INTRODUCTION TO TIME SERIES - MATH5540

A MODEL COMPARISON BETWEEN HOLT WINTERS AND STL METHODS

May 4, 2018

Koushik Ganesan
University of Colorado Boulder
Department of Physics

Contents

Introduction

In this essay we aim to perform a full analysis on a real time dataset and compare the performance of Holt Winters method to that of Seasonal and Trend decomposition using Loess (STL) methods.

The first part of this essay provides a brief introduction to Holt Winters and STL methods. We then compare their performance over dataset of 5-minute average observed wind speeds at Tillamook, WA during January 2-8, 2013. In the final part of the essay we perform a simple forecasting for January 9 2013, using these methods.

Holt Winters method

A Simple Exponential Smoothing (SES) gives past events exponentially decaying weights in a window, unlike a simple moving average filter with equal weights. Given a raw dataset $\{x_t\}$, the SES gives a smoothened dataset $\{s_t\}$ given by :

$$s_t = \alpha x_t + (1-\alpha)s_{t-1}$$

where $0 < \alpha < 1$ is the smoothing parameter.
Holt's method was to improve on SES to allow forecasting data with a trend using the set of recursive equations:

$$\hat{y}_{t+h|t} = l_t + b_t h$$

$$l_t = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1})$$

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}$$

where l_t is the level estimate of the series at t, b_t is the estimate of the slope at time t, and α is the smoothing parameter for the level and β is that of the trend.
Holt and Winters further extended Holt's linear trend model to include for seasonality. This model comprises of three smoothing equations for trend, level and seasonality.

$$\hat{y}_{t+h|t} = l_t + b_t h + s_{t-m+h*}$$

$$l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1})$$

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}$$

$$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}$$

where $h* = [(h-1) mod m] + 1$ and m denotes the seasonality in the data. It also allows for a multiplicative seasonal component but for our analysis we restrict ourselves to an additive method.

STL decomposition

STL stands for "Seasonal and Trend decomposition using Loess". It is simple and a robust method which allows the seasonal and smoothness of the trend component to vary over time as controlled by the input parameters. It is an additive decomposition as follows :

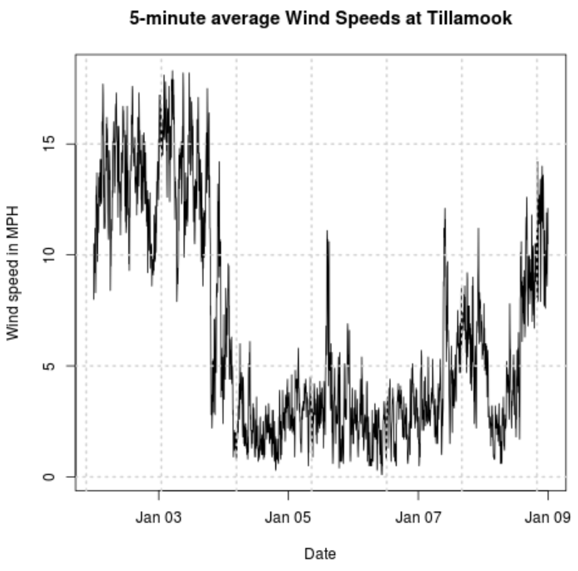
$$X_t = S_t + T_t + E_t$$

where S_t, T_t and E_t are the seasonal, trend and error components.

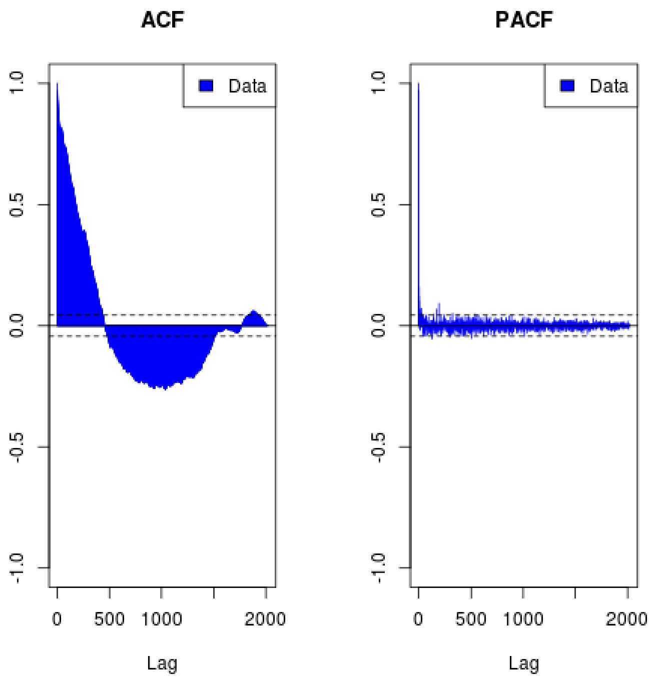
It calculates seasonally adjusted data by subtracting the seasonal component and uses Loess ("locally-weighted scatterplot smoothing") to smoothen the data, which chooses a small window over the data and performs linear regression over it by weighing the points closest to the center of the window more. The wider the window the more smoother the fitted curve. We then use this smooth seasonally adjusted data to fit an appropriate ARIMA model. Forecasting is done using this best fit ARIMA model, then re-seasonalizing the data by adding back the seasonal component.

Data Analysis

We are in a position to use the above mentioned models and test their performance over the given data by splitting it into training and testing sets respectively. We let them train on the training set and compare their performance against the test set. The given data has been plotted below. The first step is to clean the data using the function $tsclean()$ which is a convenient method for outliers removal and inputing missing values.



We immediately observe that the given data is heteroscedastic and there seems to be a number of irregularities. Just by glancing its obviously non-stationary. The ACF, PACF plots combined with the $adf.test()$ of the data, which does not reject the null hypothesis of non-stationarity, and confirms this visual inspection.



```
        Augmented Dickey-Fuller Test

data:  dat
Dickey-Fuller = -0.64379, Lag order = 288, p-value = 0.9753
alternative hypothesis: stationary
```

As this data indicates atmospheric conditions over a period of days, its fitting to assume that they are periodic with period of 288 given that we have 288 points for each day in 5-minute intervals, which is why we have used the lag order in the $adf.test()$ to be 288.

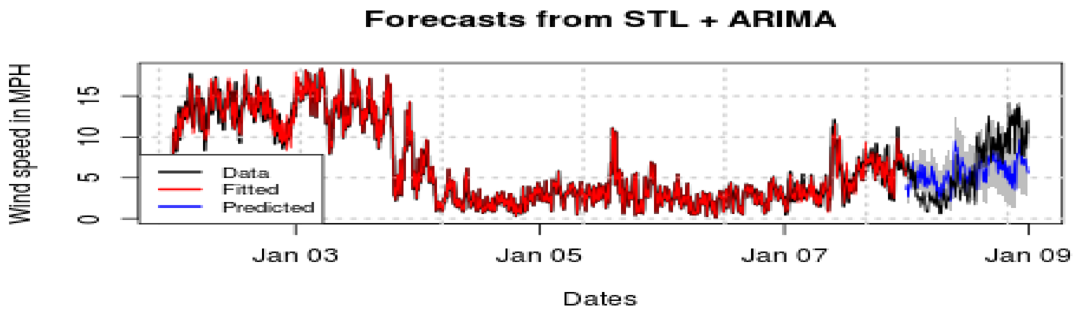
We now test the performance of the models. Our training set contains data from Jan 2-7, 2013 and the test set contains the data for Jan 8, 2013.

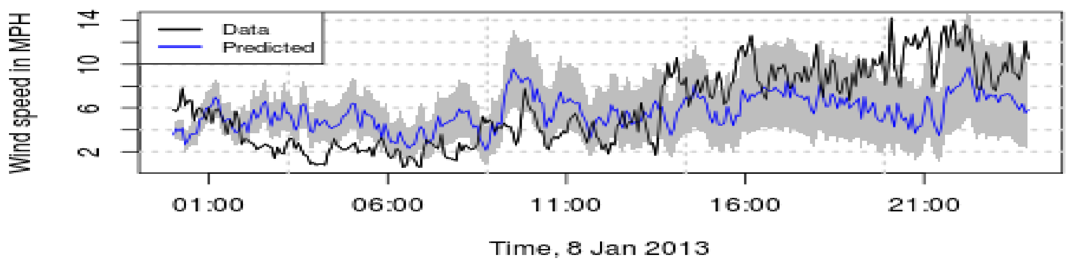
Fitting STL model

The function in *R* that fits this method on the time series data is given by *stlm*() which takes in a time series data, applies STL decomposition and then models the seasonal adjusted data with an ARIMA model. We use this function to fit the training dataset.

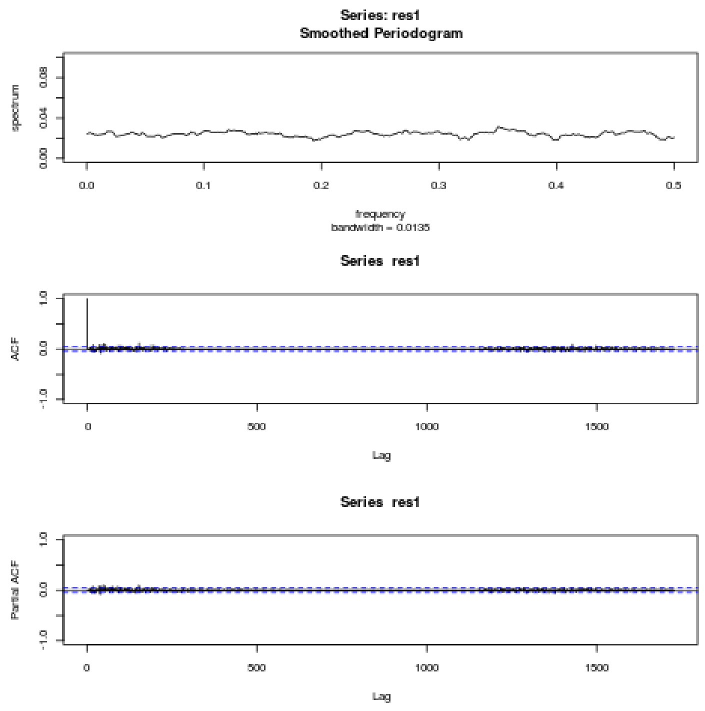
```
Series: x
ARIMA3,1,1

Coefficients:
         ar1      ar2     ar3      ma1
      0.7444  -0.0046  0.0825  -0.9456
s.e.  0.0269   0.0303  0.0249   0.0117

sigma^2 estimated as 0.02415:  log likelihood=766.53
AIC=-1523.05   AICc=-1523.02   BIC=-1495.78
```

We observe that the smoothed seasonally adjusted data has been fitted with an ARIMA(3,1,1). We can use the *forecast*() function to now forecast the model and compare it against the test data.





We can see that the we have a good fit for the training dataset. The blue line is the forecast for Jan 8, 2013. The black line indicates the observations/original data. The gray shaded region is the 95% confidence interval of the prediction. To see how well the model fits we inspect the residuals.



```
        Augmented Dickey-Fuller Test

data:  res1
Dickey-Fuller = -3.6304, Lag order = 288, p-value = 0.02975
alternative hypothesis: stationary
```

From these plots of smoothened Periodogram (parameter $m \approx \sqrt{n}$, where $n = 1728$, is the number of data points), ACF, PACF (starts at lag = 1) and the $adf.test()$ ($p < 0.05$) it can be concluded that the residuals from this fit resemble white noise.

To measure how well the prediction for Jan 8, 2013 matches with our testing set, we use statistical measures such as mean, variance and the correlation.

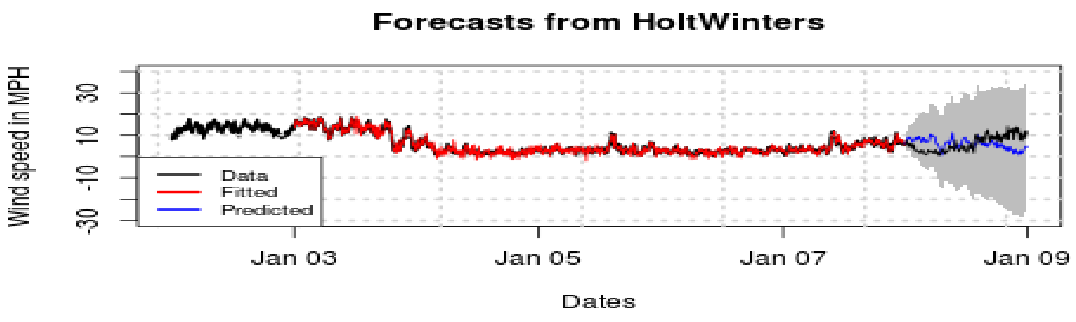
```
Mean of the difference between Prediciton and Test set:  0.4317032
Variance of the difference between Prediciton and Test set:  10.22615
Correlation between Prediciton and Test set:  0.4390643
```

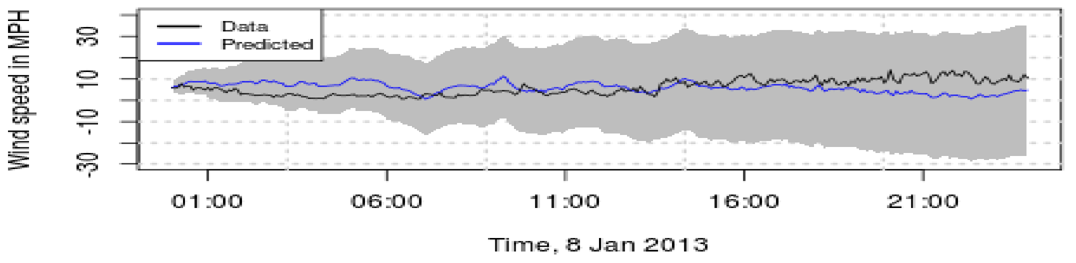
Fitting Holt Winters model

Similarly function in R that performs Holt Winters is given by $HoltWinters()$ which best fits by optimizing the three parameters. We similarly use this function on the training set and then use the $forecast()$ function to make predictions.

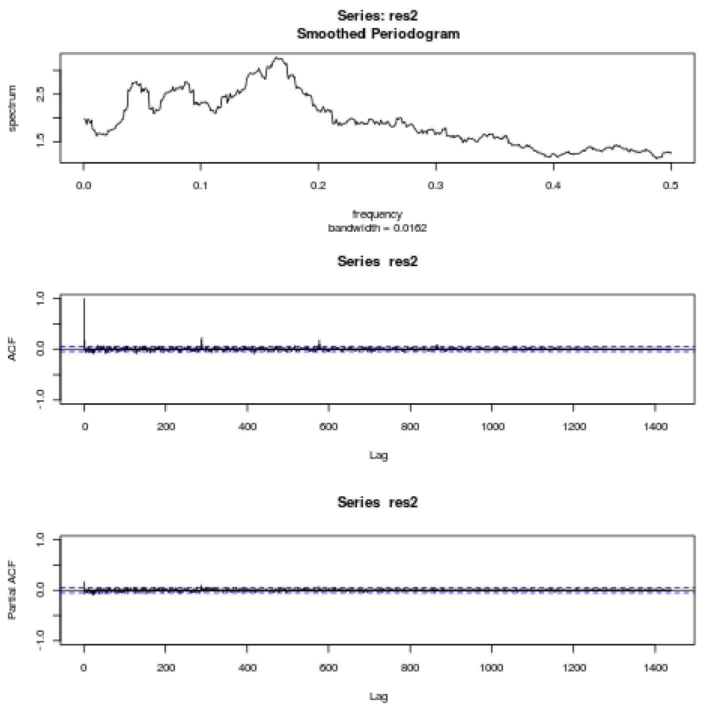
```
Holt-Winters exponential smoothing with trend and additive seasonal component.

Call:
HoltWintersx = train_dat, start.periods = 2

Smoothing parameters:
 alpha: 0.6605192
 beta : 0
 gamma: 1
```

I have used an additive seasonal component and the seasonality used is 288. We now forecast this fit for Jan 8, 2013.





As we have used the frequency to be 288, it has the fit starting from 289. Here again the blue line shows the prediction using this model for Jan 8, 2013, the red line indicates the fit, black indicates the observations and the grey shaded region is the 95% confidence interval. To check how well it fits, we again analyze the residuals.



```
        Augmented Dickey-Fuller Test

data:  res2
Dickey-Fuller = -3.4405, Lag order = 288, p-value = 0.04802
alternative hypothesis: stationary
```

We observe that even though $adf.test()$ produces a $p < 0.05$ it still contains some peaks in the ACF and also the Periodogram does not resemble that of white noise. This is a consequence of the fact that the best fit uses $\beta = 0$ and the seasonality being a fixed frequency.

Its performance against our testing set can be measured similarly.

```
Mean of the difference between Prediciton and Test set:  0.1085007
Variance of the difference between Prediciton and Test set:  25.65944
Correlation between Prediciton and Test set:  -0.4810159
```

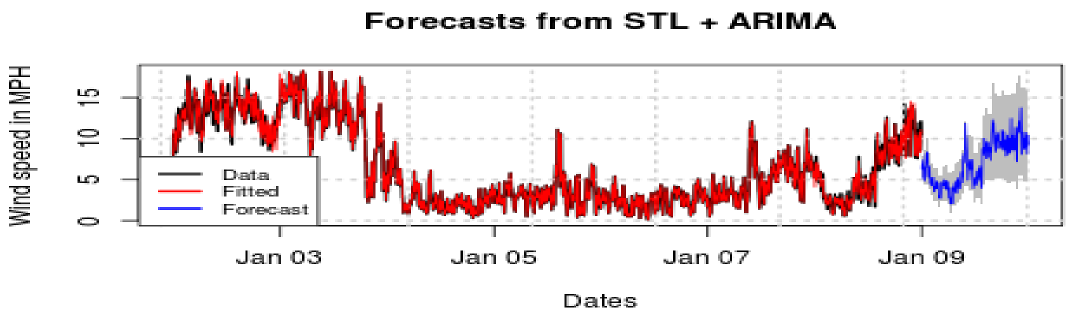
Performance analysis

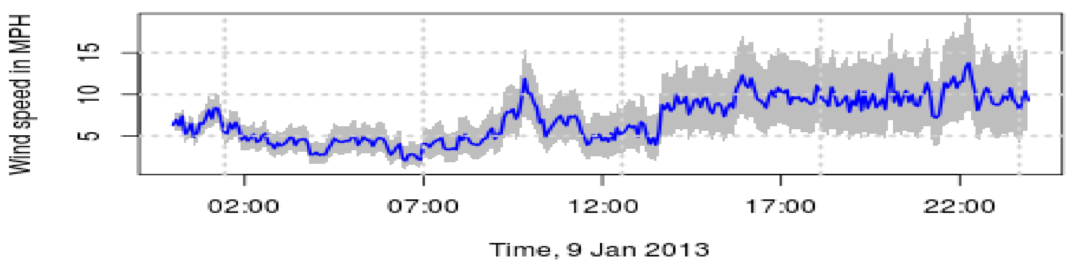
From the predictions and the test set statistics it can be seen that the STL + ARIMA model performed better than the Holt Winters on this data set. The primary reason being the adaptability of the STL method and its robustness. It allows for seasonality that changes over time, rate of change is decided by the user ($s.window = 2$ in our case), while the Holt Winters method works with the given seasonality set by the user and uses 288 seasonal indices to form the seasonal pattern. It assumes the seasonality does not change over time.

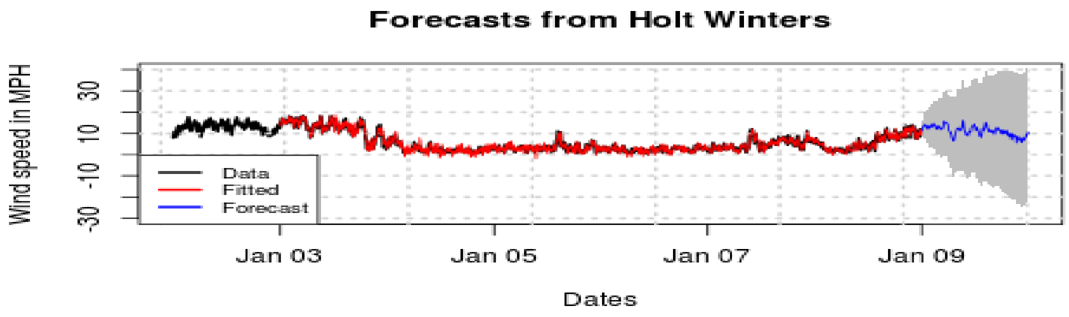
Both these methods are affected by a lag due to the smoothing process and can be seen from the plots and show deviations as time progresses but the uncertainty is larger for Holt Winters. Holt Winters best works for models where we have a dominant frequency that does not change over time. STL methods are best for data with high seasonal period such as this. Both these models are very simple to implement in *R*.

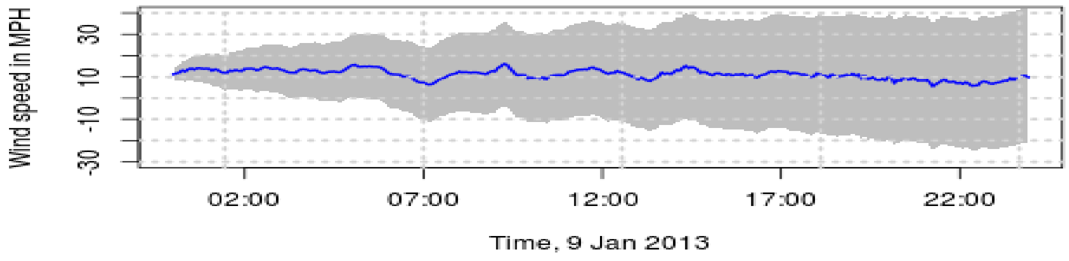
Forecasts for Jan 9, 2013

Now that we have tested the workings of both the models we move to plotting the forecasts for 5-minute average wind speed at Tillamook, WA on Jan 9, 2013.









We have used the same $forecast()$ function, except this time the given data in entirety is used to train our methods.

Conclusions and Acknowledgement

Long term forecasting using trend analysis is always prone of uncertainties especially with a volatile dataset such as weather data because historic data may not give the true underlying picture. Weather patterns are affected by a number of factors such as pressure gradient, air temperature, humidity, pollutants introduced into the atmosphere by human activities, which are difficult to model.

We have shown a naive implementation of Holt Winters and STL decomposition methods in this essay. We saw why STL+ARIMA performs better over Holt Winters on the given data set and seen some drawbacks of these methods. It is only natural to wonder how to improve on it. One possibility is the Double-seasonal Holt-Winters which allows for two levels of seasonality. Kalman filters are another class of versatile filters that adapt to the data by optimizing the Kalman gain which is the ratio of the process and measurement variance. It has been known to perform better than exponential smoothening filters and have been extensively used in stock market predictions.

I am grateful to Prof. Will Kleiber and the course MATH5540 for equipping me with the necessary knowledge regarding analysis of time series data and revealing how potent it can be with wide ranging applications, as we now live in the world of data.

References

Rob J. Hyndman, Yeasmin Khandakar, 2007. *Automatic Time Series Forecasting: The forecast Package for R.* URL: <https://www.jstatsoft.org/v027/i03>

Holt-Winters seasonal method. URL: <https://www.otexts.org/fpp/7/5>

Ruslana Dalinina, 2017. ,*Introduction to Time Series Forecasting.* URL: <https://www.datascience.com/blog/introduction-to-forecasting-with-arima-in-r-learn-data-science-tutorials>

STL decomposition. URL: <https://www.otexts.org/fpp/6/5>

Peter J. Brockwell, Richard A. Davis, *Introduction to Time Series and Forecasting.* Springer-Verlag New York, Inc., ISBN 0-387-95351-5

APPENDIX

I have attached the code used to perform the analysis here.

```
library(forecast)
library(tseries)
library(itsmr)
freq = 288

#---------READING THE DATA INTO DATAFRAME--------------------------------------
mydata = read.csv("data.csv", header = TRUE)
mydata$X <- NULL
dates <- as.POSIXct(mydata[,1],format="%m/%d/%Y␣%H:%M:%S",tz=Sys.timezone())
dat <- ts(tsclean(mydata[,2]))
png('dat.png')
plot(dates,dat, type = 'l', xlab = "Date", ylab = "Wind␣speed␣in␣MPH", main = "5-minute␣average␣Wind␣Speeds␣at␣
     Tillamook");grid(lwd = 2)
dev.off()
png('acf,pacf.png')
plota(dat,h = 'full')
dev.off()
sink("adf.txt")
adf.test(dat, k = 288)
sink()

#---------SPLITTING DATA INTO TRAINING AND TESTING------------------------------
smp_size <- floor(6/7 * length(dat))
train_dat <- window(dat, end = smp_size)
test_dat <- window(dat, start = smp_size+1)
train_dates <- window(dates, end = smp_size)
test_dates <- window(dates, start = smp_size+1)
adf.test(train_dat, k = 288)

#----------MODEL FITTING USING TRAINING DATA------------------------------------
l <- length(test_dates)
detach(package:itsmr)
train_dat <- ts(train_dat, frequency = freq)
fit2 <- HoltWinters(train_dat, start.periods = 2)
f2 <- forecast(fit2, h = l)
fit1 <- stlm(train_dat, s.window = 2, method = "arima", lambda = 'auto', biasadj = TRUE)
f1 <- forecast(fit1, h = l)
sink("fit1.txt")
fit1$model
sink()
sink("fit2.txt")
fit2
sink()

pred1 <- ts(f1$mean, frequency = 1, start = smp_size+1, end = smp_size+288)
pred_er1 <- test_dat-pred1
pred2 <- ts(f2$mean, frequency = 1, start = smp_size+1, end = smp_size+288)
pred_er2 <- test_dat-pred2
#----------------PLOTTING THE FIT AND FORECASTS------------------------------------
png('stlm.png')
par(mfrow=c(2,1))
plot(dates, c(train_dat,ts(NA, start = 1, end = 288)), lwd = 2, main = "Forecasts␣from␣STL␣+␣ARIMA", type = 'l', ylab
     = "Wind␣speed␣in␣MPH", xlab = "Dates")
grid(lwd = 2)
polygon(c(test_dates,rev(test_dates)),c(f1$lower[,"95%"],rev(f1$upper[,"95%"])),col="grey", border = NA)
lines(train_dates,fit1$fitted, col = 'red')
lines(test_dates,f1$mean, col = 'blue', lwd = 1)
lines(test_dates,test_dat, col = 'black', lwd =1)
legend("bottomleft", legend=c("Data", "Fitted", "Predicted"), col=c("black", "red", "blue"),lty = 1:1,lwd = 2:2, cex=0
     .8)
plot(test_dates,test_dat, type = "l", ylab = "Wind␣speed␣in␣MPH", xlab = "Time,␣8␣Jan␣2013");grid(lwd = 2)
polygon(c(test_dates,rev(test_dates)),c(f1$lower[,"95%"],rev(f1$upper[,"95%"])),col="grey", border = NA)
lines(test_dates,pred1, col = "blue")
lines(test_dates,test_dat, col = "black")
legend("topleft", legend=c("Data", "Predicted"), col=c("black", "blue"),lty = 1:1,lwd = 2:2, cex=0.8)
dev.off()

png('hw.png')
par(mfrow=c(2,1))
```

```
plot(dates, ylim = c(-30,40) , c(train_dat,ts(NA, start = 1, end = 288)), lwd = 2, main = "Forecasts␣from␣HoltWinters"
     , type = 'l', ylab = "Wind␣speed␣in␣MPH", xlab = "Dates")
grid(lwd = 2)
polygon(c(test_dates,rev(test_dates)),c(f2$lower[,"95%"],rev(f2$upper[,"95%"])),col="grey", border = NA)
lines(train_dates,f2$fitted, col = 'red')
lines(test_dates,f2$mean, col = 'blue', lwd = 1)
lines(test_dates,test_dat, col = 'black', lwd =1)
legend("bottomleft", legend=c("Data", "Fitted", "Predicted"), col=c("black", "red", "blue"),lty = 1:1,lwd = 2:2, cex=0
     .8)
plot(test_dates,test_dat, type = "l", ylab = "Wind␣speed␣in␣MPH", xlab = "Time,␣8␣Jan␣2013", ylim = c(-30,40));grid(
     lwd = 2)
polygon(c(test_dates,rev(test_dates)),c(f2$lower[,"95%"],rev(f2$upper[,"95%"])),col="grey", border = NA)
lines(test_dates,pred2, col = "blue")
lines(test_dates,test_dat, col = "black")
legend("topleft", legend=c("Data", "Predicted"), col=c("black", "blue"),lty = 1:1,lwd = 2:2, cex=0.8)
dev.off()

#----------RESIDUE EVALUATION-----------------------------------------------------
res1 <- ts(fit1$residuals, frequency = 1)
res2 <- ts(train_dat-fit2$fitted[,"xhat"], frequency = 1)
library(itsmr)

attach(mtcars)
png('res1.png')
par(mfrow=c(3,1))
spec.pgram(res1, k=kernel("daniell", 40),taper=0,log="no",ylim=c(0,0.1))
acf(res1, lag.max = 'full', ylim=range(-1,1))
pacf(res1, lag.max = 'full', ylim=range(-1,1))
dev.off()
sink("adfres1.txt")
adf.test(res1, k = 288)
sink()

png('res2.png')
par(mfrow=c(3,1))
spec.pgram(res2, k=kernel("daniell", 40),taper=0,log="no")
acf(res2, lag.max = 'full', ylim=range(-1,1))
pacf(res2, lag.max = 'full', ylim=range(-1,1))
dev.off()
sink("adfres2.txt")
adf.test(res2, k = 288)
sink()

#----------FORECAST EVALUATION--------------------------------

sink("pred1.txt")
cat("Mean␣of␣the␣difference␣between␣Prediciton␣and␣Test␣set:␣", mean(pred_er1), "\n")
cat("Variance␣of␣the␣difference␣between␣Prediciton␣and␣Test␣set:␣", var(pred_er1), "\n")
cat("Correlation␣between␣Prediciton␣and␣Test␣set:␣", cor(pred1, test_dat), "\n")
sink()

sink("pred2.txt")
cat("Mean␣of␣the␣difference␣between␣Prediciton␣and␣Test␣set:␣", mean(pred_er2), "\n")
cat("Variance␣of␣the␣difference␣between␣Prediciton␣and␣Test␣set:␣", var(pred_er2), "\n")
cat("Correlation␣between␣Prediciton␣and␣Test␣set:␣", cor(pred2, test_dat), "\n")
sink()

#-------------FORECAST FOR JAN 9 2013---------------------
smp_size <- floor(7/7 * length(dat))
train_dat <- window(dat, end = smp_size)
train_dates <- window(dates, end = smp_size)

test_dates <- seq.POSIXt(as.POSIXct(c("2013-01-09")), as.POSIXct(c("2013-01-10")), by = "5␣min")
test_dates <- test_dates[1:288]
l <- length(test_dates)
detach(package:itsmr)
train_dat <- ts(train_dat, frequency = freq)
fit2 <- HoltWinters(train_dat, start.periods = 2)
f2 <- forecast(fit2, h = l)
fit1 <- stlm(train_dat, s.window = 2, method = "arima", lambda = 'auto', biasadj = TRUE)
f1 <- forecast(fit1, h = l)

pred1 <- ts(f1$mean, frequency = 1, start = smp_size+1, end = smp_size+288)
pred2 <- ts(f2$mean, frequency = 1, start = smp_size+1, end = smp_size+288)
```

```
#----------------PLOTTING THE FIT AND FORECASTS------------------------------------
png('stlm_pred.png')
par(mfrow=c(2,1))
plot(c(dates,test_dates), c(train_dat,ts(NA, start = 1, end = 288)), lwd = 2, main = "Forecasts␣from␣STL␣+␣ARIMA",
      type = 'l', ylab = "Wind␣speed␣in␣MPH", xlab = "Dates")
grid(lwd = 2)
polygon(c(test_dates,rev(test_dates)),c(f1$lower[,"95%"],rev(f1$upper[,"95%"])),col="grey", border = NA)
lines(train_dates,fit1$fitted, col = 'red')
lines(test_dates,f1$mean, col = 'blue', lwd = 1)
legend("bottomleft", legend=c("Data", "Fitted", "Forecast"), col=c("black", "red", "blue"),lty = 1:1,lwd = 2:2, cex=0.
      8)
plot(test_dates,pred1, col = "blue", type = "l", ylab = "Wind␣speed␣in␣MPH", xlab = "Time,␣9␣Jan␣2013", ylim = c(1,19)
      );grid(lwd = 2)
polygon(c(test_dates,rev(test_dates)),c(f1$lower[,"95%"],rev(f1$upper[,"95%"])),col="grey", border = NA)
lines(test_dates,pred1, col = "blue", type = "l", lwd = 2);grid(lwd = 2)
dev.off()

png('hw_pred.png')
par(mfrow=c(2,1))
plot(c(dates,test_dates), c(train_dat,ts(NA, start = 1, end = 288)), ylim =c(-30,40) , lwd = 2, main = "Forecasts␣from
      ␣Holt␣Winters", type = 'l', ylab = "Wind␣speed␣in␣MPH", xlab = "Dates")
grid(lwd = 2)
polygon(c(test_dates,rev(test_dates)),c(f2$lower[,"95%"],rev(f2$upper[,"95%"])),col="grey", border = NA)
lines(train_dates,f2$fitted, col = 'red')
lines(test_dates,f2$mean, col = 'blue', lwd = 1)
legend("bottomleft", legend=c("Data", "Fitted", "Forecast"), col=c("black", "red", "blue"),lty = 1:1,lwd = 2:2, cex=0.
      8)
plot(test_dates,pred2, col = "blue", type = "l", ylab = "Wind␣speed␣in␣MPH", xlab = "Time,␣9␣Jan␣2013", ylim = c(-30,4
      0));grid(lwd = 2)
polygon(c(test_dates,rev(test_dates)),c(f2$lower[,"95%"],rev(f2$upper[,"95%"])),col="grey", border = NA)
lines(test_dates,pred2, col = "blue", type = "l", lwd = 2);grid(lwd = 2)
dev.off()
closeAllConnections()

#----------------------------------------------------------
```