Sokrati Recruitment Assignment

Disha

Sat Apr 16, 2016

Problem Statement: The input file contains measurement of some unspecified metric by date and hour of the day. The goal is to develop a model to predict the metric.

Strategy:

Time Series Forecasting:

1. Visualized/explored the series and transformed it
2. Made the series stationary and handled outliers
3. Found optimal parameters using ACF, PACF plots
4. Built SARIMA models
5. Chose model with lowest AIC values
6. Model performance and further tweaks

Supervised Learning:

1. Created new variables – day of the week, month, year, difference from Jan 1,00.
2. Removed date variable
3. Applied supervised learning algorithms and
4. Chose xgboost to be performing the best.
5. Model performance

Observations:

Is there a pattern in the value of the output variable? What is the pattern?

- *Yes, there was "daily" pattern. Also, the values peaked during the day.*

Does the particular pattern make it easy to model the problem or difficult? Why?

- *No, pattern had to be handled while modeling.*

Which parts of the code helped you determine the nature of the pattern?

- *Visualizing the series through plots*

Why did you choose a particular model for forecasting the values?

- *Xgboost performed the best in comparison to other models and was faster in processing.*

Why did you use a particular transformation of the input?

- *I applied log to the series, and the values were highly skewed. So, I used box cox transformations.*

```r
rm(list=ls())
options(scipen=999)

#set working directory
setwd("~/Personal/Assign")

################################################################
#              Invoke required Library                        #
################################################################
require("tseries")||install.packages("tseries");library("tseries")

## Loading required package: tseries

## [1] TRUE

require("timeSeries")||install.packages("timeSeries");library("timeSeries")

## Loading required package: timeSeries

## Loading required package: timeDate

## [1] TRUE

require("forecast")||install.packages("forecast");library("forecast")

## Loading required package: forecast

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following object is masked from 'package:timeSeries':
##
##     time<-

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

## This is forecast 7.0

## [1] TRUE

require("xgboost")||install.packages("xgboost");library("xgboost")

## Loading required package: xgboost

## [1] TRUE

require("ggplot2")||install.packages("ggplot2");library("ggplot2")

## Loading required package: ggplot2
```

```
## [1] TRUE
```

```r
require("lubridate")||install.packages("lubridate");library("lubridate")
```

```
## Loading required package: lubridate
```

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

```
## The following object is masked from 'package:base':
##
##     date
```

```
## [1] TRUE
```

```r
require("Matrix")||install.packages("Matrix");library("Matrix")
```

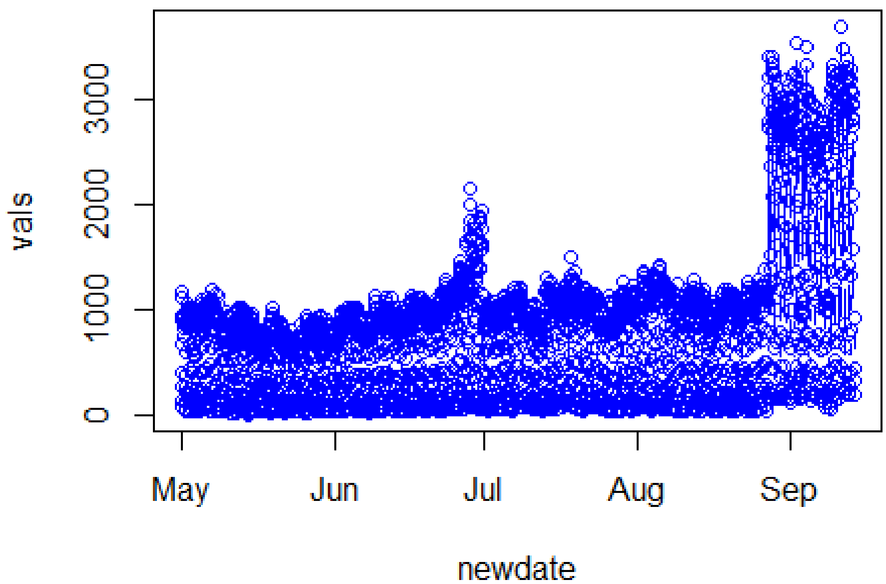
```
## Loading required package: Matrix
```

```
## [1] TRUE
```

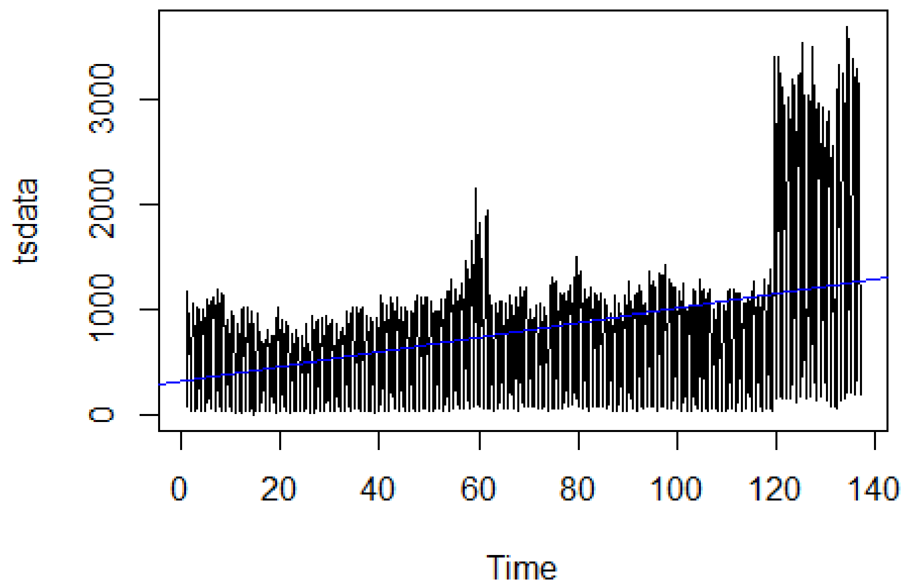
```r
#read file
data=read.delim("data_scientist_assignment.tsv",header=TRUE)
attach(data)

#join data and hour column to plot
data$newdate <- with(data, as.POSIXct(paste(date, hr_of_day), format="%Y-%m-%
d %H"))
head(data)
```

```
##         date hr_of_day vals             newdate
## 1 2014-05-01         0   72 2014-05-01 00:00:00
## 2 2014-05-01         1  127 2014-05-01 01:00:00
## 3 2014-05-01         2  277 2014-05-01 02:00:00
## 4 2014-05-01         3  411 2014-05-01 03:00:00
## 5 2014-05-01         4  666 2014-05-01 04:00:00
## 6 2014-05-01         5  912 2014-05-01 05:00:00
```

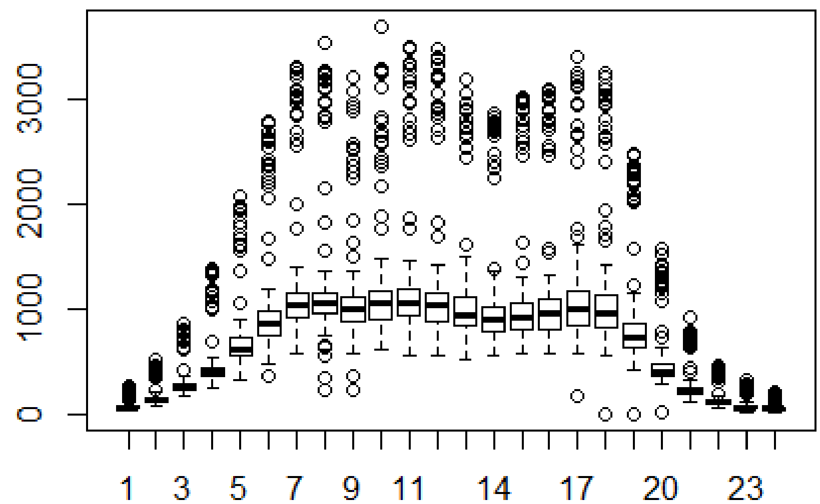
```r
#plot data
plot(vals ~ newdate, data=data, type="b",  col="blue")
```



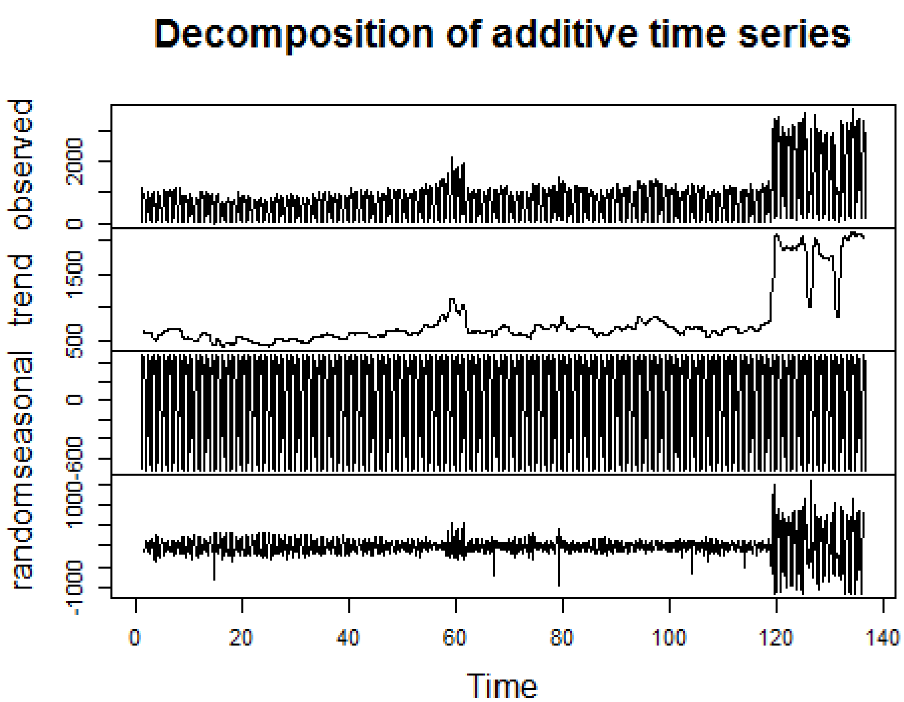

```
# ggplot(data, aes(newdate, vals)) +
#   geom_point() + geom_line()

#convert data to time series
tsdata=ts(data$vals,frequency=24)
plot(tsdata)
abline(reg=lm(tsdata~time(tsdata)),col="blue")  #add regression line
```




```
#summarize
summary(tsdata)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0   254.0   775.5   797.0  1027.0  3695.0

boxplot(tsdata~cycle(tsdata))
```

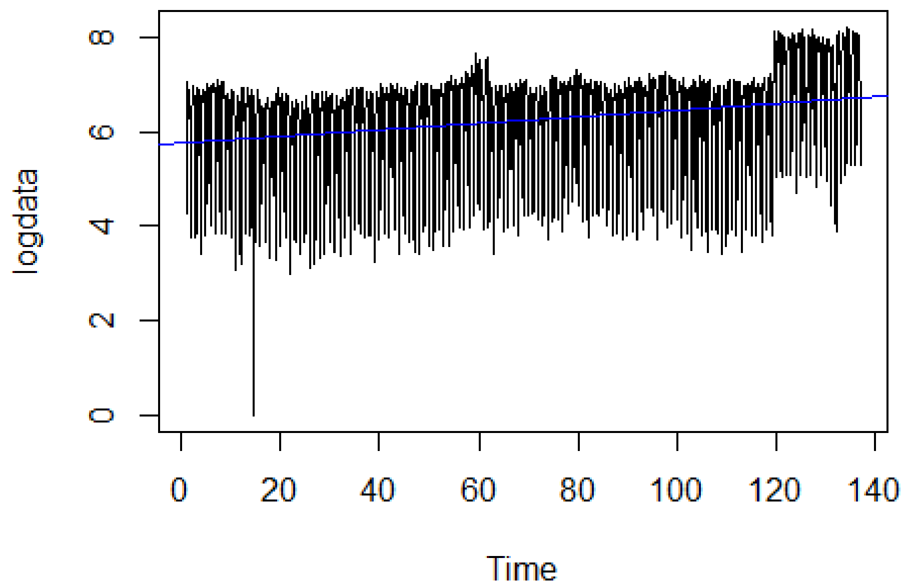



```r
plot(decompose(tsdata))
```




```r
#applying transformations
logdata=log(tsdata)
```

```r
plot(logdata)
abline(reg=lm(logdata~time(logdata)),col="blue")  #add regression line
```



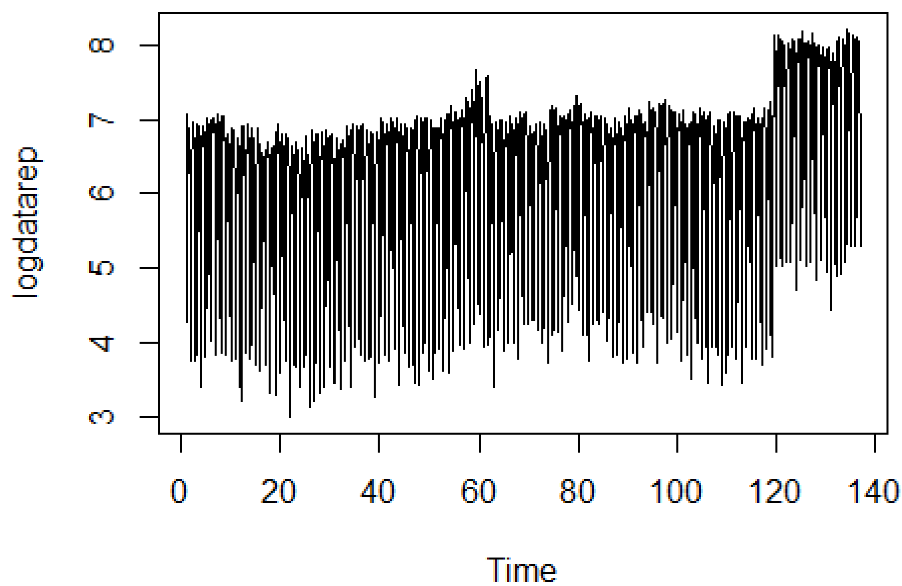

```r
#summarize
summary(logdata)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   5.537   6.654   6.228   6.934   8.215
```

```r
#check and replace outliers
(tsout=tsoutliers(logdata,iterate=24))
```

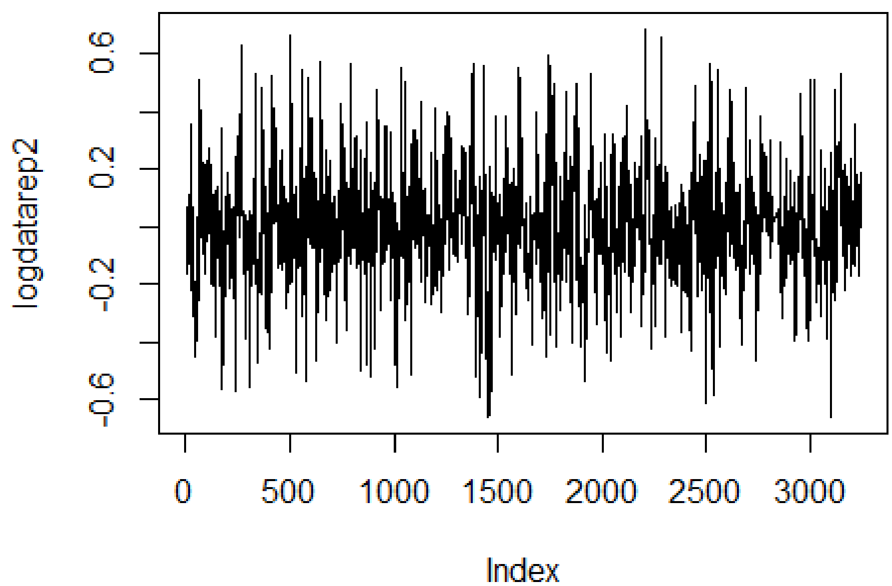
```
## $index
##  [1]  240  329  330  331  332  599 1592 1593 1880 1881 2481 2519 2718 2849
## [15] 2852 2853 2854 2857 3003 3004 3005 3006 3007 3008 3009 3010 3011 3012
## [29] 3013 3014 3014 3015 3128 3129 3130 3131 3132 3133 3134 3135 3136 3137
## [43] 3138 3139 3140 3141 3142 3143 3144 3145 3146 3147 3148
##
## $replacements
##  [1] 3.770704 6.535001 6.411456 6.048524 5.428406 3.678908 6.708429
##  [8] 6.551026 6.797759 6.872728 6.860881 3.613011 6.778815 8.119824
## [15] 7.206495 6.514478 5.805506 5.245971 6.594734 7.067189 7.503844
## [22] 7.825367 8.024186 8.055249 7.844593 7.896204 8.050302 8.049513
## [29] 7.961742 7.888576 7.888576 7.990307 7.880823 7.768186 7.812941
## [36] 7.865448 7.860553 7.754675 7.686634 7.741566 7.762117 7.765354
## [43] 7.748324 7.490677 6.954890 6.309994 5.795902 5.233290 4.908768
## [50] 5.089946 5.718969 6.329602 6.830544
```

```r
logdatarep=logdata
logdatarep[tsout$index]=tsout$replacements
plot(logdatarep,type="l")
```




```r
summary(logdatarep)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.996   5.552   6.656   6.245   6.940   8.215

#differencing
logdatas=logdatarep[25:3264]-logdatarep[25:3264-24]

#outliers replace - 2
(tsout2=tsoutliers(logdatas,iterate=24))

## $index
##  [1]  263  480 1376 1458 1488 2207 2256 2493 2821 2822 2823 2824 2825 2826
## [15] 2827 2828 2829 2830 2831 2832 2833 2834 2835 2836 2837 2838 2839 2840
## [29] 2841 2842 2843 2844 2858 2882
##
## $replacements
##  [1]  0.45800417 -0.11497480  0.50719921 -0.61076144  0.19941603
##  [6]  0.64538391 -0.09680094 -0.39618282  0.01851545  0.02020965
## [11]  0.02190385  0.02359805  0.02529225  0.02698644  0.02868064
## [16]  0.03037484  0.03206904  0.03376324  0.03545743  0.03715163
## [21]  0.03884583  0.04054003  0.04223423  0.04392842  0.04562262
## [26]  0.04731682  0.04901102  0.05070522  0.05239942  0.05409361
## [31]  0.05578781  0.05748201 -0.17048911  0.12639699
```

```r
logdatarep2=logdatas
logdatarep2[tsout2$index]=tsout2$replacements
plot(logdatarep2,type="l")
```




```r
summary(logdatarep2)
```

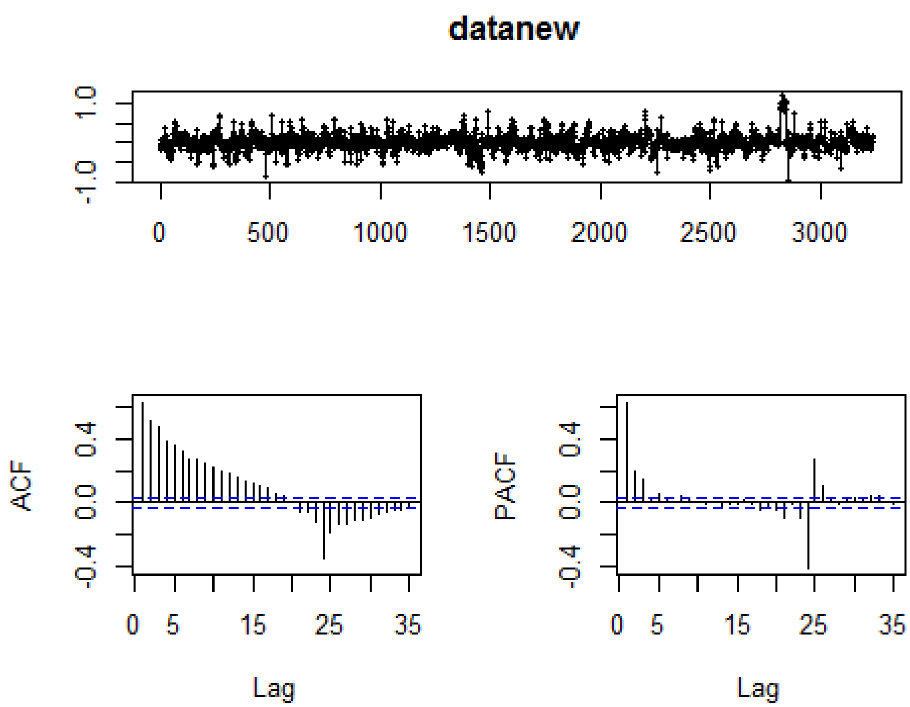
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
## -0.662900 -0.096580  0.000000  0.002368  0.100300  0.688100
```

```r
#final data
datanew = logdatas
rm(logdatas)
rm(logdatarep2)
rm(logdatarep)
rm(logdata)

#Augmented Dickey-Fuller Test
adf.test(datanew, alternative="stationary", k=0)
```

```
## Warning in adf.test(datanew, alternative = "stationary", k = 0): p-value
## smaller than printed p-value
```

```
## 
##  Augmented Dickey-Fuller Test
## 
## data:  datanew
## Dickey-Fuller = -27.236, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

```
#Plot ACF/PACF and find optimal parameters
tsdisplay(datanew)
```



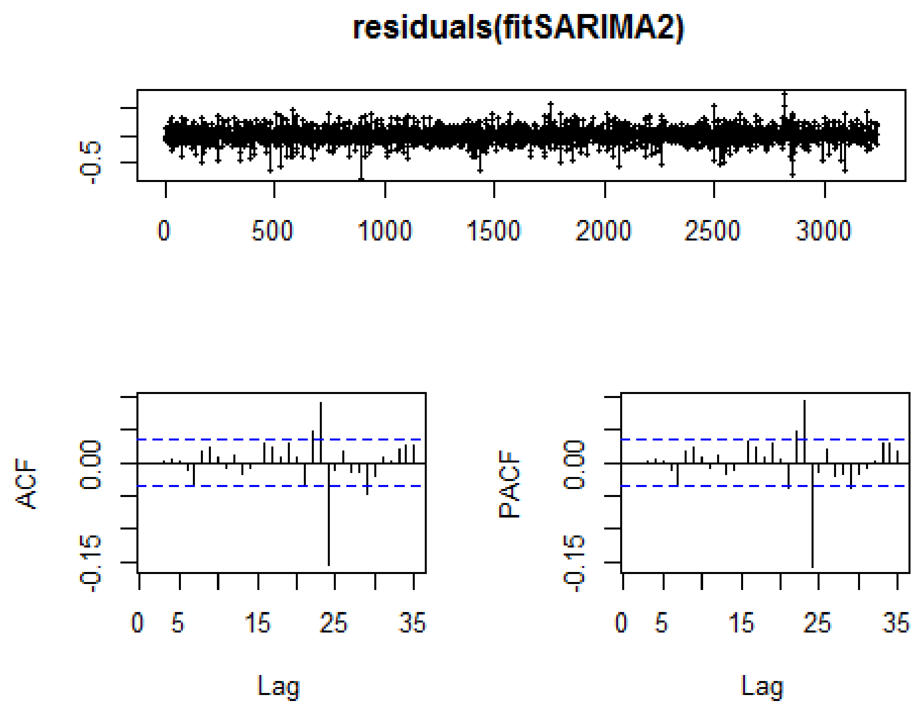

```
#decomposing
# plot(decompose(datanew))
# fitstl=stl(datanew, t.window = NULL, s.window=24, robust=TRUE)
# plot(fitstl)

#Build Arima model
(findbest <- auto.arima(datanew)) #ARIMA(2,0,2)

## Series: datanew
## ARIMA(2,0,2) with zero mean
##
## Coefficients:
##          ar1     ar2     ma1      ma2
##       0.3393  0.4727  0.1297  -0.3054
## s.e.  0.1328  0.1110  0.1311   0.0490
##
## sigma^2 estimated as 0.02232:  log likelihood=1563.74
## AIC=-3117.49   AICc=-3117.47   BIC=-3087.07

#after multiple iterations
(fitSARIMA <- arima(datanew, order = c(3,0,4), seasonal= list(order=c(1,0,0),
period=24)))

##
## Call:
```

```
## arima(x = datanew, order = c(3, 0, 4), seasonal = list(order = c(1, 0, 0),
period = 24))
##
## Coefficients:
##          ar1     ar2      ar3      ma1      ma2     ma3      ma4     sar1
##       0.9668  0.1415  -0.1458  -0.4648  -0.2436  0.0439  -0.0587  -0.5034
## s.e.  0.2476  0.3510   0.1570   0.2473   0.2416  0.0679   0.0255   0.0153
##       intercept
##          0.0088
## s.e.     0.0111
##
## sigma^2 estimated as 0.01671:  log likelihood = 2027.59,  aic = -4035.17
```

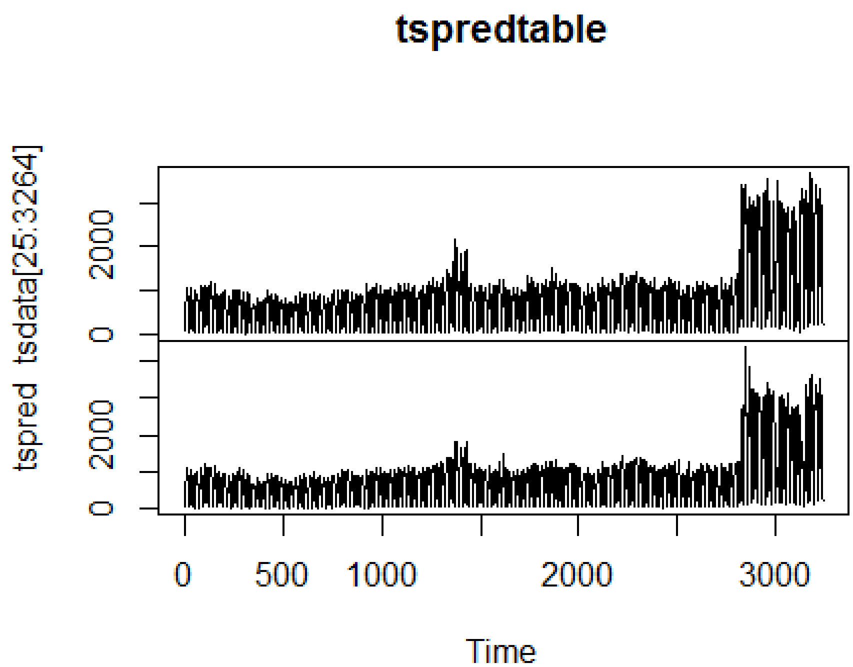
```
(fitSARIMA2 <- arima(datanew, order = c(5,0,2), seasonal= list(order=c(1,0,0)
, period=24)))
```

```
##
## Call:
## arima(x = datanew, order = c(5, 0, 2), seasonal = list(order = c(1, 0, 0),
period = 24))
##
## Coefficients:
##          ar1     ar2      ar3      ar4     ar5      ma1      ma2     sar1
##       1.0668  0.0448  -0.0914  -0.0879  0.0354  -0.5646  -0.1966  -0.5035
## s.e.  0.3628  0.4888   0.1116   0.0275  0.0264   0.3628   0.3085   0.0153
##       intercept
##          0.0084
## s.e.     0.0111
##
## sigma^2 estimated as 0.01671:  log likelihood = 2027.87,  aic = -4035.75
```

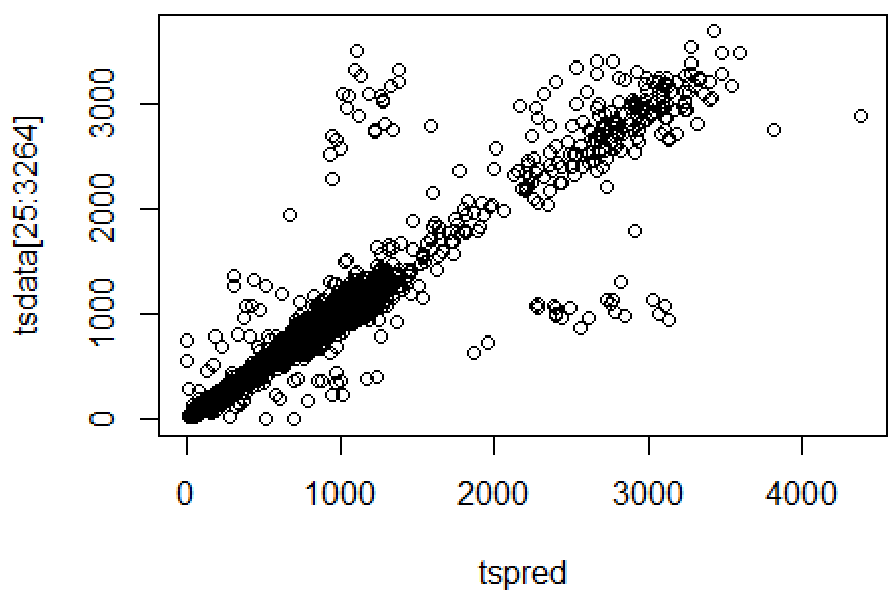
```
#residuals
tsdisplay(residuals(fitSARIMA2))
```



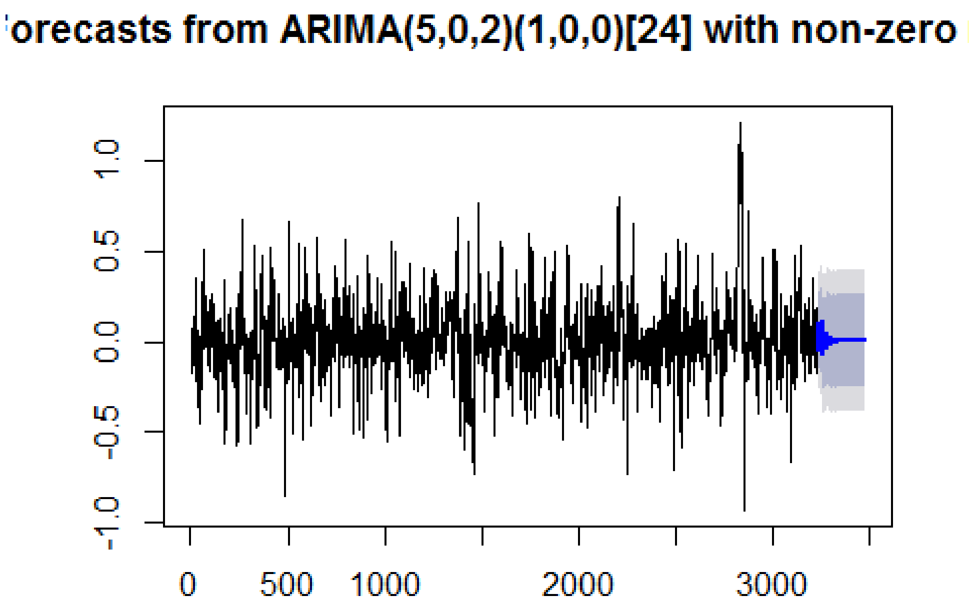

```
Box.test(residuals(fitSARIMA2), lag=24, fitdf=4, type="Ljung-Box")

##
##  Box-Ljung test
##
## data:  residuals(fitSARIMA2)
## X-squared = 131.77, df = 20, p-value < 0.00000000000000022

#Make predictions
tspred=tsdata[25:3264-24]*exp(fitted.values(fitSARIMA2))
tspredtable=cbind(tsdata[25:3264],tspred)
head(tspredtable)

##      tsdata[25:3264]    tspred
## [1,]              61  68.25842
## [2,]             136 117.59485
## [3,]             272 277.85381
## [4,]             385 402.39480
## [5,]             590 638.32365
## [6,]             869 845.81689

plot(tspredtable)
```




```
plot(tspred, tsdata[25:3264])
```




```
plot(forecast(fitSARIMA2,h=10*24))
```




```
forecastSARIMA <- forecast(fitSARIMA2, level=c(80,95), h=10*24)

##############################################################################
##########
#supervised learning parameters
#day of the year
data$yearday=format(data$newdate,"%j")
data$yearday<-as.integer(data$yearday)

#Day of the month
data$monthday =day(data$newdate)

#Day of the week
data$weekday=format(data$newdate,"%w")
data$weekday=as.integer(data$weekday)

#Ordinal date
data$datefrom1=difftime(data$newdate,"01-01-1900", units="days")
data$datefrom1=as.integer(data$datefrom1)

features=c("hr_of_day","yearday","monthday","weekday","vals")
data1=data[,features]
head(data1)

##   hr_of_day yearday monthday weekday vals
## 1         0     121        1       4   72
```

```
## 2          1     121        1       4  127
## 3          2     121        1       4  277
## 4          3     121        1       4  411
## 5          4     121        1       4  666
## 6          5     121        1       4  912

str(data1)

## 'data.frame':    3264 obs. of  5 variables:
##  $ hr_of_day: int  0 1 2 3 4 5 6 7 8 9 ...
##  $ yearday  : int  121 121 121 121 121 121 121 121 121 121 ...
##  $ monthday : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ weekday  : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ vals     : int  72 127 277 411 666 912 1164 1119 951 929 ...

summary(data1)

##    hr_of_day        yearday         monthday        weekday
##  Min.   : 0.00   Min.   :121.0   Min.   : 1.00   Min.   :0.000
##  1st Qu.: 5.75   1st Qu.:154.8   1st Qu.: 7.00   1st Qu.:1.000
##  Median :11.50   Median :188.5   Median :14.00   Median :3.000
##  Mean   :11.50   Mean   :188.5   Mean   :15.03   Mean   :3.044
##  3rd Qu.:17.25   3rd Qu.:222.2   3rd Qu.:23.00   3rd Qu.:5.000
##  Max.   :23.00   Max.   :256.0   Max.   :31.00   Max.   :6.000
##       vals
##  Min.   :   1.0
##  1st Qu.: 254.0
##  Median : 775.5
##  Mean   : 797.0
##  3rd Qu.:1027.0
##  Max.   :3695.0

#Train Test
set.seed(2345)
sub = sample(nrow(data1), floor(nrow(data1) * 0.7))
train = data1[sub,]
yTrain = train$vals
test = data1[-sub,]
yTest = test$vals
feature.formula = formula(vals~hr_of_day+yearday+monthday+weekday)

# Matrix
indexes <- sample(seq_len(nrow(train)), floor(nrow(train)*0.85))
datax <- sparse.model.matrix(feature.formula, data = train[indexes, ])
sparseMatrixColNamesTrain <- colnames(datax)
dtrain <- xgb.DMatrix(datax, label = train[indexes, 'vals'])
rm(datax)
dvalid <- xgb.DMatrix(sparse.model.matrix(feature.formula, data = train[-inde
xes, ]),
                      label = train[-indexes, 'vals'])
dtest <- sparse.model.matrix(feature.formula, data = test)
```

```r
watchlist <- list(valid = dvalid, train = dtrain)

# XGBOOST
params <- list(booster = "gbtree", objective = "reg:linear",
               max_depth = 8, eta = 0.02,
               colsample_bytree = 0.8, subsample = 0.9)
model <- xgb.train(params = params, data = dtrain,
                   nrounds = 500, early.stop.round = 50,
                     maximize = F,
                   watchlist = watchlist, print.every.n = 10)
```

```
## Warning in xgb.train(params = params, data = dtrain, nrounds = 500,
## early.stop.round = 50, : Only the first data set in watchlist is used for
## early stopping process.
```

```
## [0]  valid-rmse:1025.929077  train-rmse:1018.357239
## [10] valid-rmse:864.494263   train-rmse:855.674072
## [20] valid-rmse:726.161987   train-rmse:716.214050
## [30] valid-rmse:610.534851   train-rmse:599.262512
## [40] valid-rmse:523.667419   train-rmse:510.600861
## [50] valid-rmse:448.172302   train-rmse:434.052032
## [60] valid-rmse:389.560577   train-rmse:373.387756
## [70] valid-rmse:337.719147   train-rmse:319.381897
## [80] valid-rmse:297.137604   train-rmse:276.902100
## [90] valid-rmse:264.565826   train-rmse:242.656158
## [100]    valid-rmse:232.425339   train-rmse:209.231369
## [110]    valid-rmse:211.596466   train-rmse:185.574417
## [120]    valid-rmse:195.251816   train-rmse:167.791168
## [130]    valid-rmse:178.350235   train-rmse:148.796616
## [140]    valid-rmse:165.145325   train-rmse:134.262711
## [150]    valid-rmse:156.686813   train-rmse:123.865700
## [160]    valid-rmse:149.091492   train-rmse:114.267960
## [170]    valid-rmse:143.323898   train-rmse:107.297165
## [180]    valid-rmse:138.162888   train-rmse:100.619820
## [190]    valid-rmse:134.102432   train-rmse:95.109680
## [200]    valid-rmse:130.269592   train-rmse:89.679359
## [210]    valid-rmse:127.279480   train-rmse:85.567764
## [220]    valid-rmse:125.126366   train-rmse:82.102058
## [230]    valid-rmse:122.580467   train-rmse:78.089035
## [240]    valid-rmse:120.760513   train-rmse:74.511421
## [250]    valid-rmse:119.855919   train-rmse:72.133446
## [260]    valid-rmse:118.609772   train-rmse:69.583153
## [270]    valid-rmse:117.392372   train-rmse:67.191322
## [280]    valid-rmse:116.502174   train-rmse:64.866669
## [290]    valid-rmse:115.863640   train-rmse:62.712147
## [300]    valid-rmse:114.882523   train-rmse:60.757000
## [310]    valid-rmse:114.232582   train-rmse:58.991615
## [320]    valid-rmse:113.470139   train-rmse:57.156296
## [330]    valid-rmse:112.742691   train-rmse:55.813206
```

```
## [340]    valid-rmse:112.041779   train-rmse:54.217514
## [350]    valid-rmse:111.672371   train-rmse:52.389591
## [360]    valid-rmse:111.336700   train-rmse:51.234535
## [370]    valid-rmse:111.258522   train-rmse:49.819126
## [380]    valid-rmse:111.076233   train-rmse:48.701645
## [390]    valid-rmse:111.024078   train-rmse:47.338081
## [400]    valid-rmse:110.709038   train-rmse:46.774048
## [410]    valid-rmse:110.039749   train-rmse:45.462280
## [420]    valid-rmse:109.760010   train-rmse:44.154945
## [430]    valid-rmse:109.786224   train-rmse:43.163120
## [440]    valid-rmse:109.549042   train-rmse:42.153442
## [450]    valid-rmse:109.352615   train-rmse:41.135254
## [460]    valid-rmse:109.062637   train-rmse:40.078926
## [470]    valid-rmse:108.953300   train-rmse:39.122906
## [480]    valid-rmse:108.687958   train-rmse:38.369881
## [490]    valid-rmse:108.401329   train-rmse:37.459492
```

```r
pred <- predict(model, dtest)

plot(pred,yTest)
```