Predicting the tomorrow's hourly electricity consumption of Turkey

yunus emre erdogan

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

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

## Loading required package: ggplot2

##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## NOTE: Either Arial Narrow or Roboto Condensed fonts are required to use these themes.

##       Please use hrbrthemes::import_roboto_condensed() to install Roboto Condensed and

##       if Arial Narrow is not on your system, please see https://bit.ly/arialnarrow
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

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

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Loading required package: fma

## Loading required package: expsmooth

## Loading required package: lmtest

## Loading required package: tseries

##
## Attaching package: 'xts'

## The following objects are masked from 'package:data.table':
##
##     first, last

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

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v tibble  3.1.0     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
## v purrr   0.3.4
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x lubridate::as.difftime() masks base::as.difftime()
## x data.table::between()    masks dplyr::between()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks plotly::filter(), stats::filter()
## x xts::first()             masks data.table::first(), dplyr::first()
## x data.table::hour()       masks lubridate::hour()
## x lubridate::intersect()   masks base::intersect()
## x data.table::isoweek()    masks lubridate::isoweek()
## x dplyr::lag()             masks stats::lag()
## x xts::last()              masks data.table::last(), dplyr::last()
## x data.table::mday()       masks lubridate::mday()
## x data.table::minute()     masks lubridate::minute()
## x data.table::month()      masks lubridate::month()
## x data.table::quarter()    masks lubridate::quarter()
## x data.table::second()     masks lubridate::second()
## x lubridate::setdiff()     masks base::setdiff()
## x purrr::transpose()       masks data.table::transpose()
## x lubridate::union()       masks base::union()
## x data.table::wday()       masks lubridate::wday()
## x data.table::week()       masks lubridate::week()
## x data.table::yday()       masks lubridate::yday()
## x data.table::year()       masks lubridate::year()

## Loading required package: lattice

##
## Attaching package: 'caret'

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

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

##
## Attaching package: 'GGally'

## The following object is masked from 'package:fma':
##
##     pigs
```

Introduction and Preprocessing

Data Investigation

The assignment objective is to investigate electricity consumption data from [https://seffaflik.epias.com.tr/transparency/tuketim/gerceklesen-tuketim/gercek-zamanli-tuketim.xhtml] and to decompose it its trend-cycle,seasonal and random components using AR, MA and ARMA models and finally to forecast the electricity consumption between 6-20 Mayin Turkey. First things first , i read the data and applied some transformations :

```
data_path='C:/Users/yeerd/OneDrive/Masaüstü/ie360_hw3/cons.csv'
consumption=fread(data_path)
consumption[,datetime:=with(consumption, dmy(Date) + hm(Hour))]

consumption[,Date:=dmy(Date)]

consumption[,Hour:=as.numeric(hm(Hour))/3600]
consumption[,`Consumption (MWh)`:=as.numeric(gsub(",", "", `Consumption (MWh)`))]
head(consumption,25)
```

```
##           Date Hour Consumption (MWh)            datetime
##  1: 2016-01-01    0          26277.24 2016-01-01 00:00:00
##  2: 2016-01-01    1          24991.82 2016-01-01 01:00:00
##  3: 2016-01-01    2          23532.61 2016-01-01 02:00:00
##  4: 2016-01-01    3          22464.78 2016-01-01 03:00:00
##  5: 2016-01-01    4          22002.91 2016-01-01 04:00:00
##  6: 2016-01-01    5          21957.08 2016-01-01 05:00:00
##  7: 2016-01-01    6          22203.54 2016-01-01 06:00:00
##  8: 2016-01-01    7          21844.16 2016-01-01 07:00:00
##  9: 2016-01-01    8          23094.73 2016-01-01 08:00:00
## 10: 2016-01-01    9          25202.27 2016-01-01 09:00:00
## 11: 2016-01-01   10          27224.96 2016-01-01 10:00:00
## 12: 2016-01-01   11          28908.04 2016-01-01 11:00:00
## 13: 2016-01-01   12          28789.25 2016-01-01 12:00:00
## 14: 2016-01-01   13          29367.70 2016-01-01 13:00:00
## 15: 2016-01-01   14          29548.32 2016-01-01 14:00:00
## 16: 2016-01-01   15          29390.89 2016-01-01 15:00:00
## 17: 2016-01-01   16          30734.97 2016-01-01 16:00:00
## 18: 2016-01-01   17          32048.02 2016-01-01 17:00:00
## 19: 2016-01-01   18          31438.11 2016-01-01 18:00:00
## 20: 2016-01-01   19          30728.47 2016-01-01 19:00:00
## 21: 2016-01-01   20          30166.14 2016-01-01 20:00:00
## 22: 2016-01-01   21          29461.28 2016-01-01 21:00:00
## 23: 2016-01-01   22          29242.83 2016-01-01 22:00:00
## 24: 2016-01-01   23          28069.09 2016-01-01 23:00:00
## 25: 2016-01-02    0          26224.60 2016-01-02 00:00:00
##           Date Hour Consumption (MWh)            datetime
```

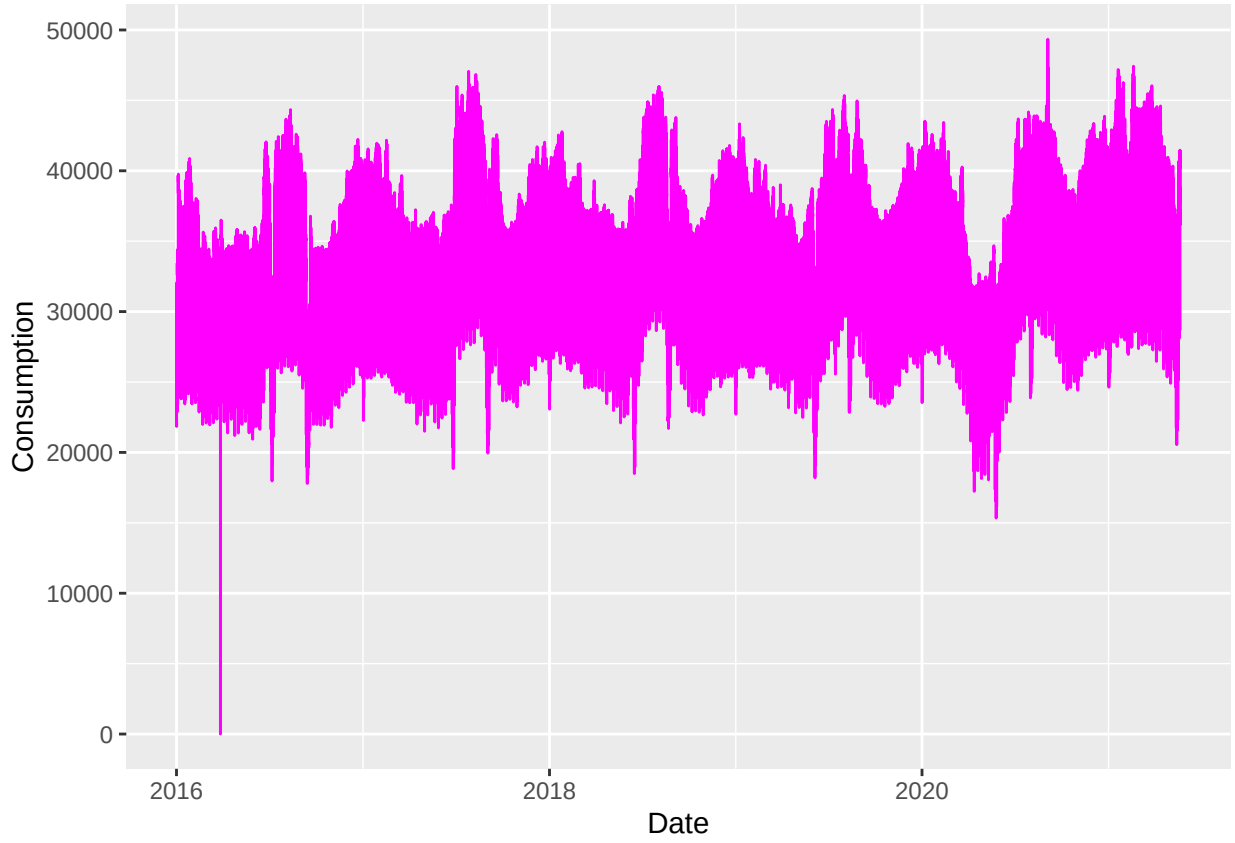
```
str(consumption)
```

```
## Classes 'data.table' and 'data.frame':   47208 obs. of  4 variables:
##  $ Date             : Date, format: "2016-01-01" "2016-01-01" ...
##  $ Hour             : num  0 1 2 3 4 5 6 7 8 9 ...
```

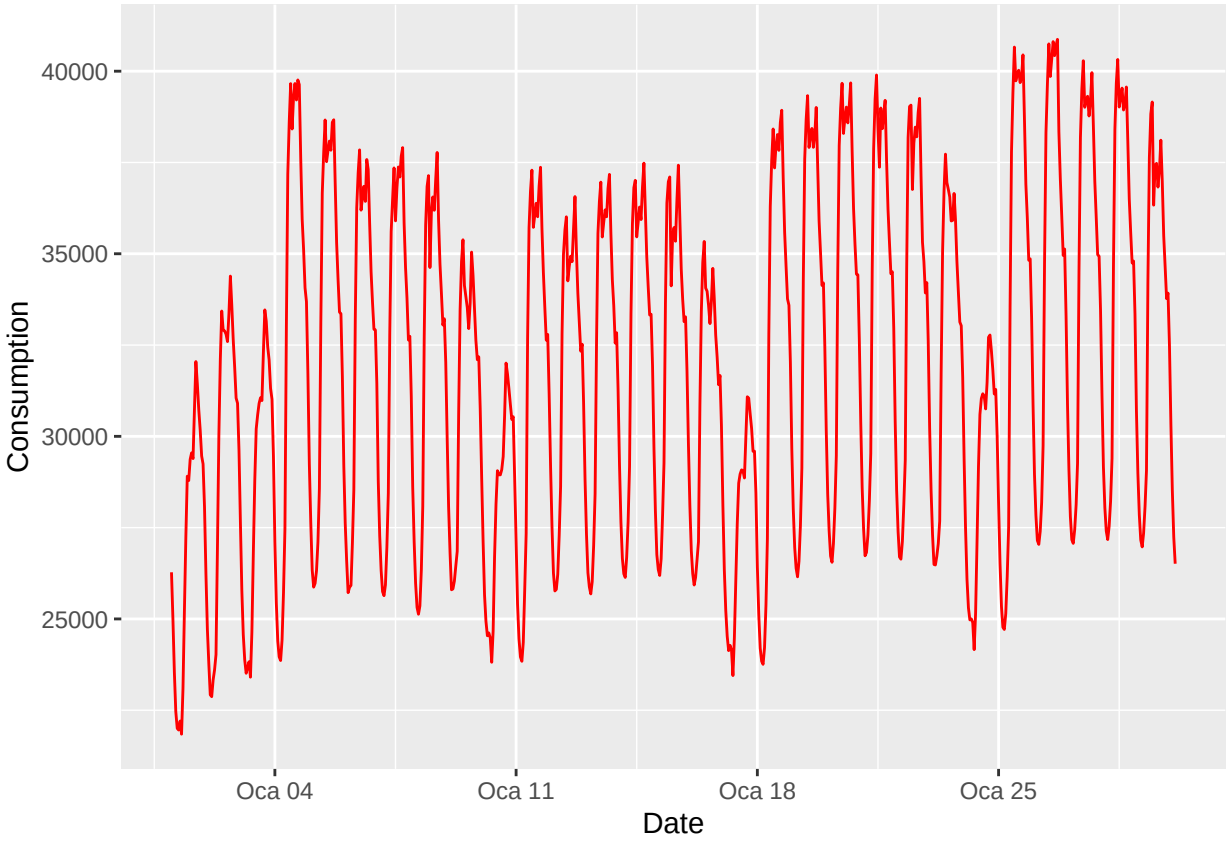
```
##  $ Consumption (MWh): num  26277 24992 23533 22465 22003 ...
##  $ datetime         : POSIXct, format: "2016-01-01 00:00:00" "2016-01-01 01:00:00" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

We converted characters to Date,num,num and POSIXct respectivelly. Here is the time-series plot of the data :

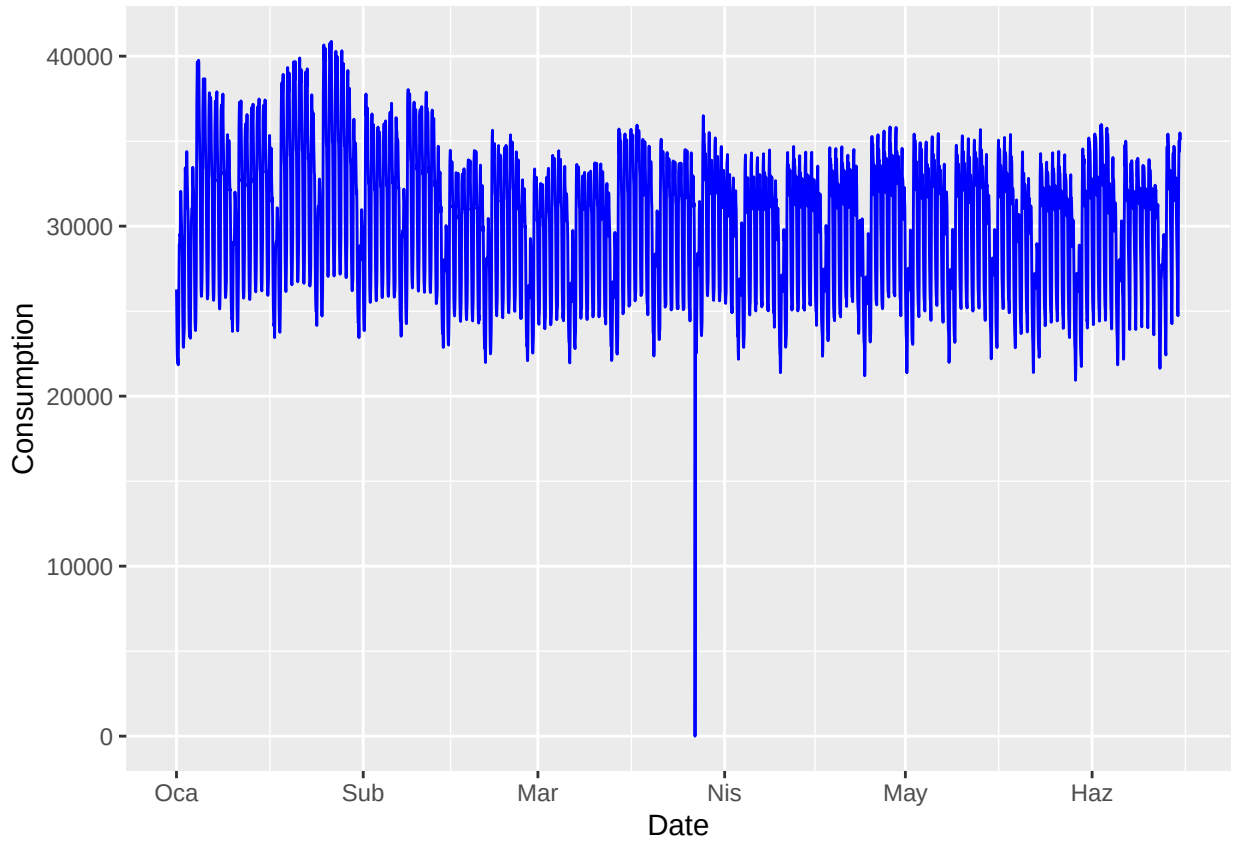
```
ggplot( consumption,aes(x=datetime, y=`Consumption (MWh)`)) +
  geom_line(color="magenta") +
  ylab("Consumption") +
  xlab("Date")
```



We can observe that there seems to be a weekly repeating pattern in data if we zoom in closer :



If we zoom out a bit , now we can monthly repeating patterns. Both weekly and monthly behaviour will be critical to choose the correct seasonality frequency.



Stationarity and Differencing

Before applying AR,MA and ARMA models , we need to check whether the given data is stationary or not. Nonstationary data is problematic and it violates our arima model assumptions.

```
## Loading required package: urca
```

```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 18 lags.
##
## Value of test-statistic is: 12.695
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

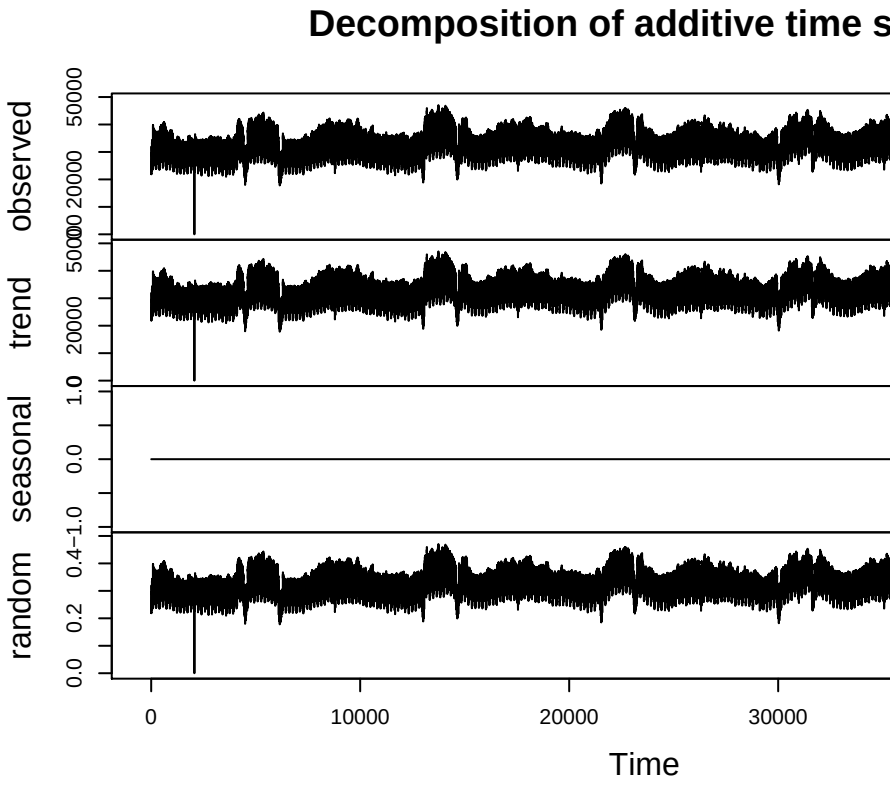
Raw data give us 12.695 as the test-statistic,which means that we should reject the null hypothesis stating data is stationary.Hence the data is not stationary.

```r
daily_consumption=consumption$`Consumption (MWh)`-shift(consumption$`Consumption (MWh)`,168)

unt_test=ur.kpss(daily_consumption)
summary(unt_test)
```

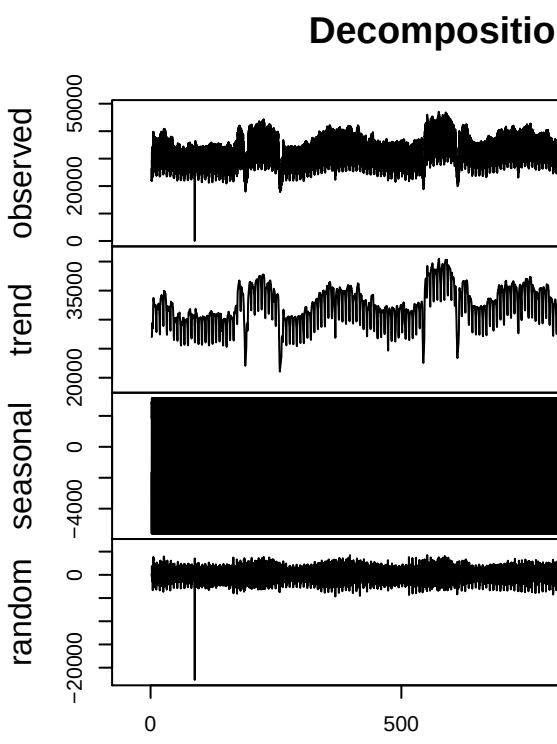
```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 18 lags.
##
## Value of test-statistic is: 0.052
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

If we apply 168-lag shifting (considering both weekly and daily seasonality), then we get 0.052 as the test-statistic which means that we can not reject the null hypothesis.Hence the lagged data is stationary.

Hourly Decomposition To hourly decompose i used frequency equaling 1 , here we can see there is no particular hourly seasonality when frequency is 1.

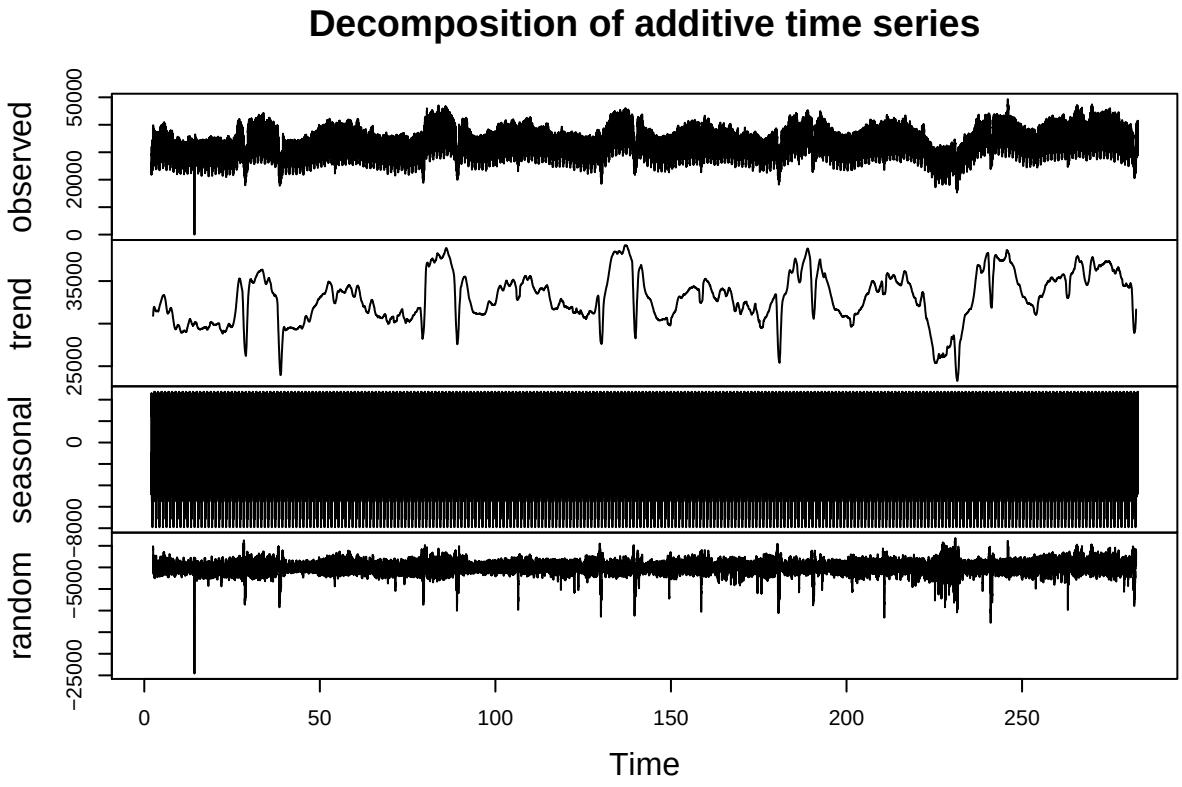


Daily Decomposition To daily decompose i used frequency equaling 24. Every 24-hour cycle we will no-



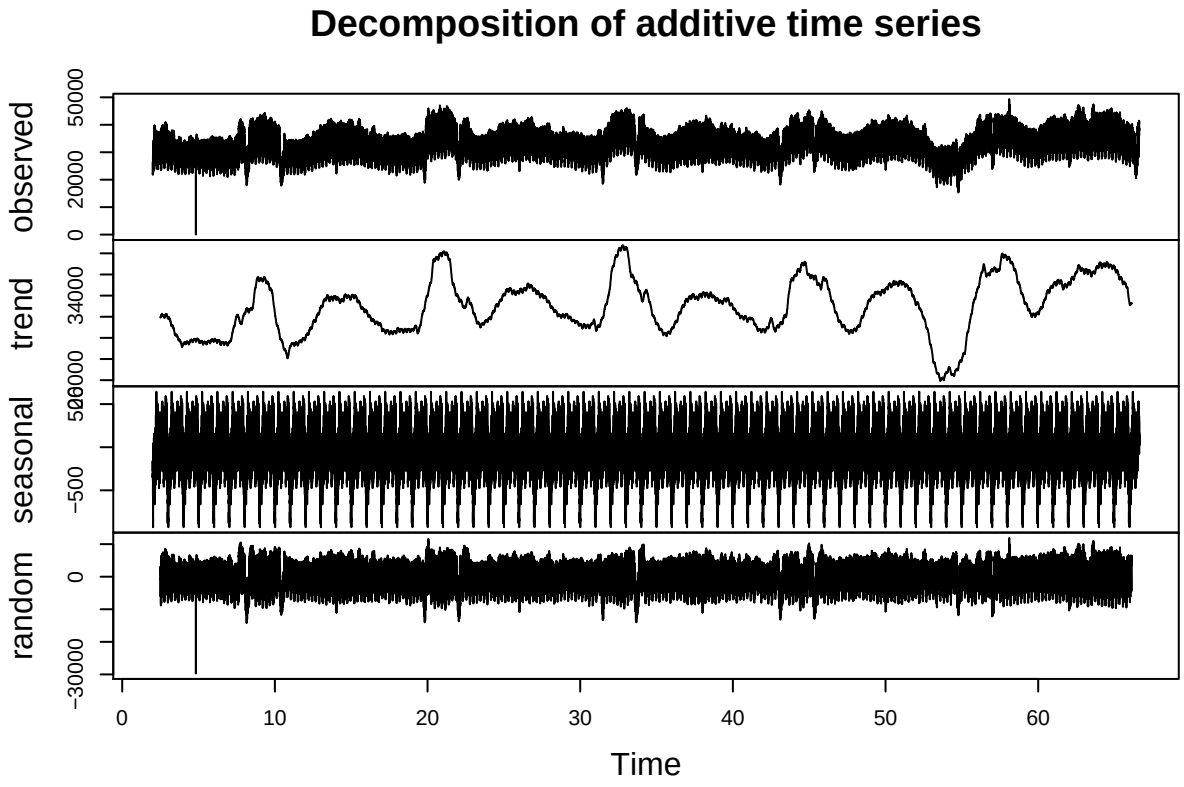
tice daily seasonality.As we can see there is a rapid seasonality in daily data.

Weekly Decomposition I used frequency equaling 168(24*7) since we had hourly raw data.The trend component got rid of its noise and the random component seems to be white but we need to look at it after-

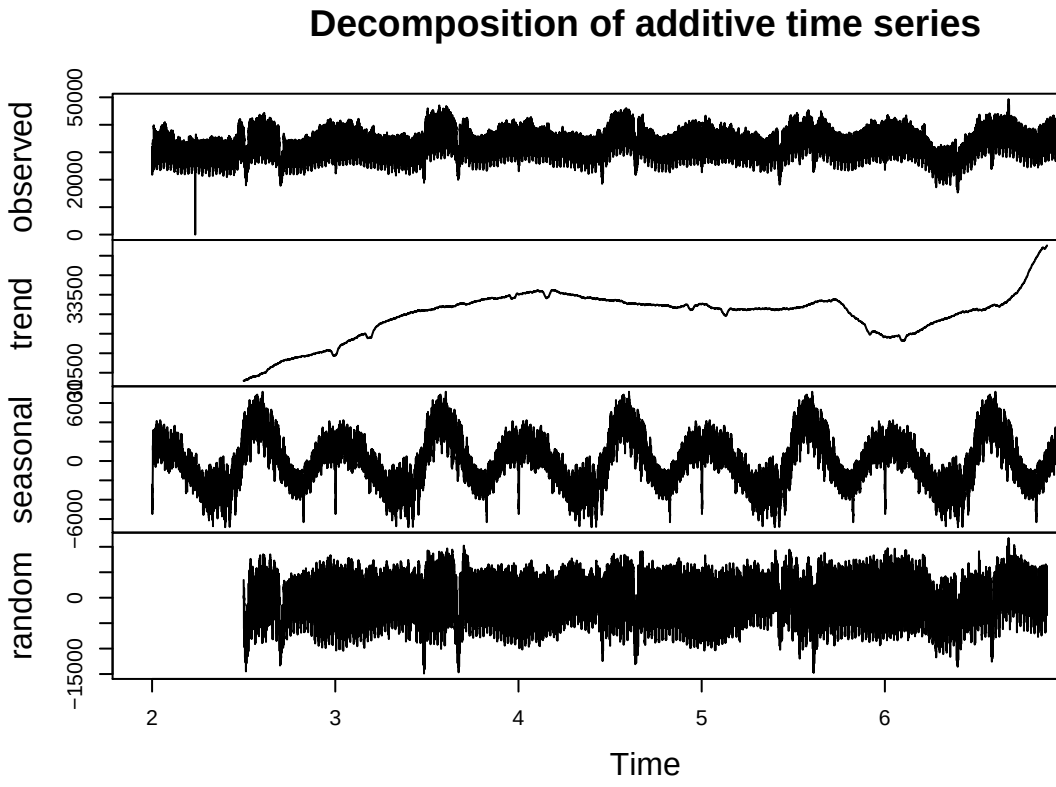


wards.

Monthly Decomposition To apply monthly decomposition i used frequency equaling 24*365.25/12 since every four year we have a leap year and number of days in a year should be 365+1/4 and we have to multiply it by 24 for each hourly data and divide it 12 to get monthly seasonality.Here we can see that trend component is still readable but we have lots of random noise which is overwhelming.



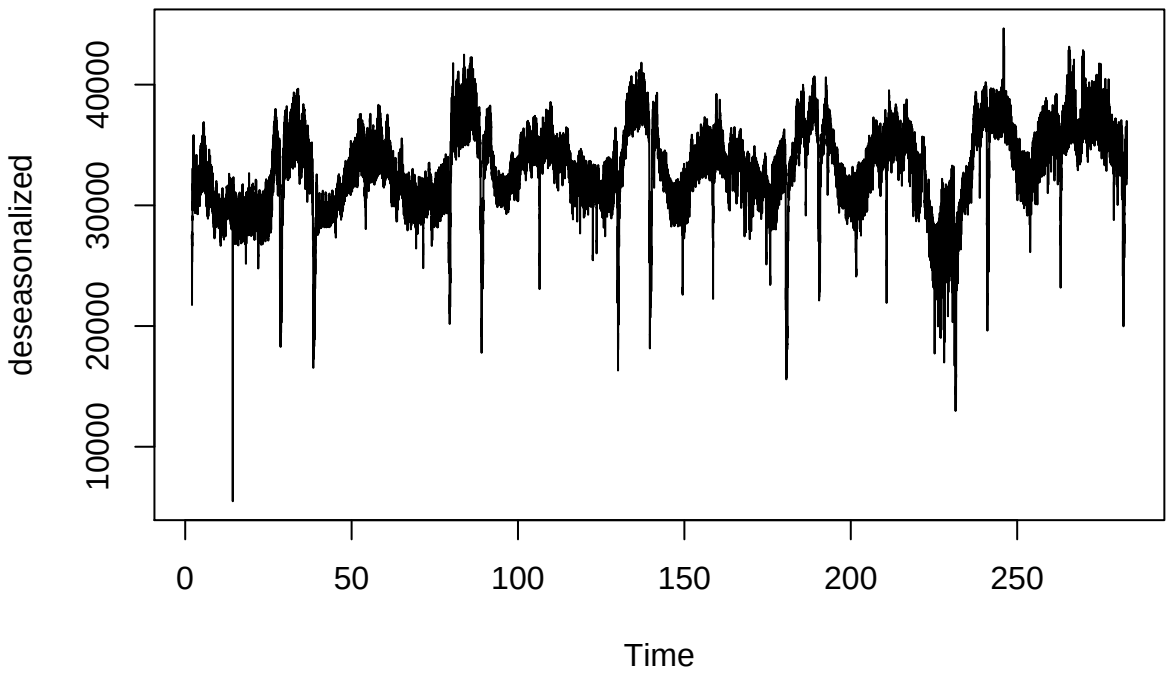
Yearly Decomposition Yearly decomposition is same as in monthly one except we dont divide the frequency by 12. Here we see a random-walkish pattern in trend-cycle component and the noise is highly accu-

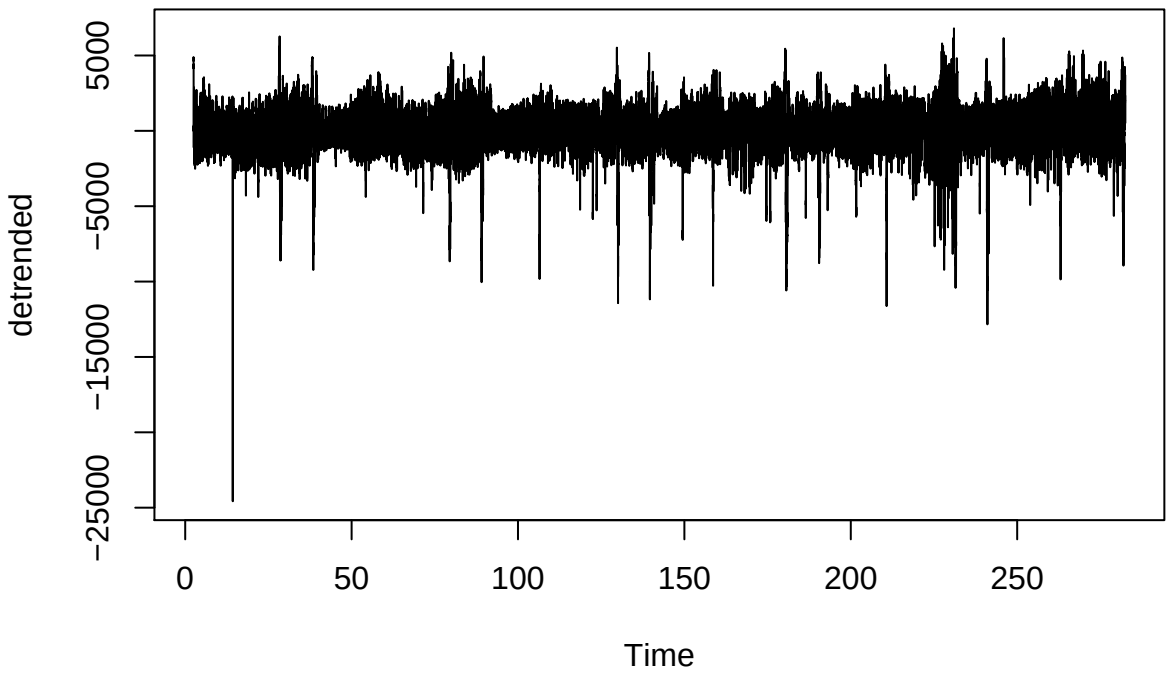


mulated in random component.

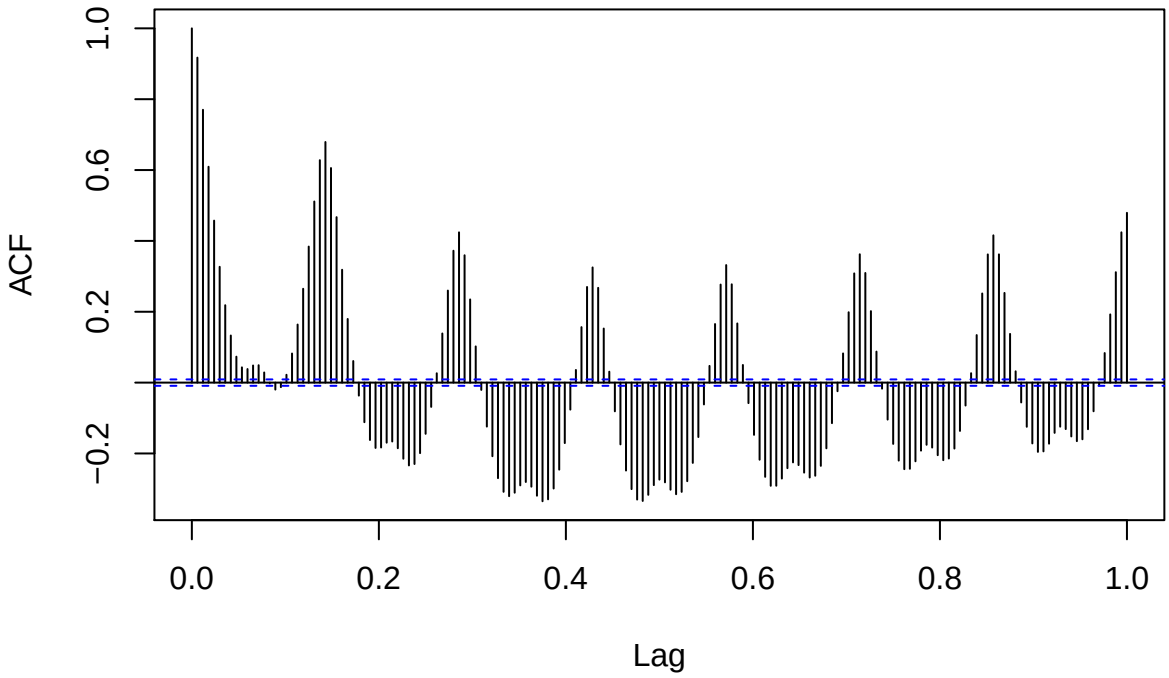
Deseasonalize and Detrend

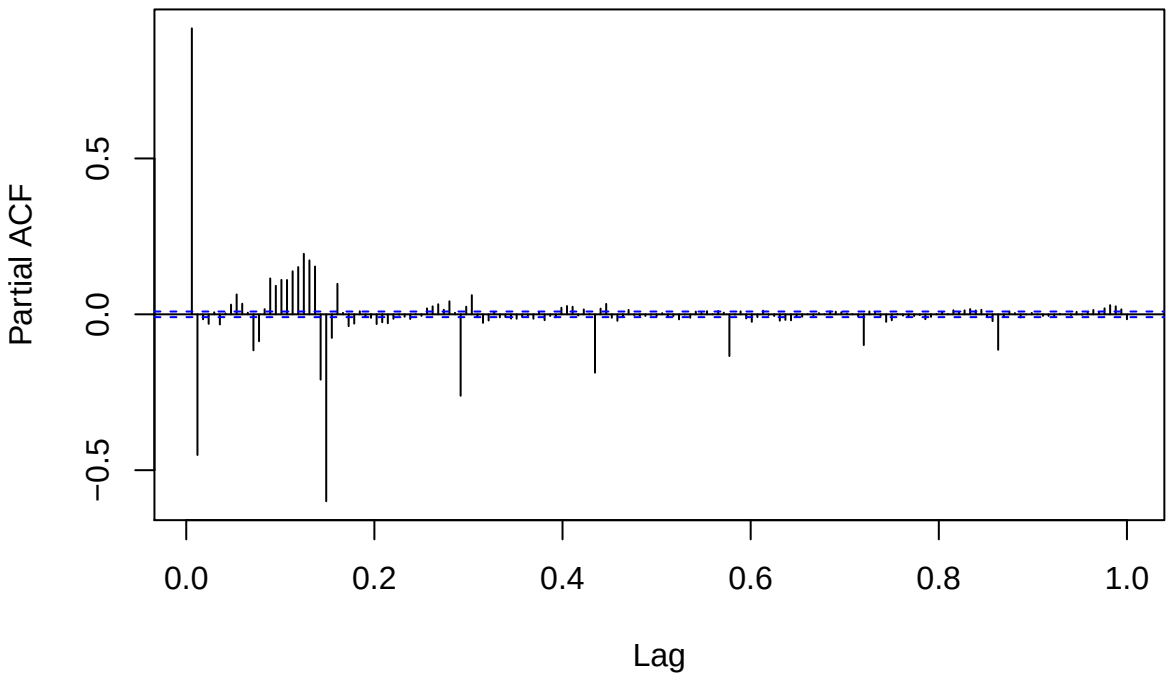
I chose 168 as seasonality frequency since both daily and weekly seasons seem to be effective and they dont have huge noise as in monthly and yearly ones.Here we see firstly the deseasonalized data and then we also extract the trend so that we get the random component.Random noise seems to be zero-mean and constant variance but there are several abnormal spikes which may be caused by some special days,events.


deseasonalized
10000
20000
30000
40000
0
50
100
150
200
250
Time



Before selecting p and q values in ARMA models , we need to investigate acf and pacf graphs.As we can see there is an exponentially decaying seasonality in the random component of decomposed data. This occurs due to not introducing differencing.When we substract raw data from its 168-lagged version, we get rid of this repetitive seasonality pattern in random component but it did not effect too much in my simulations, hence i continued with the original random component.


Series detrended
ACF
1.0
0.6
0.2
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
Lag



From acf graph , we may play with p=1 to 5 or 6 and q=1 or 2 but we should also try q = 1 to 5 since the random component still contains seasonality effects.

```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 18 lags.
##
## Value of test-statistic is: 0.0042
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

Reviewing the root test , we observe that the random component is stationary since it approves the null hypothesis saying that the data is stationary.Now we can continue with AR models.

AR,MA and ARMA models

AR Models

I tried p=1 to 5.Since we dont apply any differencing and the modelling is a non-seasonal arima ,we get huge AIC,BIC values ranging around 720-730k.

```
##
## Call:
## arima(x = detrended, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.9177    -1.0773
## s.e.  0.0018    33.0085
##
## sigma^2 estimated as 347441:  log likelihood = -366824.2,  aic = 733654.4
```

```
## [1] 733654.4
```

```
## [1] 733680.7
```

```
##
## Call:
## arima(x = detrended, order = c(2, 0, 0))
##
## Coefficients:
##          ar1      ar2  intercept
##       1.3324  -0.4521    -1.0783
## s.e.  0.0041   0.0041    20.2591
##
## sigma^2 estimated as 276455:  log likelihood = -361449,  aic = 722906
```

```
## [1] 722906
```

```
## [1] 722941.1
```

```
##
## Call:
## arima(x = detrended, order = c(3, 0, 0))
##
## Coefficients:
##          ar1      ar2      ar3  intercept
##       1.3254  -0.4314  -0.0155    -1.0787
## s.e.  0.0046   0.0074   0.0046    19.9511
##
## sigma^2 estimated as 276389:  log likelihood = -361443.4,  aic = 722896.7
```

```
## [1] 722896.7
```

```
## [1] 722940.5
```

```
##
## Call:
## arima(x = detrended, order = c(4, 0, 0))
##
## Coefficients:
##          ar1      ar2     ar3      ar4  intercept
##       1.3249  -0.4444  0.0247  -0.0304    -1.0779
```

```
## s.e.  0.0046   0.0077  0.0077   0.0046    19.3530
##
## sigma^2 estimated as 276133:  log likelihood = -361421.6,  aic = 722855.1
```

```
## [1] 722855.1
```

```
## [1] 722907.7
```

```
##
## Call:
## arima(x = detrended, order = c(5, 0, 0))
##
## Coefficients:
##          ar1      ar2     ar3      ar4     ar5  intercept
##       1.3251  -0.4447  0.0278  -0.0393  0.0067    -1.0770
## s.e.  0.0046   0.0077  0.0079   0.0077  0.0046    19.4726
##
## sigma^2 estimated as 276121:  log likelihood = -361420.5,  aic = 722855.1
```

```
## [1] 722855.1
```

```
## [1] 722916.4
```

Here i chose p=3, eventhough p=5,6 or 7 were giving better results , it was not marginally effecive. To save performance in terms of time , i chose p=3 and it will be sufficient enough.

MA models

Again applying q=1 to 5, we observe significant changes in AIC values but not much. I increased q=5 up to 10 , it improved but again increase margin was not satisfactory, hence i chose 5 as q parameter.

```
##
## Call:
## arima(x = detrended, order = c(0, 0, 1))
##
## Coefficients:
##          ma1  intercept
##       0.8539    -1.0796
## s.e.  0.0017     7.5591
##
## sigma^2 estimated as 782071:  log likelihood = -385906.9,  aic = 771819.8
```

```
## [1] 771819.8
```

```
## [1] 771846.1
```

```
##
## Call:
## arima(x = detrended, order = c(0, 0, 2))
##
## Coefficients:
```

```
##          ma1     ma2  intercept
##       1.2026  0.6142    -1.0790
## s.e.  0.0037  0.0029     8.8633
##
## sigma^2 estimated as 465785:  log likelihood = -373718.6,  aic = 747445.1
```

```
## [1] 747445.1
```

```
## [1] 747480.2
```

```
##
## Call:
## arima(x = detrended, order = c(0, 0, 3))
##
## Coefficients:
##          ma1     ma2     ma3  intercept
##       1.2872  1.0099  0.4779    -1.0788
## s.e.  0.0043  0.0046  0.0034    10.3582
##
## sigma^2 estimated as 354186:  log likelihood = -367276.5,  aic = 734563
```

```
## [1] 734563
```

```
## [1] 734606.8
```

```
##
## Call:
## arima(x = detrended, order = c(0, 0, 4))
##
## Coefficients:
##          ma1     ma2     ma3     ma4  intercept
##       1.3190  1.1665  0.7768  0.3124    -1.0784
## s.e.  0.0049  0.0066  0.0050  0.0041    11.8271
##
## sigma^2 estimated as 314484:  log likelihood = -364480.3,  aic = 728972.5
```

```
## [1] 728972.5
```

```
## [1] 729025.1
```

```
##
## Call:
## arima(x = detrended, order = c(0, 0, 5))
##
## Coefficients:
##          ma1     ma2     ma3     ma4     ma5  intercept
##       1.3303  1.2636  1.0113  0.6190  0.2464    -1.0771
## s.e.  0.0045  0.0069  0.0071  0.0061  0.0040    13.6351
##
## sigma^2 estimated as 292278:  log likelihood = -362758,  aic = 725530.1
```

```
## [1] 725530.1
```

```
## [1] 725591.4
```

ARMA Model

Now our model is arma with p=3 and q=5. After getting the proper model i obtained the predicted random component and then i applied additive ts model to get back estimated comsumption values.Here is its visualization :

```
model <- arima(detrended, order=c(3,0,5))
print(model)
```

```
##
## Call:
## arima(x = detrended, order = c(3, 0, 5))
##
## Coefficients:
##          ar1     ar2      ar3     ma1      ma2      ma3      ma4      ma5
##       0.7094  0.5861  -0.4581  0.6170  -0.2148  -0.0729  -0.0367  -0.0076
## s.e.  0.0401  0.0528   0.0258  0.0404   0.0325   0.0189   0.0128   0.0078
##       intercept
##         -0.6667
## s.e.    19.1362
##
## sigma^2 estimated as 275814:  log likelihood = -361394.4,  aic = 722808.8
```

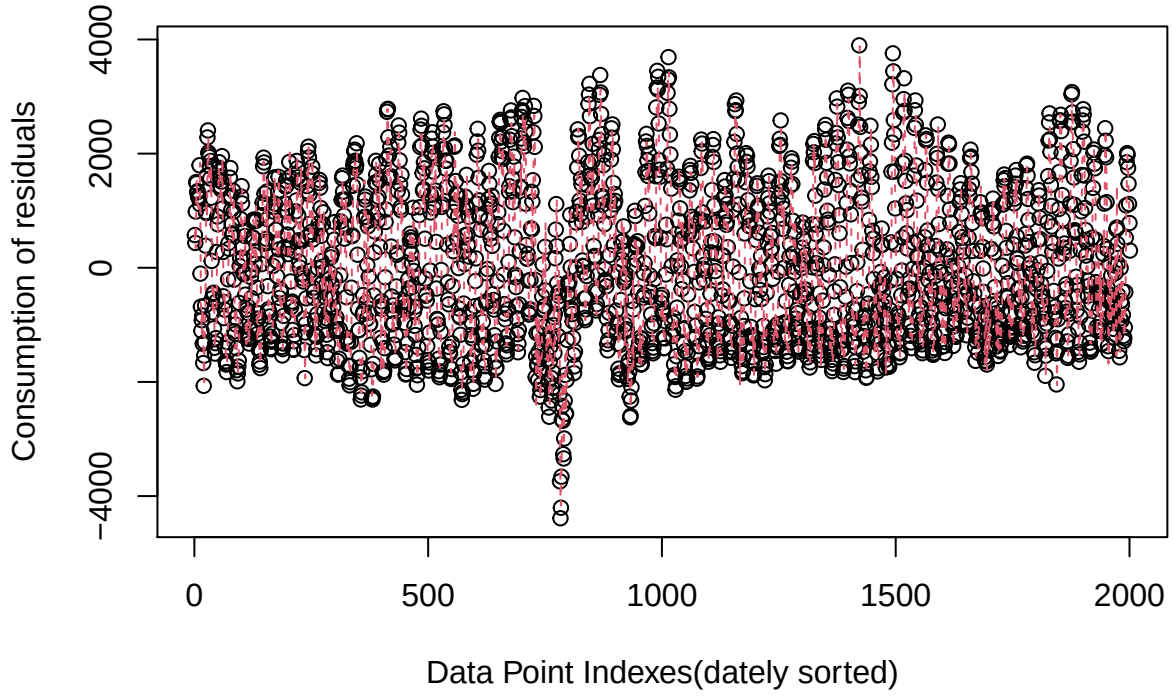
```
AIC(model)
```

```
## [1] 722808.8
```

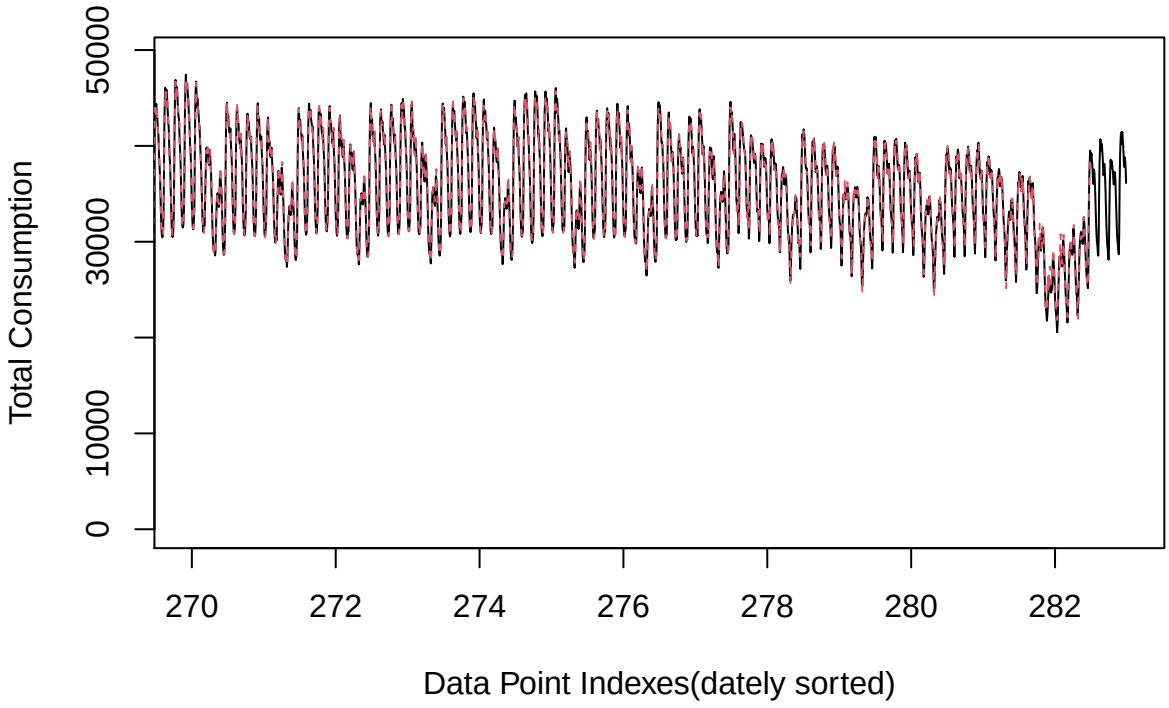
```
BIC(model)
```

```
## [1] 722896.4
```

```
model_fitted <- detrended - residuals(model)
model_fitted_transformed <- model_fitted+weekly_dec$trend+weekly_dec$seasonal

plot(detrended[8000:10000], xlab = "Data Point Indexes(dately sorted)", ylab = "Consumption of residuals
points(model_fitted[8000:10000], type = "l", col = 2, lty = 2)
```




```
plot(weekly, xlab = "Data Point Indexes(dately sorted)", ylab = "Total Consumption",xlim = c(270,283))
points(model_fitted_transformed, type = "l", col = 2, lty = 2,xlim = c(270,283))
```



We can see that residuals resemble each other and we nearly got perfect fitting in training data. Data seem to be overfitted but its not our objective today.

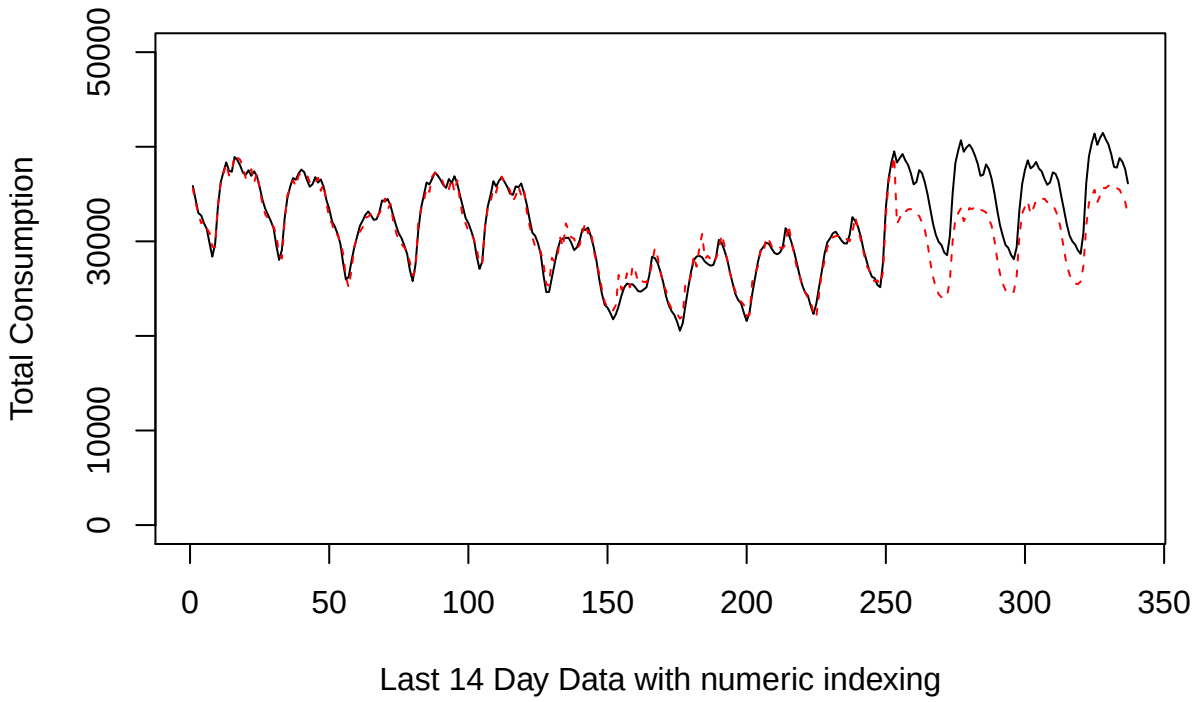
Forecasting

Before applying forecasting, i need to point out something , at start i used all data as training data from 6 May to 20 May , hence the test data is vanished in my approach but not quite. We know that there is always NA values at the edges by 168/2=84 due to seasonality approach ; therefore i predicted last 84-hour NA values in data but i actually modelled and predicted the remaining part of last 336(14*24)-84=252 values since the residuals of model gave me the coefficients of remaining 336-84= 252-hour values in 14 day span. I also replaced the NA values in first 84-hour values by the mean of next 84-hour values. Then i applied predict function.

```
#forecasting
Residuals_fitted = residuals((model))
consumption[,fitted_residuals:=Residuals_fitted]
consumption[,modelled_consumption:=model_fitted_transformed]
first_na=mean(consumption$modelled_consumption[85:168])
first_na_res=mean(consumption$fitted_residuals[85:168])
consumption$modelled_consumption[1:84]=first_na
consumption$fitted_residuals[1:84]=first_na_res

test_data = consumption$`Consumption (MWh)`[46849:47208]

res_tes=predict(model,n.ahead = 84)$pred
```

Here i obtained the last trend and last season values from decomposed data since we need to return in additive ts model format. Since last 84 values are missing, i took next 84 non-NA values from trend and season components. Then i add them up with additive manner.

```
last_trend = tail(weekly_dec$trend[!is.na(weekly_dec$trend)],84)
last_season= tail(weekly_dec$seasonal[!is.na(weekly_dec$seasonal)],84)
comb = res_tes+last_trend+last_season

consumption$modelled_consumption[47125:47208]=comb

plot(consumption$`Consumption (MWh)`[46872:47208], xlab = "Last 14 Day Data with numeric indexing", ylab
points(consumption$modelled_consumption[46872:47208], type = "l", col = "red", lty = 2,ylim=range(0:500
```



Here we see the last 14-day span electricity consumption data and its predictions.Since i used test data inside the training data , we get nearly perfect estimations up to last 84 hour.After not getting current updated real data , the model struggled a bit to mimic and predict the remaining hours.I can name couple of reasons why we got relatively large errors: 1) We used ARMA model not ARIMA , hence not introducing integrated model may give us distortions at the end. 2) Differencing is not applied , even we would have been applied it would not affect too much, i tried it aswell. No reason to not mention about it , nevertheless. 3) we used non-seasonal arma model , SARIMA model would perform better after seeing acf and pacf graphs of random component. 4) We did not add any external regressor , hence ARIMAX or SARIMA with regressor would overperform our model.

WMAPE Evaluation

To grasp a good overall performance notion, i used weighted mean percentage error as an evaluation method.Here i use this site : [https://ibf.org/knowledge/glossary/weighted-mean-absolute-percentage-error-wmape-299] and at the end i took average WMAPE and it gave me 4.31 error percentage, it seems like the model did a good job but it is not fully correct. Only last 84-hour WAPE gave me around 12 percent error which is quite a lot.

```
err=abs(consumption$`Consumption (MWh)`[46872:47208]-consumption$modelled_consumption[46872:47208])
percentage=err/consumption$`Consumption (MWh)`[46872:47208]
wape=(sum(percentage)/360)*100
```

Conclusion

I investigated the electricity consumption data and decomposed it its components then i took the random component to further apply ar,ma and arma models. The performance of arma model is better between those three since we have power of lags and errors at the same time in ARMA model. Autocorrelation is one of the main issues even if we use arma model, and including test data in the training data performs much better . Forecasting was not a direct forecasting , i mean that i forecasted missing values to get forecasted estimated data , hence i doubled the possibility of error.Nevertheless , the results are promising and i look forward to apply more advance and sofisticated methods. Stay tuned !