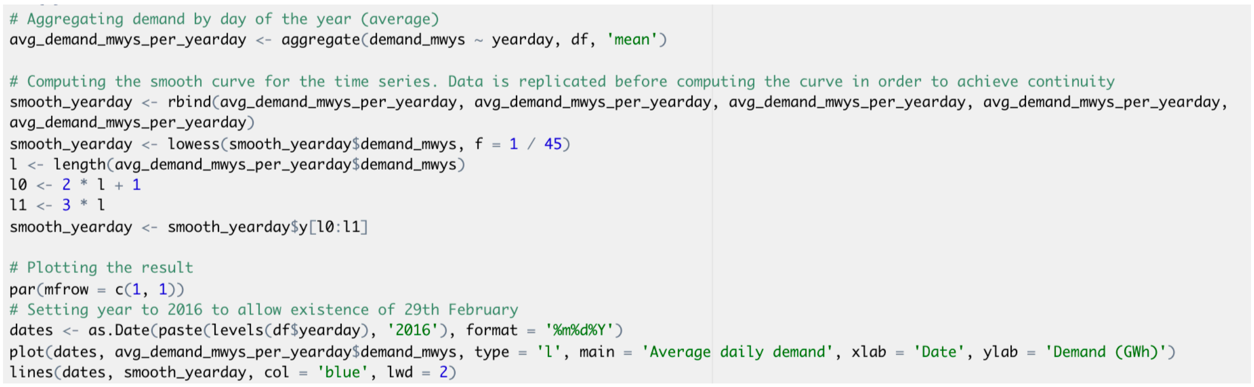




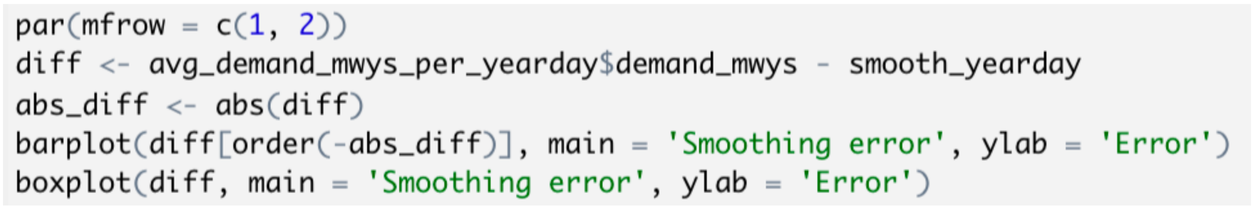
We look at the frequency decomposition to further our analysis to identify trends, seasonality, and noise. The decomposition process is initially applied with leap-years taken into perspective, and then without accounting for frequency matching. This is calculated from the original observation minus the weekly seasonal data using the existing function in R as per the below code and graphs. Frequency analysis helps predict how often specific values of variables may be prevalent while assessing the prediction's reliability. Accordingly, a new time series forms from the original observation minus the weekly and yearly seasonal data. Plotting the average daily demand (demand minus the seasonal data) provides a perspective on the new errors, ACF, and PACF.



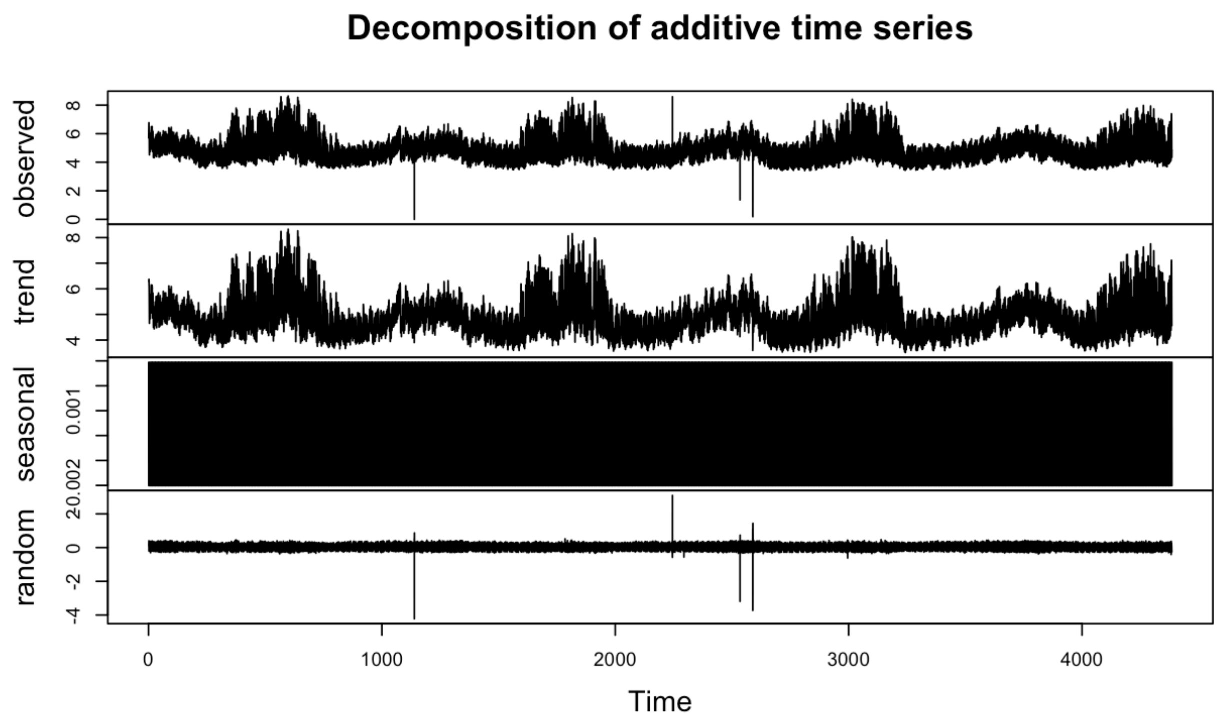
```r
library(zoo)

wts <- ts(ts, frequency = 7)
wts <- na.locf(wts)
dec_wts <- decompose(wts)
plot(dec_wts)

df$demand_mws <- df$demand - as.numeric(dec_wts$season)
yts <- ts(subset(df, yearday != '0229')$demand_mws, frequency = 365)
yts <- na.locf(yts)
dec_yts <- decompose(yts)
plot(dec_yts)

days365 <- which(df$yearday != '0229')
february29ths <- which(df$yearday == '0229')
df$demand_mwys[days365] <- df$demand_mws[days365] - as.numeric(dec_yts$season)
# Fill values on February 29th
df$demand_mwys[february29ths] <- df$demand_mws[february29ths]
par(mfrow = c(1, 1))
ts_mwys <- ts(df$demand_mwys, frequency = 1)
demandts_mwys <- xts(df$demand_mwys, df$date)
plot(demandts_mwys, main = 'Energy demand minus seasonal data', xlab = 'Date', ylab = 'Demand (GWh)')
```

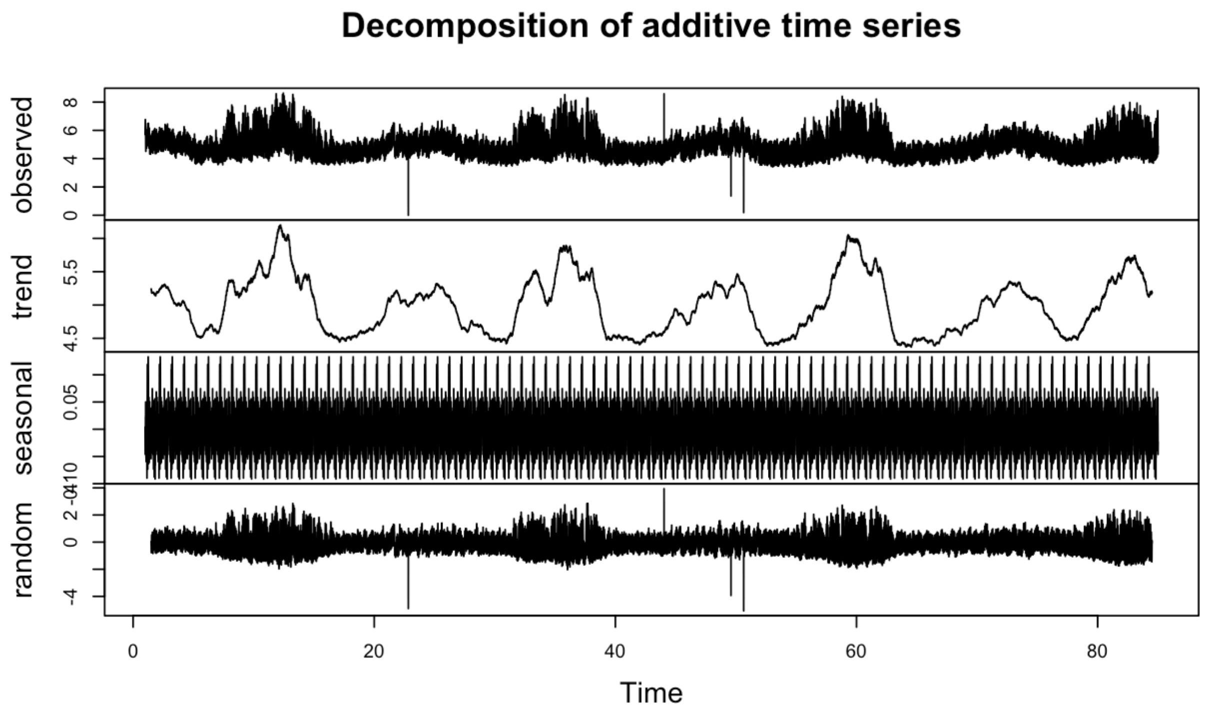


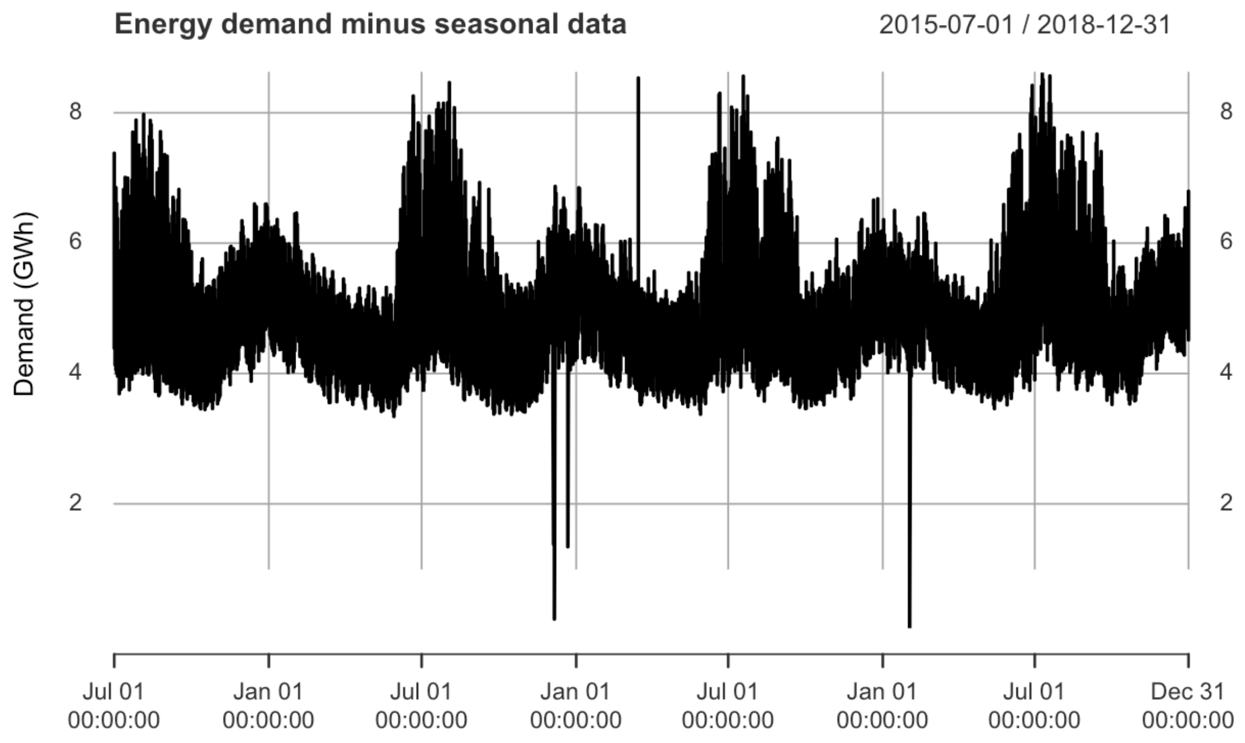
```r
# Aggregating demand by day of the year (average)
avg_demand_mwys_per_yearday <- aggregate(demand_mwys ~ yearday, df, 'mean')

# Computing the smooth curve for the time series. Data is replicated before computing the curve in order to achieve continuity
smooth_yearday <- rbind(avg_demand_mwys_per_yearday, avg_demand_mwys_per_yearday, avg_demand_mwys_per_yearday, avg_demand_mwys_per_yearday,
avg_demand_mwys_per_yearday)
smooth_yearday <- lowess(smooth_yearday$demand_mwys, f = 1 / 45)
l <- length(avg_demand_mwys_per_yearday$demand_mwys)
l0 <- 2 * l + 1
l1 <- 3 * l
smooth_yearday <- smooth_yearday$y[l0:l1]

# Plotting the result
par(mfrow = c(1, 1))
# Setting year to 2016 to allow existence of 29th February
dates <- as.Date(paste(levels(df$yearday), '2016'), format = '%m%d%Y')
plot(dates, avg_demand_mwys_per_yearday$demand_mwys, type = 'l', main = 'Average daily demand', xlab = 'Date', ylab = 'Demand (GWh)')
lines(dates, smooth_yearday, col = 'blue', lwd = 2)
```

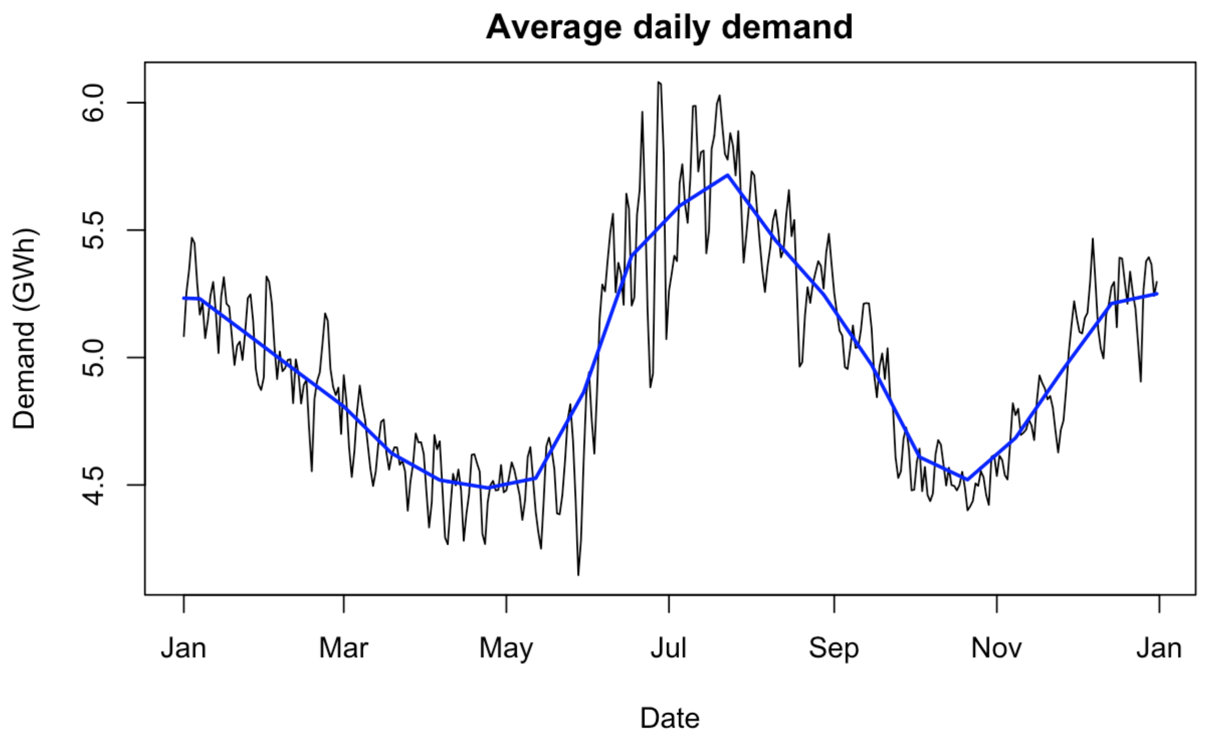


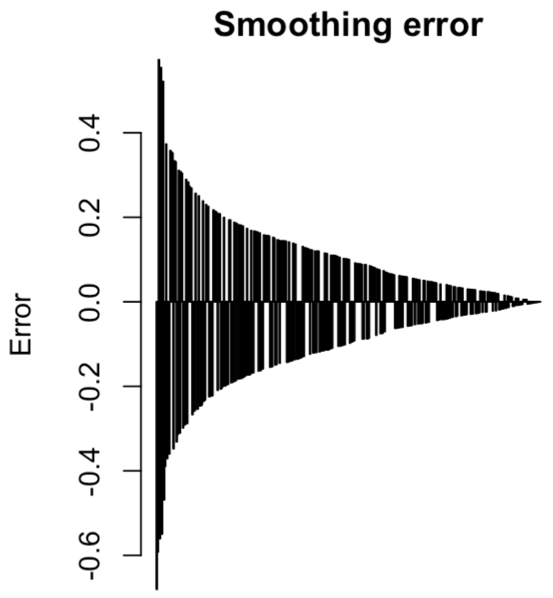
```r
par(mfrow = c(1, 2))
diff <- avg_demand_mwys_per_yearday$demand_mwys - smooth_yearday
abs_diff <- abs(diff)
barplot(diff[order(-abs_diff)], main = 'Smoothing error', ylab = 'Error')
boxplot(diff, main = 'Smoothing error', ylab = 'Error')
```

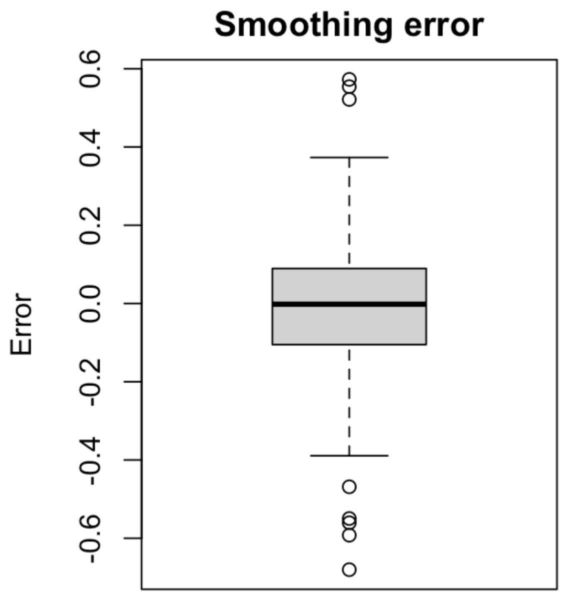

Decomposition of additive time series
observed
trend
seasonal
random
Time
0
1000
2000
3000
4000

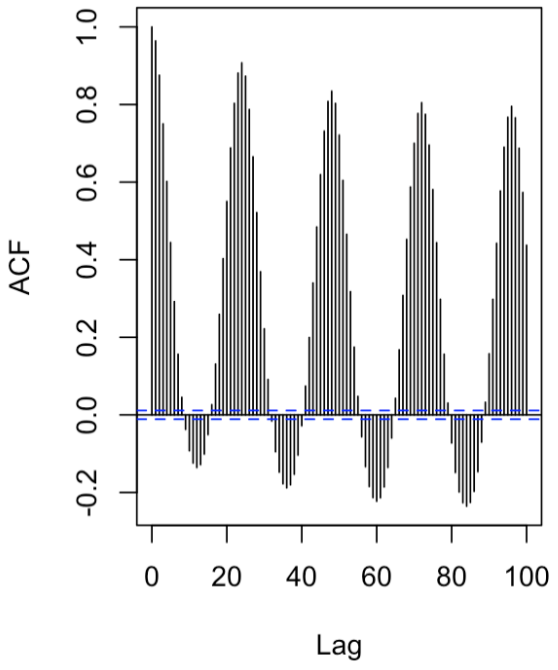

Decomposition of additive time series
observed
trend
seasonal
random
Time
0
20
40
60
80

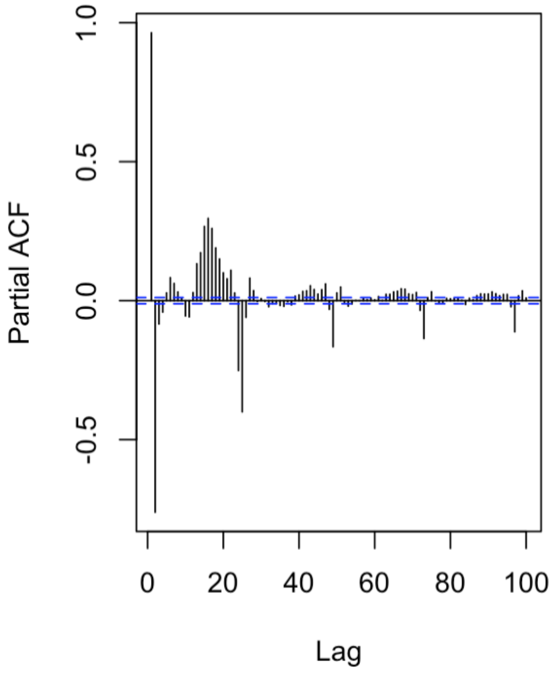

Energy demand minus seasonal data
2015-07-01 / 2018-12-31
Demand (GWh)
8
6
4
2
Jul 01 00:00:00
Jan 01 00:00:00
Jul 01 00:00:00
Jan 01 00:00:00
Jul 01 00:00:00
Jan 01 00:00:00
Jul 01 00:00:00
Dec 31 00:00:00


Average daily demand
Demand (GWh)
6.0
5.5
5.0
4.5
Jan
Mar
May
Jul
Sep
Nov
Jan
Date


Smoothing error
Error
0.4
0.2
0.0
-0.2
-0.4
-0.6

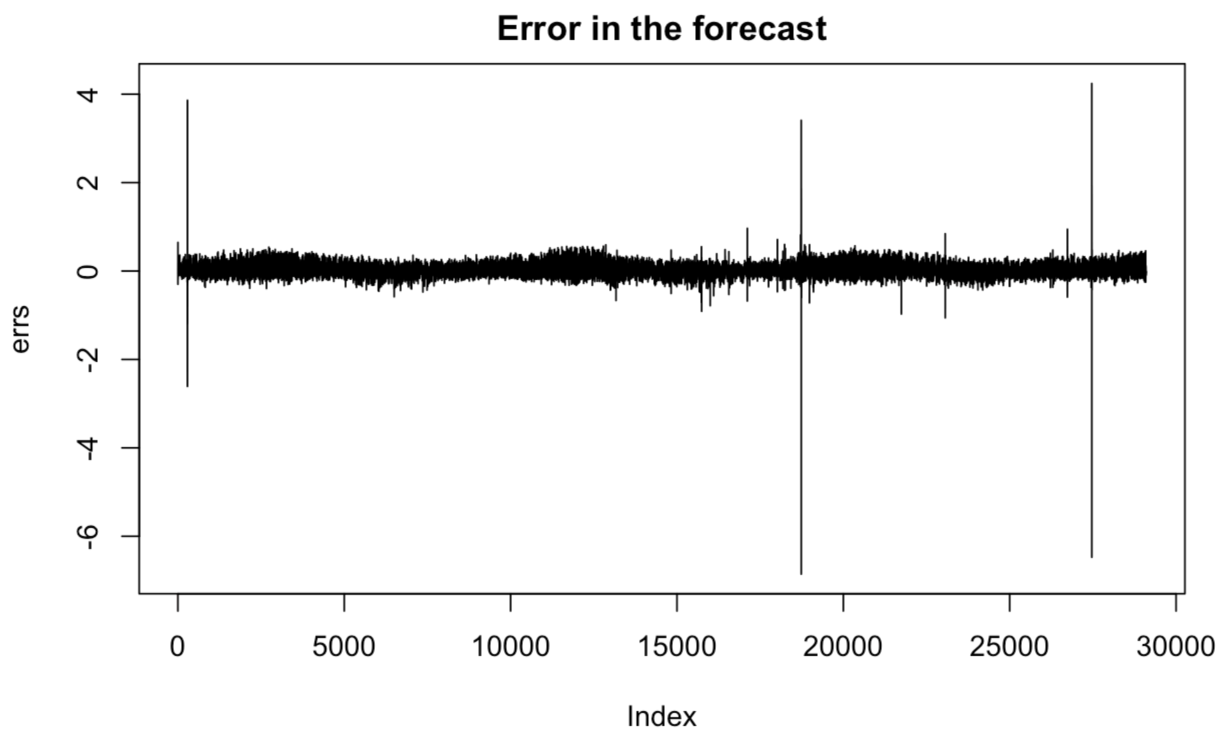

Smoothing error
Error
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6

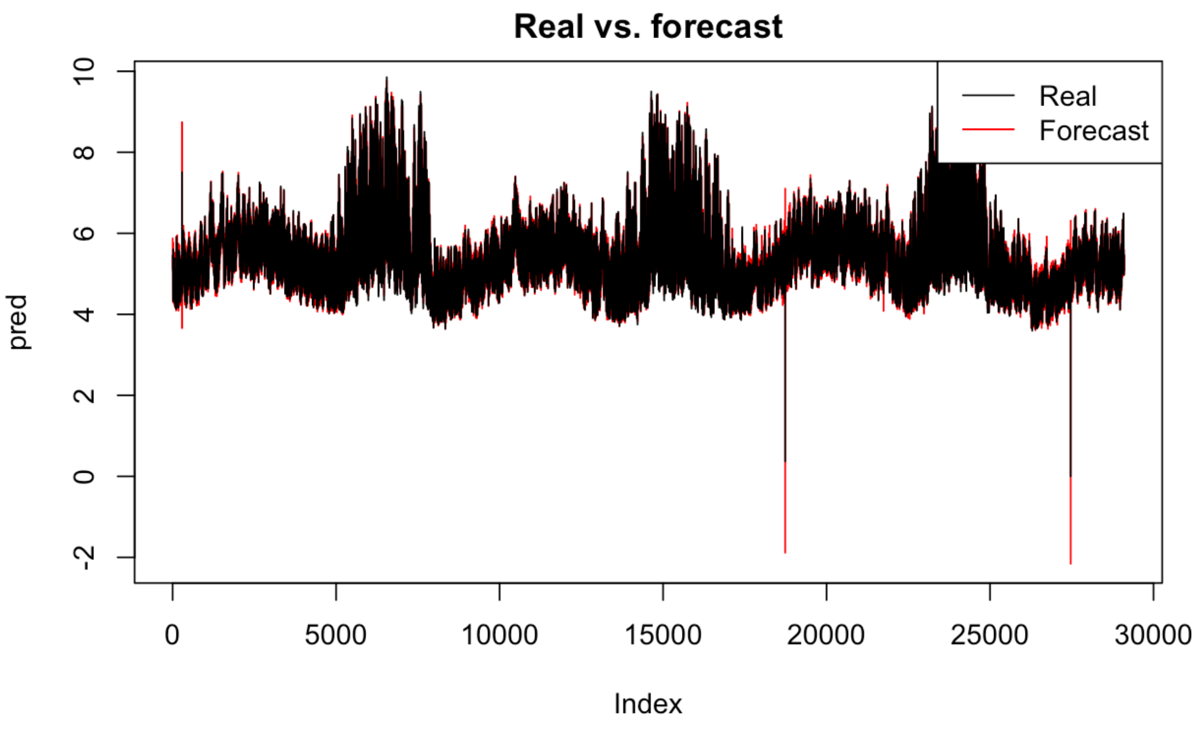

ACF
Lag
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
0
20
40
60
80
100


Partial ACF
Lag
1.0
0.5
0.0
-0.5
0
20
40
60
80
100

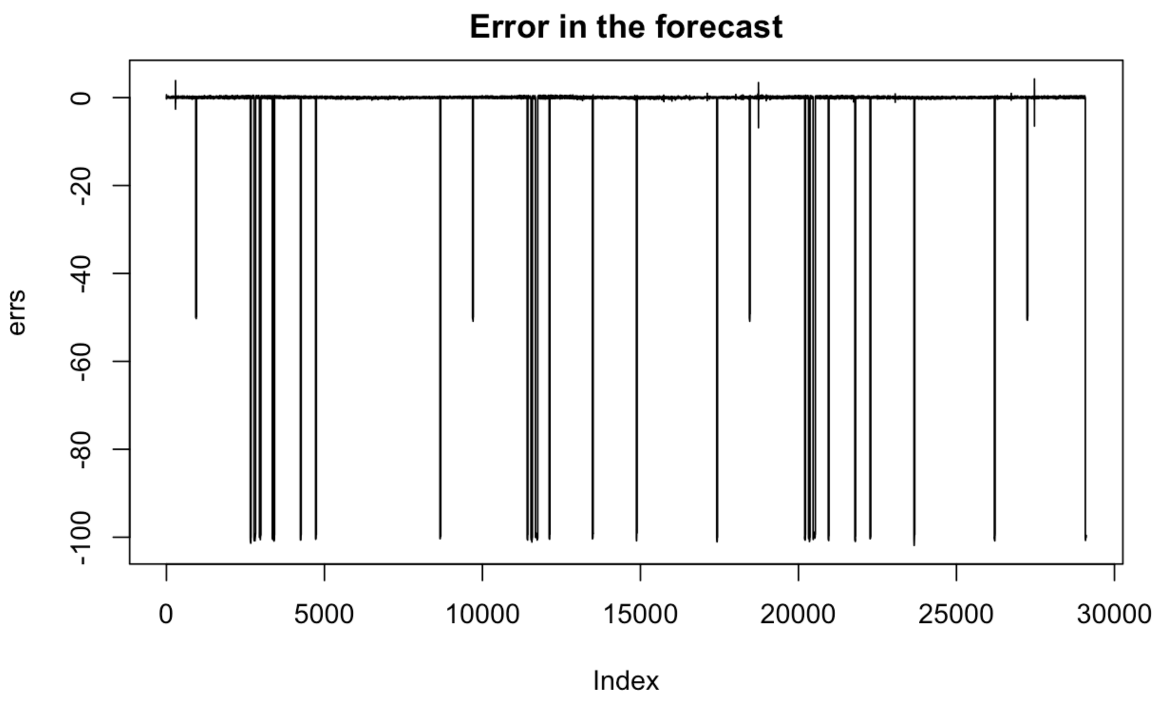
Given the prevailing seasonality in the data, we utilize Seasonal Autoregressive Integrated Moving Average (SARIMA). The difference between ARIMA and SARIMA is the seasonality of the dataset, while our variables (p, q, d) values are unchanged. This allows the ability to capture trends with nonseasonal differencing and seasonality with seasonal differencing. The ARIMA parameters are defined through R's auto.arima() function, while the differencing parameter (d) is selected using the Kwiatkowski Phillips Schmidt Shin (KPSS) test, which is used in testing the null hypothesis of an observable time series for stationarity around a deterministic trend against the alternative of a unit root. If we fail to reject the null hypothesis with the KPSS application to the original time series, then we assume (d = 0). Else, the series is differenced until the KPSS test fails to reject the null hypothesis. Once successful, the parameters (p) and (q) are identified by utilizing either the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC), which provide measures of the model performance while accounting for the model complexity in proportion to its number of parameters. Furthermore, the SARIMA model is created using the ARIMA parameters, while the forecasting error can be calculated by iterating through the test data frame. Accounting for some days where demand is significantly less than others, We run the model once more to reduce the errors. As per the code and figures below, the mean error for the original model across the test data frame is approximately (+/- 0.0 GWh), while the mean error for the adjusted model across the test data frame is approximately (+/- GWh). Clearly the original model has a better fit, though it raises the concern of overfitting the model, though the utilized methods in the process so far does not indicate any conflicts in the decision to proceed with the original model for forecasting and prediction.

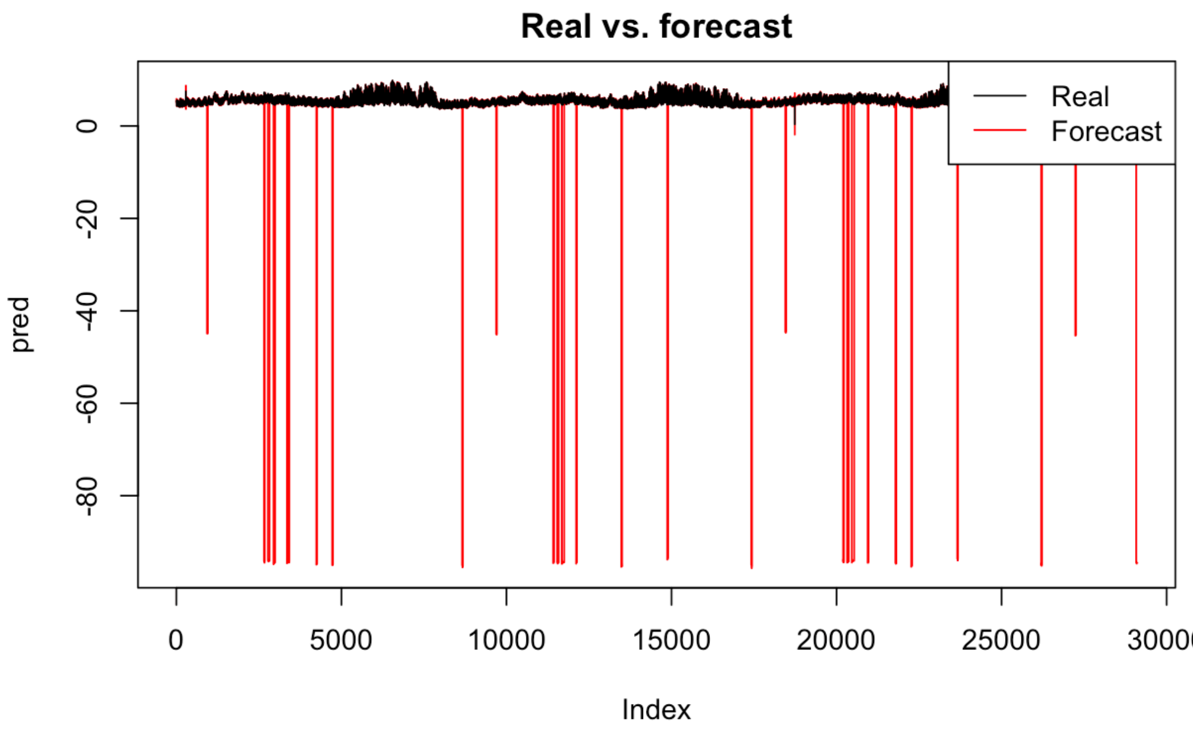
```
model <- Arima(ts, order = c(2, 1, 2), list(order = c(1, 1, 1), period = 7))

auxts <- ts
auxmodel <- model
errs <- c()
pred <- c()
perc <- c()
for (i in 1:nrow(df_test)) {
  p <- as.numeric(predict(auxmodel, newdata = auxts, n.ahead = 1)$pred)
  pred <- c(pred, p)
  errs <- c(errs, p - df_test$demand[i])
  perc <- c(perc, (p - df_test$demand[i]) / df_test$demand[i])
  auxts <- ts(c(auxts, df_test$demand[i]), frequency = 7)
  auxmodel <- Arima(auxts, model = auxmodel)
}
par(mfrow = c(1, 1))
plot(errs, type = 'l', main = 'Error in the forecast')

plot(pred, type = 'l', main = 'Real vs. forecast', col = 'red')
lines(df_test$demand)
legend('topright', c('Real', 'Forecast'), lty = 1, col = c('black', 'red'))

abserr <- mean(abs(errs)); abserr
percerr <- mean(abs(perc)) * 100; percerr
```



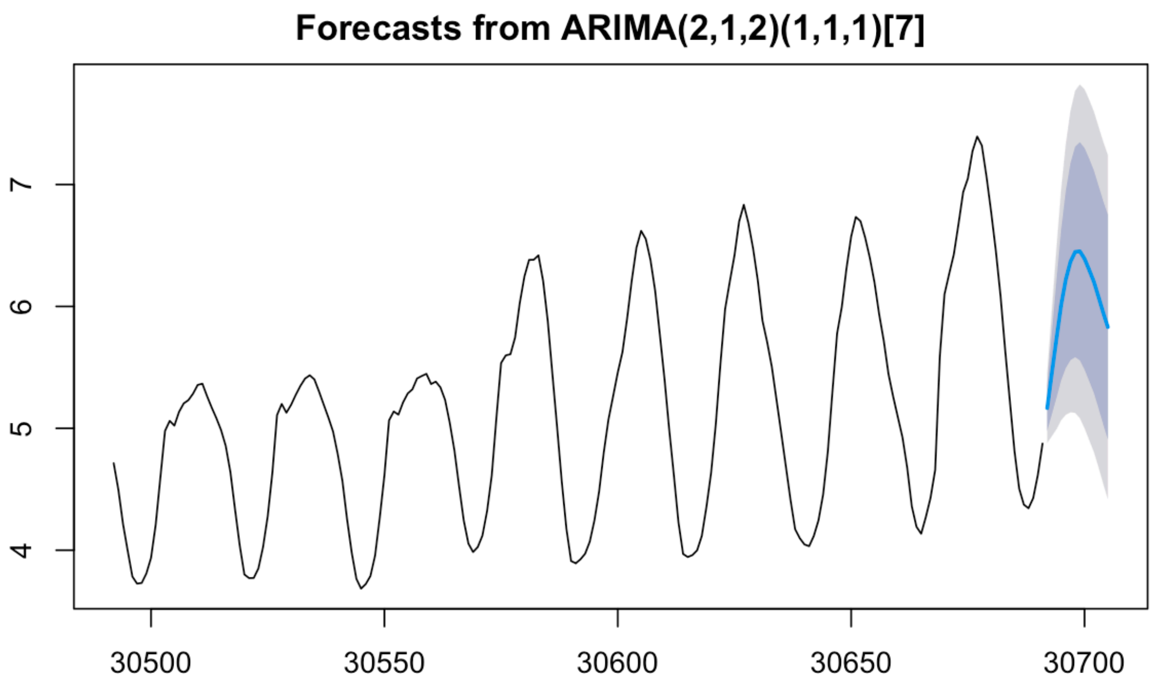


```r
specialday <- function(day) {
  correction = 0
  if (format(day, '%m%d') %in% c('0101', '0501', '0106', '0815', '1012', '1101', '1206', '1208', '1224', '1225', '1226', '1231'))
      correction = -100
  else if (format(day, '%m%d') %in% c('0319'))
    correction = -50
  # On Sunday, do not apply correction
  if (as.factor(strftime(day, format = '%A')) == 'Sunday')
    return(0)
  return(correction)
}

model <- Arima(ts, order = c(2, 1, 2), list(order = c(1, 1, 1), period = 7))
auxts <- ts
auxmodel <- model
errs <- c()
pred <- c()
perc <- c()
for (i in 1:nrow(df_test)) {
  p <- as.numeric(predict(auxmodel, newdata = auxts, n.ahead = 1)$pred)
  correction = specialday(df_test$date[i])
  pred <- c(pred, p + correction)
  errs <- c(errs, p + correction - df_test$demand[i])
  perc <- c(perc, (p + correction - df_test$demand[i]) / df_test$demand[i])
  if (!correction)
    auxts <- ts(c(auxts, df_test$demand[i]), frequency = 7)
  else
    auxts <- ts(c(auxts, p), frequency = 7)
  auxmodel <- Arima(auxts, model = auxmodel)
}
par(mfrow = c(1, 1))
plot(errs, type = 'l', main = 'Error in the forecast')

plot(pred, type = 'l', main = 'Real vs. forecast', col = 'red')
lines(df_test$demand)
legend('topright', c('Real', 'Forecast'), lty = 1, col = c('black', 'red'))

abserr <- mean(abs(errs))
percerr <- mean(abs(perc)) * 100
```


Error in the forecast
errs
0
-20
-40
-60
-80
-100
0
5000
10000
15000
20000
25000
30000
Index


Real vs. forecast
pred
Real
Forecast
0
-20
-40
-60
-80
0
5000
10000
15000
20000
25000
30000
Index

```
plot(forecast(Arima(tail(ts, 200), model = model)))
forc <- forecast(Arima(tail(ts, 200), model = model))
forc
```



	Point Forecast <dbl>	Lo 80 <dbl>	Hi 80 <dbl>	Lo 95 <dbl>	Hi 95 <dbl>
30692	5.165890	4.979434	5.352345	4.880730	5.451049
30693	5.455835	5.115914	5.795755	4.935971	5.975698
30694	5.739005	5.251479	6.226532	4.993398	6.484612
30695	6.013896	5.395027	6.632765	5.067418	6.960375
30696	6.222852	5.496046	6.949658	5.111298	7.334406
30697	6.367498	5.559142	7.175854	5.131225	7.603771
30698	6.447850	5.583875	7.311825	5.126514	7.769186
30699	6.452179	5.557245	7.347112	5.083496	7.820861
30700	6.391188	5.482446	7.299929	5.001387	7.780988
30701	6.299349	5.386895	7.211802	4.903871	7.694826
30702	6.202608	5.290222	7.114994	4.807234	7.597981
30703	6.077822	5.164857	6.990787	4.681563	7.474082
30704	5.949373	5.032957	6.865788	4.547836	7.350909
30705	5.830322	4.907722	6.752922	4.419327	7.241316

Conclusion

Given the model, its errors, and prediction, we can note several areas of adjustments we can consider for further improvement. The model and process utilized in forecasting the future demand for energy based on historical data succeed in demonstrating the ability of ARIMA to formalize a mathematically sound prediction model. However, the parameters used by the model require further investigation to identify possible areas of improvement. Furthermore, while the model considers the data provided, it fails to identify any additional possible predictors beyond the data. Further investigation would be needed to identify possible predictors, analyze their data, and test their associated hypotheses. Finally, it remains to be seen how accurate the prediction model is, which can shed some added light on the scale of improvement needed to enhance the forecasting power.

References

- https://towardsdatascience.com/energy-consumption-time-series-forecasting-with-python-and-lstm-deep-learning-model-7952e2f9a796
- https://towardsdatascience.com/part-2-time-series-analysis-predicting-hourly-energy-consumption-of-san-diego-ii-f09665796c9
- https://www.hindawi.com/journals/mpe/2020/4181045
- https://digitalcommons.montclair.edu/cgi/viewcontent.cgi?article=1679&context=etd
- https://rpubs.com/Peque/energy-demand-forecast
- https://en.wikipedia.org/wiki/KPSS_test