SARIMA modeling

Nitin

Wednesday, May 06, 2015

SARIMA modeling

This is an analysis of time series analysis of a time series dataset provided for a job interview. The dataset includes a time series variable named simply val, and is indexed for 139 days by hour of the day.

Before moving on let us first load the libraries.

```
require(forecast)
```

```
## Loading required package: forecast
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Loading required package: timeDate
## This is forecast 5.9
```

```
require(digest)
```

```
## Loading required package: digest
```

```
require(foreign)
```

```
## Loading required package: foreign
```

```
require(TSA)
```

```
## Loading required package: TSA
## Loading required package: leaps
## Loading required package: locfit
## locfit 1.5-9.1    2013-03-22
## Loading required package: mgcv
## Loading required package: nlme
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:forecast':
##
##     getResponse
##
```

```
## This is mgcv 1.8-6. For overview type 'help("mgcv-package")'.
## Loading required package: tseries
##
## Attaching package: 'TSA'
##
## The following object is masked from 'package:forecast':
##
##     fitted.Arima
##
## The following objects are masked from 'package:timeDate':
##
##     kurtosis, skewness
##
## The following objects are masked from 'package:stats':
##
##     acf, arima
##
## The following object is masked from 'package:utils':
##
##     tar
```

Let us now read the data from the `tsv` or the *tab separated value* file. To read such a file we will use the read.table command providing the separater as "". We also mention that the header is `TRUE` as the file containes the header.

```
dat=read.table("data_scientist_assignment.tsv",header=TRUE,sep="\t")
```

After loading the data we will explore it a bit by looking at the head, tail, and the summary of the data.

```
head(dat)
```

```
##         date hr_of_day vals
## 1 2014-05-01         0   72
## 2 2014-05-01         1  127
## 3 2014-05-01         2  277
## 4 2014-05-01         3  411
## 5 2014-05-01         4  666
## 6 2014-05-01         5  912
```

```
tail(dat)
```

```
##            date hr_of_day vals
## 3259 2014-09-13        18 2088
## 3260 2014-09-13        19 1585
## 3261 2014-09-13        20  930
## 3262 2014-09-13        21  443
## 3263 2014-09-13        22  327
## 3264 2014-09-13        23  200
```

```
summary(dat)
```

```
##          date        hr_of_day          vals
##  2014-05-01:  24   Min.   : 0.00   Min.   :   1.0
##  2014-05-02:  24   1st Qu.: 5.75   1st Qu.: 254.0
##  2014-05-03:  24   Median :11.50   Median : 775.5
##  2014-05-04:  24   Mean   :11.50   Mean   : 797.0
##  2014-05-05:  24   3rd Qu.:17.25   3rd Qu.:1027.0
##  2014-05-06:  24   Max.   :23.00   Max.   :3695.0
##  (Other)   :3120
```

```
str(dat)
```

```
## 'data.frame':    3264 obs. of  3 variables:
##  $ date     : Factor w/ 136 levels "2014-05-01","2014-05-02",..: 1 1 1 1 1 1 1 1 1 ...
##  $ hr_of_day: int  0 1 2 3 4 5 6 7 8 9 ...
##  $ vals     : int  72 127 277 411 666 912 1164 1119 951 929 ...
```

Next, we convert the date in the `Date` class available in R.

```
dat[,"date"]=as.Date(dat[,"date"],origin="2014-05-01")
head(dat)
```

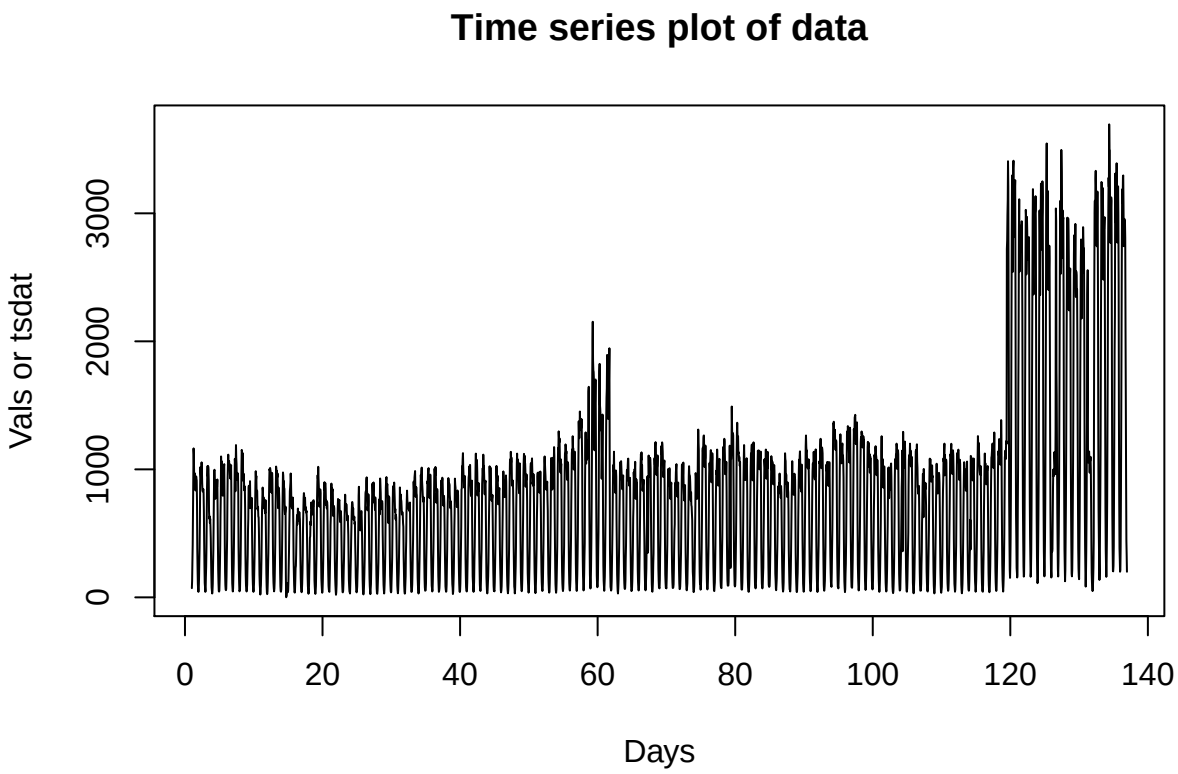
```
##         date hr_of_day vals
## 1 2014-05-01         0   72
## 2 2014-05-01         1  127
## 3 2014-05-01         2  277
## 4 2014-05-01         3  411
## 5 2014-05-01         4  666
## 6 2014-05-01         5  912
```

Clearly, we have a time series at our hand and with a frequency 24. So, we create a new variable `tsdat` to store this time series in standard time series format.

```
tsdat=ts(dat$vals,frequency=24)
```

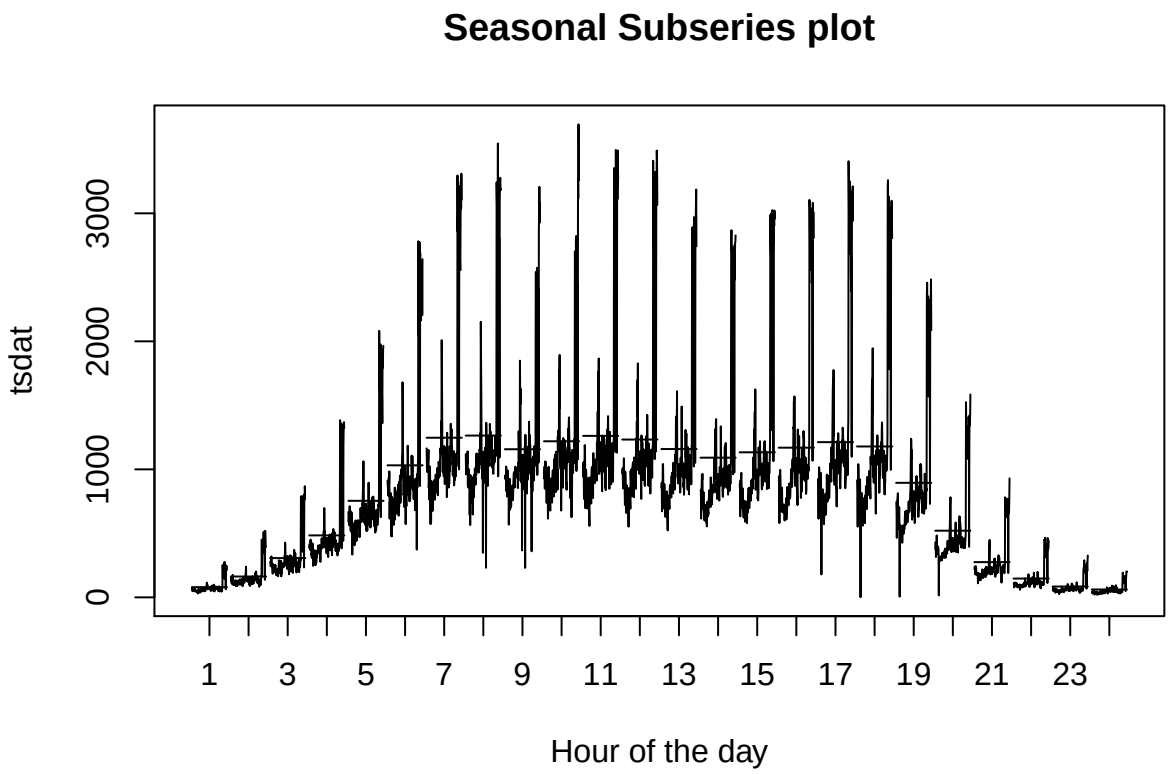
Let us now look at the plot of the time series.

```
plot(tsdat, type="l", xlab="Days", ylab="Vals or tsdat", main="Time series plot of data")
```



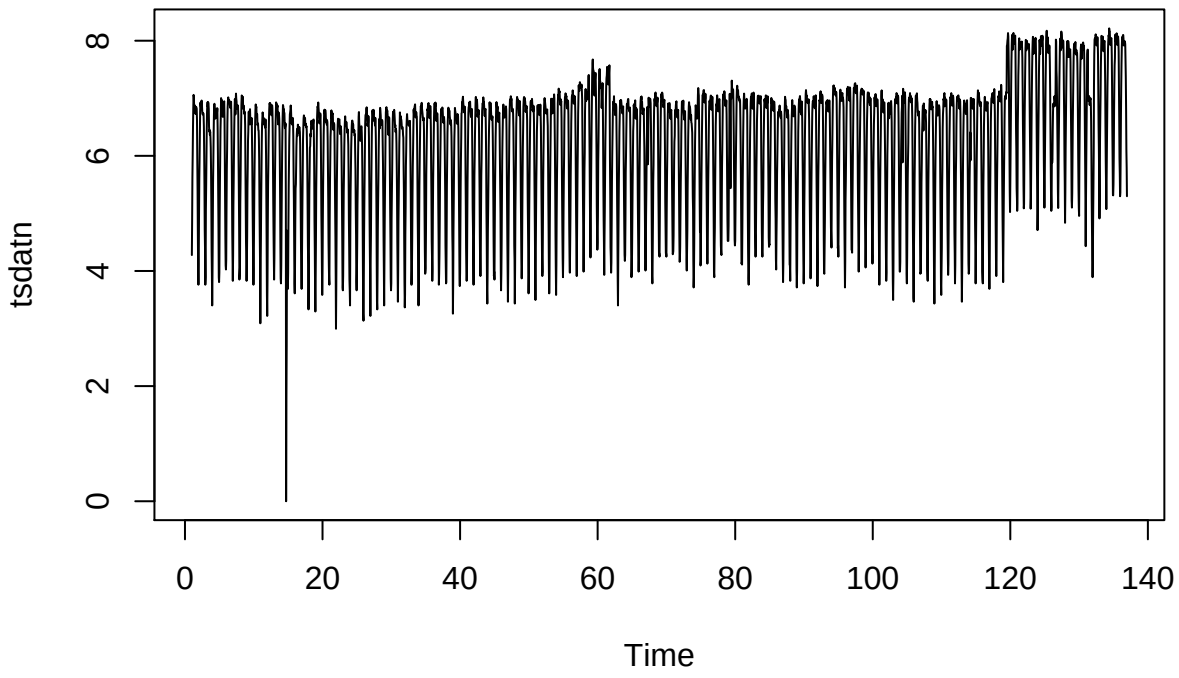
The plot shows a good amount of seasonality in the data as well as a need for transformation. Let us look at the seasonality first by plotting the seasonal subseries by the `monthplot` function.

```
monthplot(tsdat, xlab="Hour of the day", main="Seasonal Subseries plot")
```



We can not expect a better indication of seasonality than this plot. So, we need to remove the seasonality from the time series, but before that we would like to naturalize the data by the **Box Cox Transformation** with $\lambda = 0$, i.e by taking log of the entire time series. We store the new time series in the variable `tsdatn`.

```
tsdatn=log(tsdat)
plot(tsdatn)
```



Let us have a look at the outliers.

```
tsdatnout=tsoutliers(tsdatn,iterate=24)
tsdatnout
```

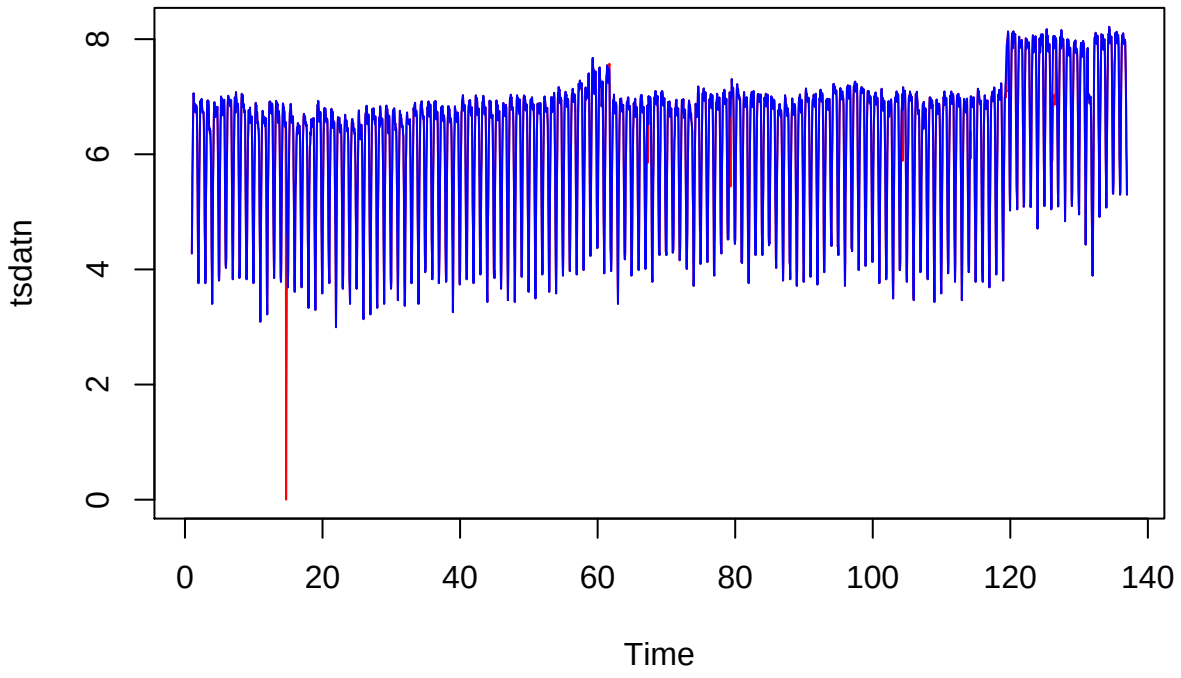
```
## $index
##  [1]  329  330  331  332 1458 1592 1593 1880 1881 2481 2517 2519 2718 2840
## [15] 2841 2842 2843 2844 2882 3003 3004 3005 3006 3007 3008 3009 3010 3011
## [29] 3012 3013 3014 3015 3146 3147 3148
##
## $replacements
##  [1] 6.533037 6.409921 6.047968 5.427865 7.433260 6.709223 6.551854
##  [8] 6.795086 6.869956 6.861109 5.135808 3.611321 6.778259 7.197064
## [15] 7.260165 7.448555 7.642722 7.786124 5.887753 6.592619 7.065635
## [22] 7.499296 7.827612 8.023199 8.055988 7.834740 7.884291 8.055621
## [29] 8.055617 7.966985 7.892695 7.991032 4.997855 5.852595 6.595542
```

Notice how we have the index for the replacements required to remove the outliers. Let us do the replacements.

```
tsdatr=tsdatn
tsdatr[tsdatnout$index]=tsdatnout$replacements
```

Let us have a look at how we removed some of the outliers from the timeseries:

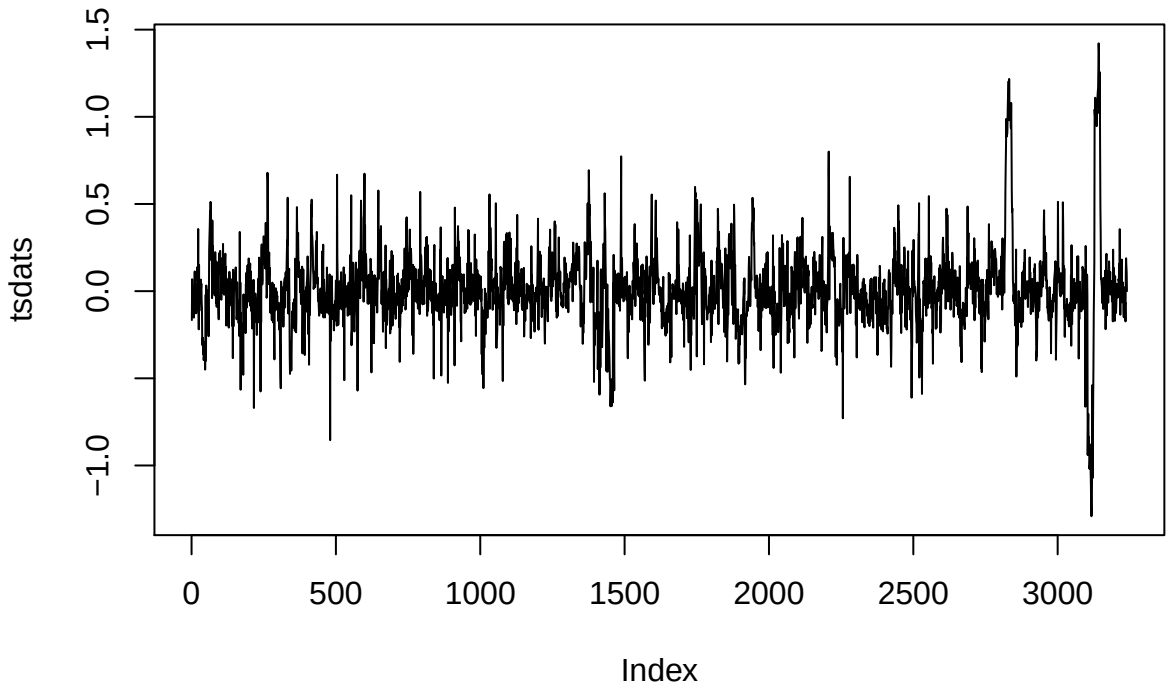
```
plot(tsdatn,col="red")
lines(tsdatr,col="blue")
```



Notice that the outliers are not totally removed. But we are at a much better stage than before and thats why we will just leave it as it is.

Let us now remove the seasonality by doing a seasonal difference.

```
tsdats=tsdatr[25:3264]-tsdatr[25:3264-24]
plot(tsdats, type="l")
```

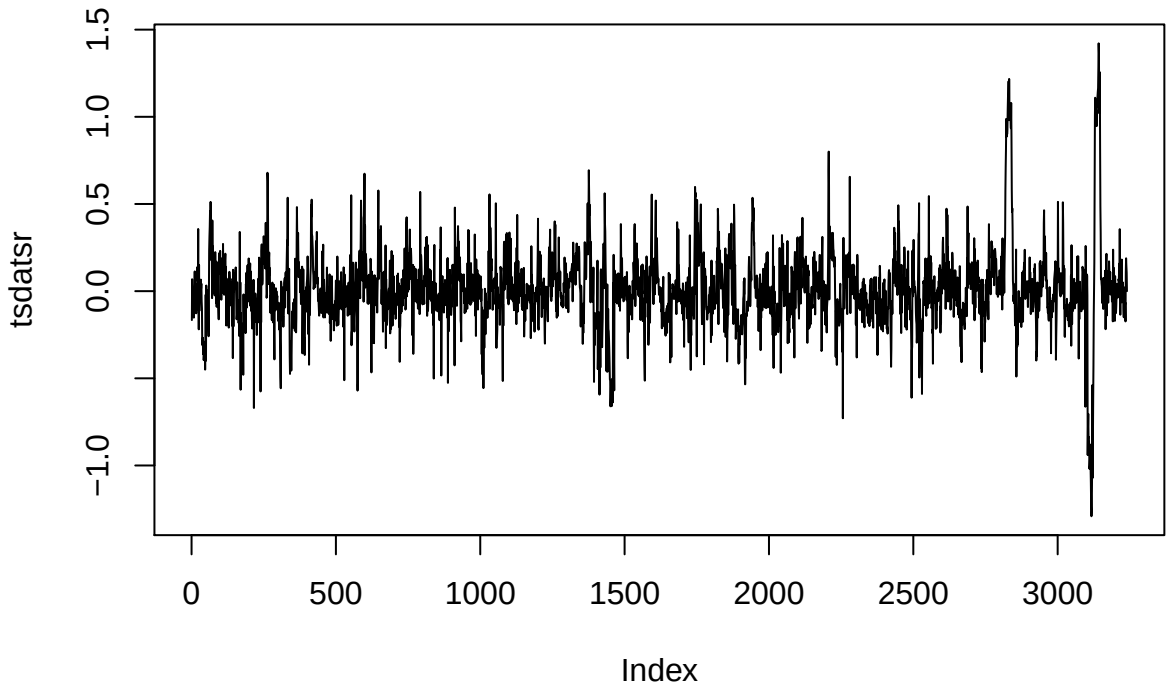


This appears a resonably stationary data. We do another sweep for the outliers. And do the suggested replacements.

```
tsdatnout=tsoutliers(tsdats,iterate=24)
tsdatnout
```

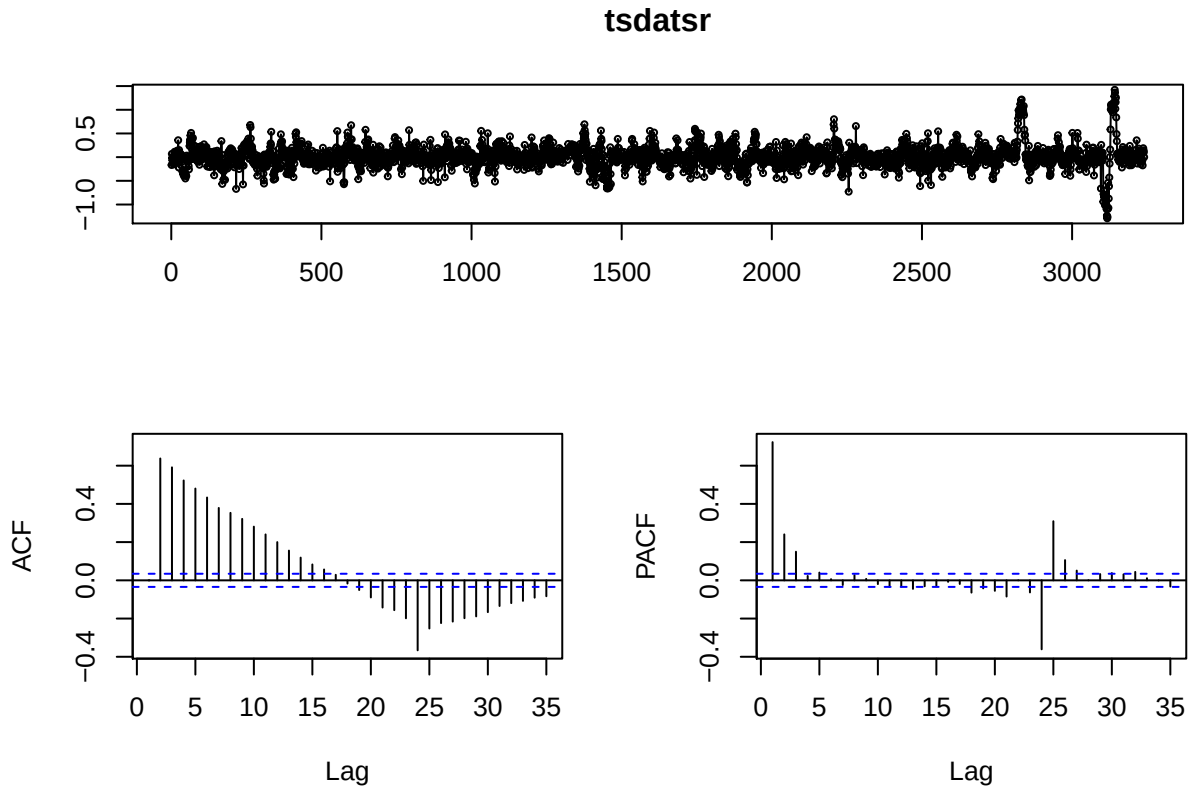
```
## $index
## [1]  480  504 1488 3128
##
## $replacements
## [1] -0.1149748  0.0000000  0.1994160  0.5849162
```

```
tsdatsr=tsdats
tsdatsr[tsdatnout$index]=tsdatnout$replacements
plot(tsdatsr,type="l")
```



Let us now check the `ACF` and `PCF` curves to get an idea of the given time series. We are using the `tsdisplay` function to get a good summary here.

```
tsdisplay(tsdatsr)
```



The `Auto-Correlation Function` has a shape that is alternating positive and negative and it appears to be decreasing to zero. That suggests an autoregressive model. Looking at the `Partial Auto-Correlation Function` we see that there are spikes at 24. 25, 26, 28, 29, 30, 31 and 32. Also, we notice that 24th spike is negative and the spikes after this one looks very similar to the initial spikes. This indicates that a seasonal autoregressive factor should be included along with autoregressive factors.

Let us explain a part of these spikes by a seasonal autoregression factor and the rest by ARIMA(5,0,2) which is giving coefficients corresponding spikes.

So, we are trying to fit a seasonal ARIMA model SARIMA

$$(5,0,2)(1,0,0)_24$$

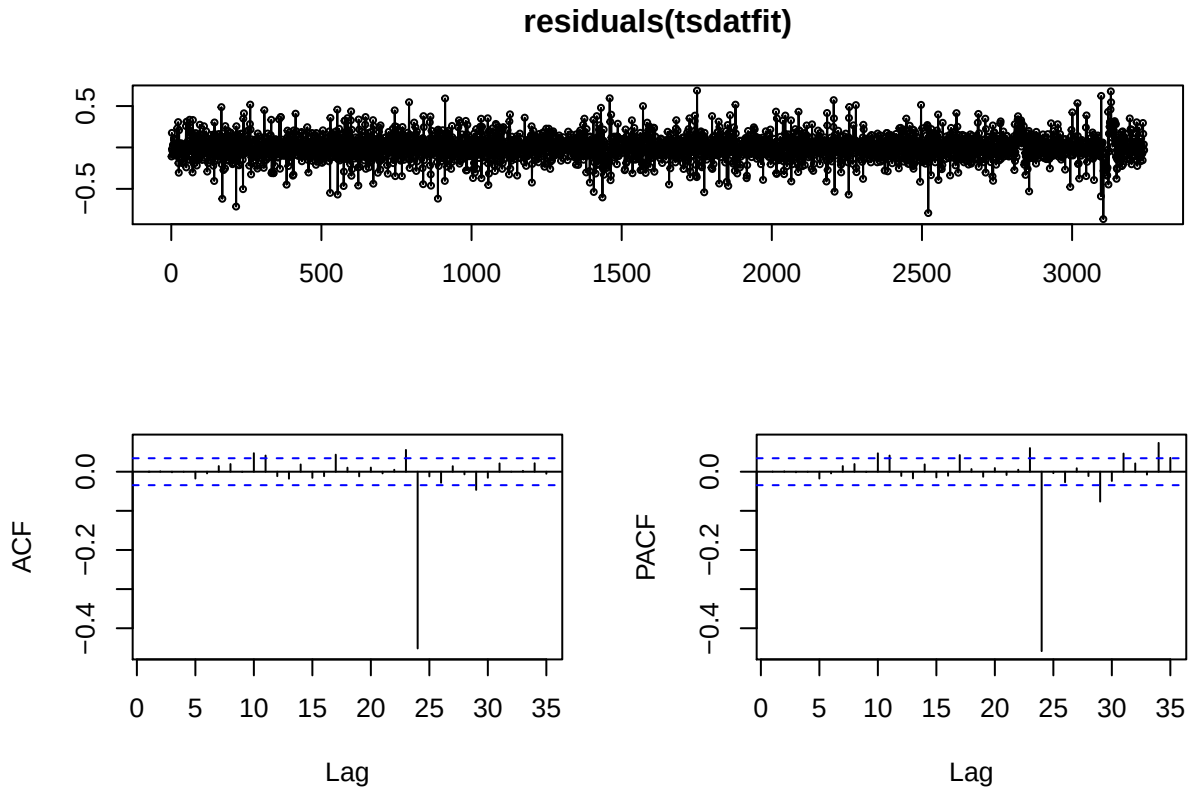
```
tsdatfit=arima(tsdatsr,order=c(5,0,2),seasonal=c(1,0,0))
tsdatfit
```

```
##
## Call:
## arima(x = tsdatsr, order = c(5, 0, 2), seasonal = c(1, 0, 0))
##
## Coefficients:
##          ar1      ar2     ar3     ar4     ar5     ma1     ma2     sar1
##       0.2245  -0.5908  0.8236  0.0252  0.1791  0.5228  0.9945  -0.2263
## s.e.  0.0803   0.0404  0.0455  0.0696  0.0197  0.0030  0.0020   0.0819
##       intercept
##          0.0033
## s.e.     0.0156
```

```
##
## sigma^2 estimated as 0.02136:  log likelihood = 1631.1,  aic = -3244.2
```

As an indicator for the fit let us look at the residuals of this model, by plotting the `tsdisplay` of the residuals and performing the `Ljung-Box` test which *is a type of statistical test of whether any of a group of autocorrelations of a time series are different from zero.*

```
tsdisplay(residuals(tsdatfit))
```




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

```
##
##  Box-Ljung test
##
## data:  residuals(tsdatfit)
## X-squared = 703.4065, df = 20, p-value < 2.2e-16
```

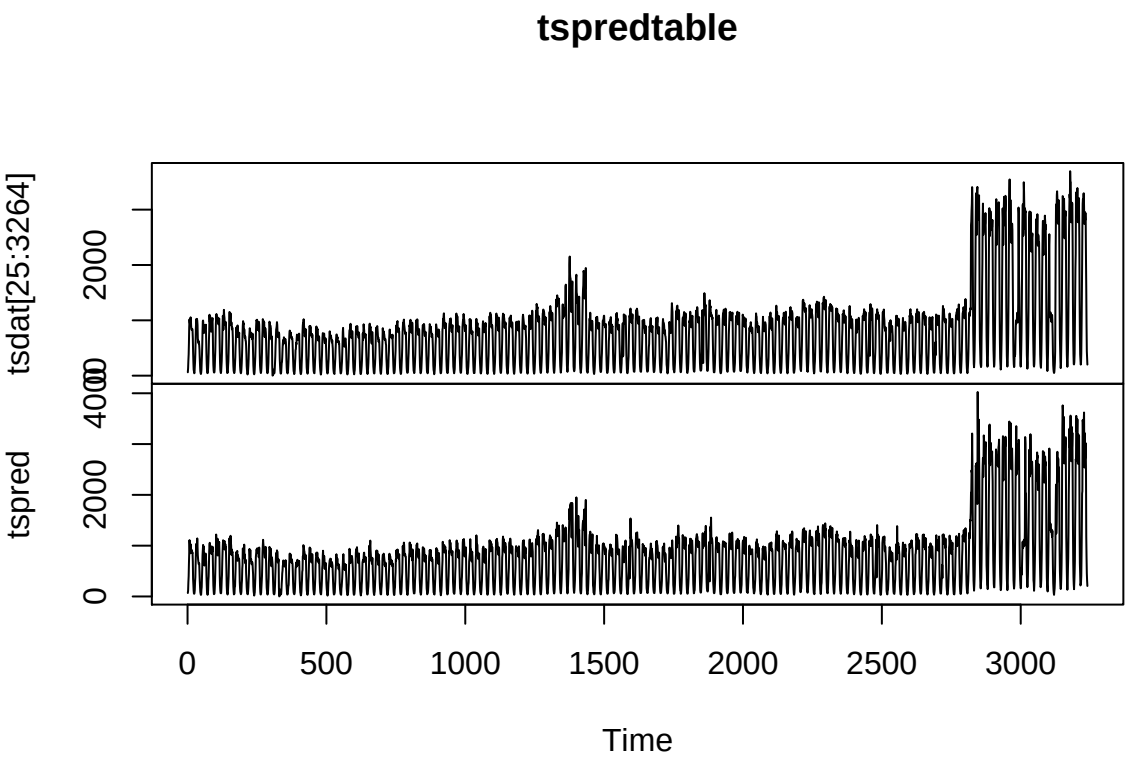
The `ACF` and `PACF` plots are showing possiblity for an extra seasonal autoregresser although the values are still within margin or erro. The small p-value returned by `Ljung-Box` give a firm answer that we do not need to change the model further.

Let us define a function to get the predicted values from this model for the original data and plot the predicted value against the given value.

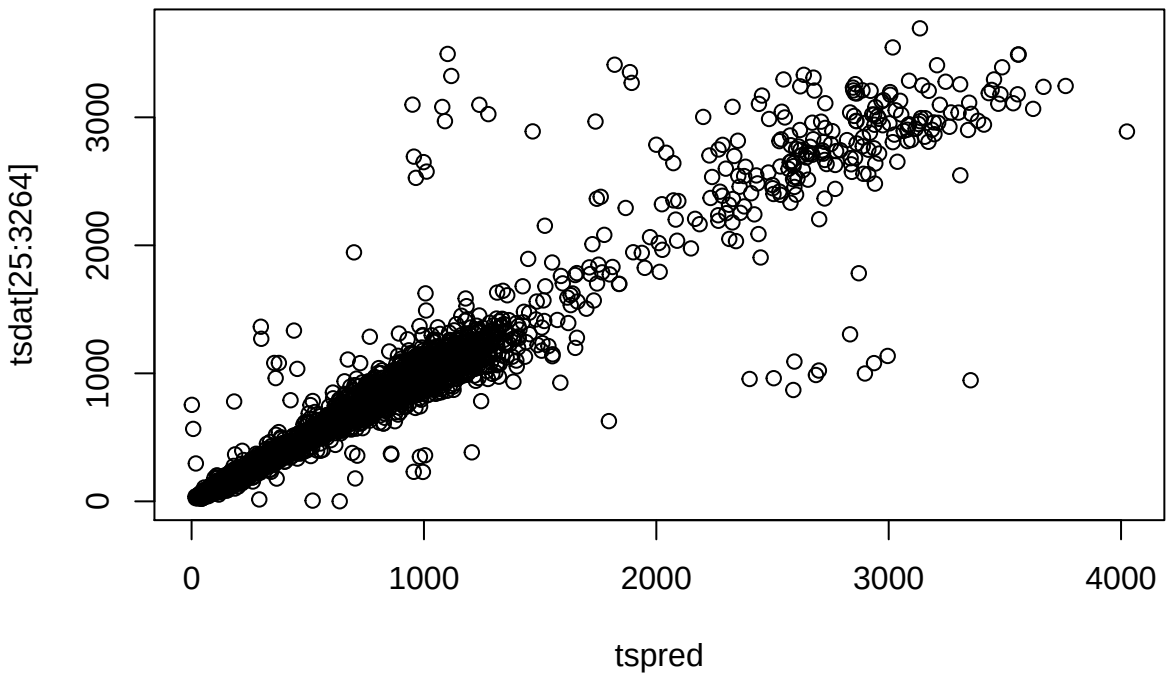
```
tspred=tsdat[25:3264-24]*exp(fitted.values(tsdatfit))
tspredtable=cbind(tsdat[25:3264],tspred)
head(tspredtable)
```

```
##      tsdat[25:3264]    tspred
## [1,]             61  68.03837
## [2,]            136 114.08181
## [3,]            272 276.39221
## [4,]            385 401.34895
## [5,]            590 644.88946
## [6,]            869 840.93076
```

```
plot(tspredtable)
```




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


tsdat[25:3264]
0
1000
2000
3000
0
1000
2000
3000
4000
tspred