645 final

Zijing Gao

2019/11/26

```r
library('dplyr')

##
## 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

library('ggplot2')
library('forecast')

## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo

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

## Registered S3 methods overwritten by 'forecast':
##   method             from
##   fitted.fracdiff    fracdiff
##   residuals.fracdiff fracdiff

library('tseries')
```

objective

plot, examine, and prepare series for modeling

```r
# read data
dat = read.csv("weatherHistory.csv",header = T)
```

Now, I need to fetch and manage the data.

At first, I need t remove some columns that we do not need at all.

```r
# check colnames
colnames(dat)

##  [1] "Formatted.Date"           "Summary"
##  [3] "Precip.Type"              "Temperature..C."
##  [5] "Apparent.Temperature..C." "Humidity"
##  [7] "Wind.Speed..km.h."        "Wind.Bearing..degrees."
##  [9] "Visibility..km."          "Loud.Cover"
## [11] "Pressure..millibars."     "Daily.Summary"
## [13] "Precip.Value"

# remove the columns we may not need.
dat.remove = subset(dat, select=-c(Loud.Cover,Daily.Summary,Summary, Precip.T
ype))

colnames(dat.remove)

## [1] "Formatted.Date"           "Temperature..C."
## [3] "Apparent.Temperature..C." "Humidity"
## [5] "Wind.Speed..km.h."        "Wind.Bearing..degrees."
## [7] "Visibility..km."          "Pressure..millibars."
## [9] "Precip.Value"

# order the data by date.

dat.order = dat.remove[order(as.Date(dat.remove$Formatted.Date, format="%Y-%m
-%d")),]
# write.csv(new_data,file = "645_final_data.csv")

# we format the date in a better way so that we can use dplyr to select the s
ubset of the data that we may use.

dat.order$Formatted.Date = as.POSIXct(dat.order$Formatted.Date,format = "%Y-%
m-%d %H:%M:%S")


dat.subset = dat.order%>%
  select(Formatted.Date, Temperature..C., Humidity, Wind.Speed..km.h., Wind.B
earing..degrees., Visibility..km., Pressure..millibars.)%>%
  filter(Formatted.Date > "2016-08-01 02:00:00 EST" &
           Formatted.Date < "2016-08-31 02:00:00 EST")

head(dat.subset)

##        Formatted.Date Temperature..C. Humidity Wind.Speed..km.h.
## 1 2016-08-01 03:00:00        19.88333     0.88           19.8513
## 2 2016-08-01 04:00:00        19.81667     0.88           10.6099
## 3 2016-08-01 05:00:00        19.74444     0.89            9.4668
## 4 2016-08-01 06:00:00        18.84444     0.93           12.5580
## 5 2016-08-01 07:00:00        19.93333     0.88            9.5795
## 6 2016-08-01 08:00:00        21.07778     0.87           14.0553
```

```
##   Wind.Bearing..degrees. Visibility..km. Pressure..millibars.
## 1                    340         15.8263              1012.58
## 2                      0         16.1000              1011.40
## 3                     57         14.9569              1010.84
## 4                    301          9.9015              1011.12
## 5                    301         15.8263              1012.64
## 6                    313          9.9820              1013.06
```

```
dim(dat.subset)
```

```
## [1] 719   7
```

use correlation matrix to find explanatory variables

```
dat.cor = subset(dat.remove, select=-c(Formatted.Date))
corr = cor(dat.cor)
corr
```

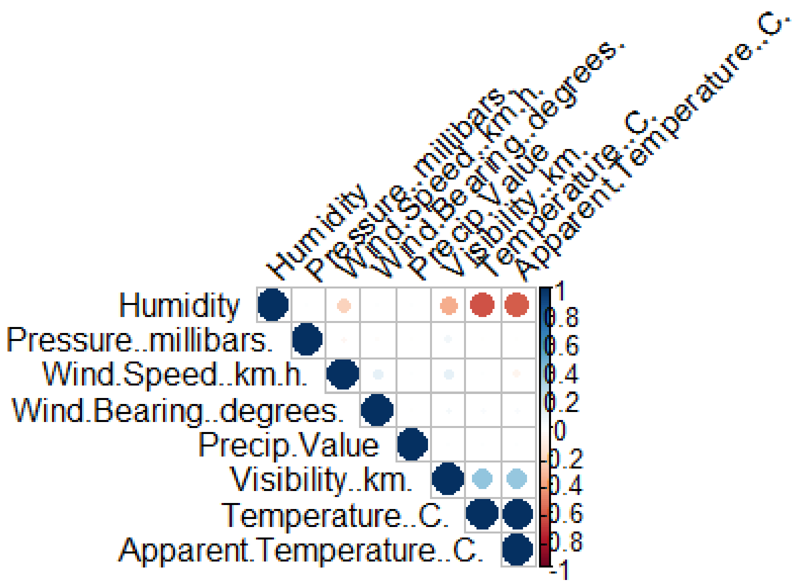
```
##                          Temperature..C. Apparent.Temperature..C.
## Temperature..C.             1.0000000000             0.9926285642
## Apparent.Temperature..C.    0.9926285642             1.0000000000
## Humidity                   -0.6322546750            -0.6025709956
## Wind.Speed..km.h.           0.0089569683            -0.0566496983
## Wind.Bearing..degrees.      0.0299882045             0.0290305198
## Visibility..km.             0.3928465717             0.3817184705
## Pressure..millibars.       -0.0054471062            -0.0002189998
## Precip.Value                0.0008619473             0.0010608217
##                               Humidity Wind.Speed..km.h.
## Temperature..C.          -0.6322546750       0.008956968
## Apparent.Temperature..C. -0.6025709956      -0.056649698
## Humidity                  1.0000000000      -0.224951456
## Wind.Speed..km.h.        -0.2249514559       1.000000000
## Wind.Bearing..degrees.    0.0007346454       0.103821508
## Visibility..km.          -0.3691725006       0.100749284
## Pressure..millibars.      0.0054542633      -0.049262806
## Precip.Value              0.0002169395       0.004804537
##                          Wind.Bearing..degrees. Visibility..km.
## Temperature..C.                    0.0299882045     0.392846572
## Apparent.Temperature..C.           0.0290305198     0.381718470
## Humidity                           0.0007346454    -0.369172501
## Wind.Speed..km.h.                  0.1038215077     0.100749284
## Wind.Bearing..degrees.             1.0000000000     0.047594175
## Visibility..km.                    0.0475941753     1.000000000
## Pressure..millibars.              -0.0116508848     0.059818381
## Precip.Value                      -0.0039252047     0.008057509
##                          Pressure..millibars.  Precip.Value
## Temperature..C.                 -0.0054471062  0.0008619473
## Apparent.Temperature..C.        -0.0002189998  0.0010608217
## Humidity                         0.0054542633  0.0002169395
## Wind.Speed..km.h.               -0.0492628055  0.0048045372
```

```
## Wind.Bearing..degrees.          -0.0116508848 -0.0039252047
## Visibility..km.                  0.0598183810  0.0080575088
## Pressure..millibars.             1.0000000000 -0.0096836303
## Precip.Value                    -0.0096836303  1.0000000000
```

```r
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
corr = cor(dat.cor)
corrplot(corr, type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45)
```



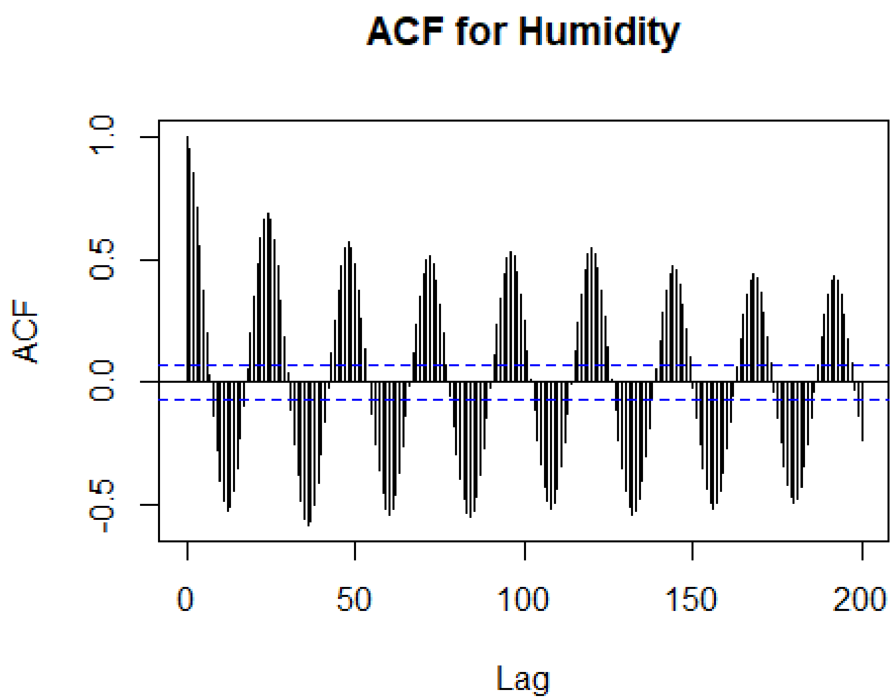

```r
corr
```

```
##                          Temperature..C. Apparent.Temperature..C.
## Temperature..C.             1.0000000000             0.9926285642
## Apparent.Temperature..C.    0.9926285642             1.0000000000
## Humidity                   -0.6322546750            -0.6025709956
## Wind.Speed..km.h.           0.0089569683            -0.0566496983
## Wind.Bearing..degrees.      0.0299882045             0.0290305198
## Visibility..km.             0.3928465717             0.3817184705
## Pressure..millibars.       -0.0054471062            -0.0002189998
## Precip.Value                0.0008619473             0.0010608217
##                              Humidity Wind.Speed..km.h.
## Temperature..C.          -0.6322546750       0.008956968
## Apparent.Temperature..C. -0.6025709956      -0.056649698
## Humidity                  1.0000000000      -0.224951456
```

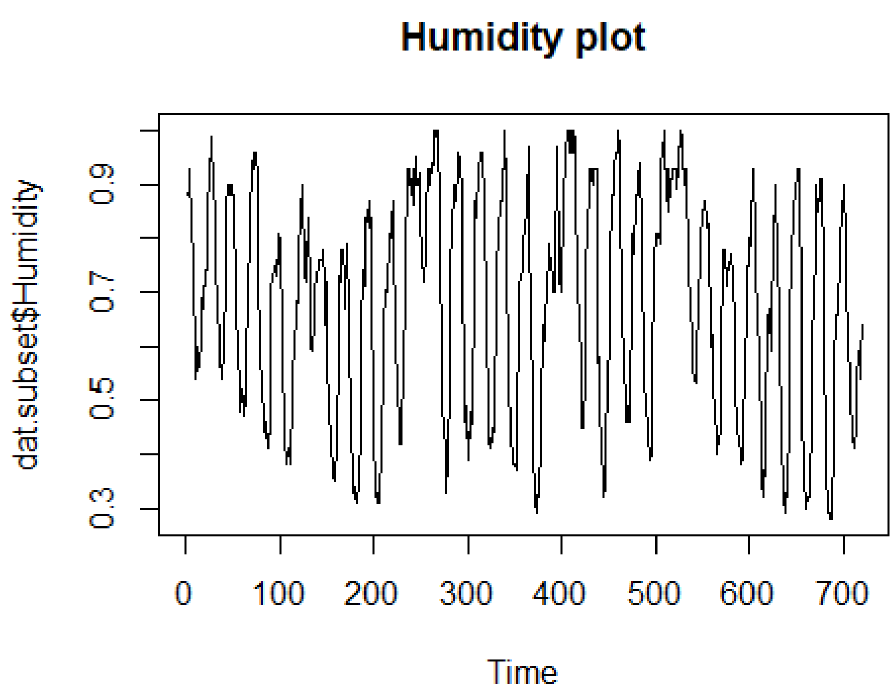
```
## Wind.Speed..km.h.        -0.2249514559       1.000000000
## Wind.Bearing..degrees.    0.0007346454       0.103821508
## Visibility..km.          -0.3691725006       0.100749284
## Pressure..millibars.      0.0054542633      -0.049262806
## Precip.Value              0.0002169395       0.004804537
##                          Wind.Bearing..degrees. Visibility..km.
## Temperature..C.                    0.0299882045     0.392846572
## Apparent.Temperature..C.           0.0290305198     0.381718470
## Humidity                           0.0007346454    -0.369172501
## Wind.Speed..km.h.                  0.1038215077     0.100749284
## Wind.Bearing..degrees.             1.0000000000     0.047594175
## Visibility..km.                    0.0475941753     1.000000000
## Pressure..millibars.              -0.0116508848     0.059818381
## Precip.Value                      -0.0039252047     0.008057509
##                          Pressure..millibars.  Precip.Value
## Temperature..C.                 -0.0054471062  0.0008619473
## Apparent.Temperature..C.        -0.0002189998  0.0010608217
## Humidity                         0.0054542633  0.0002169395
## Wind.Speed..km.h.               -0.0492628055  0.0048045372
## Wind.Bearing..degrees.          -0.0116508848 -0.0039252047
## Visibility..km.                  0.0598183810  0.0080575088
## Pressure..millibars.             1.0000000000 -0.0096836303
## Precip.Value                    -0.0096836303  1.0000000000

# choose temperature and visiblity
```

#Humidity, Temp, Visibility

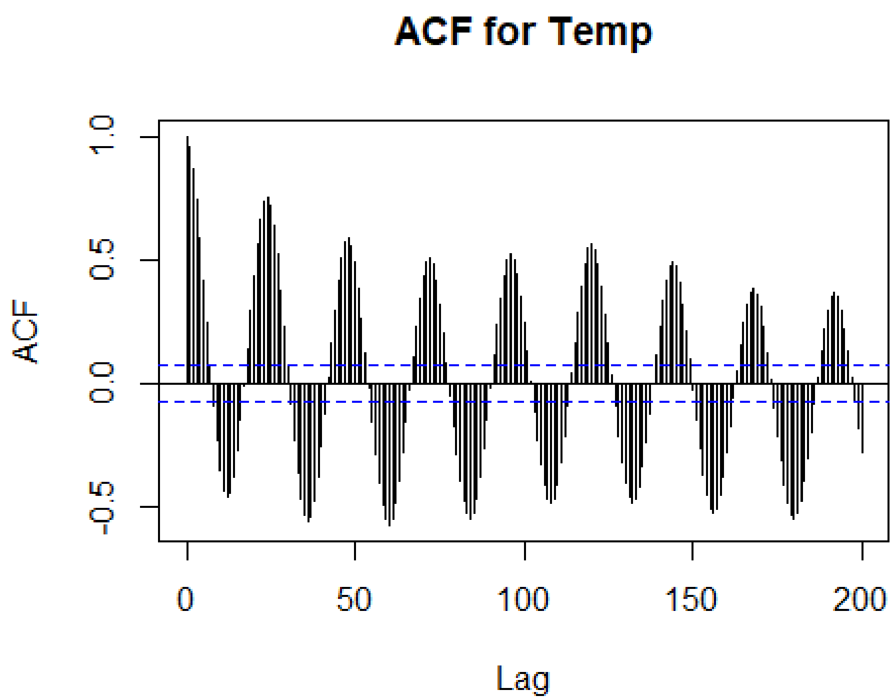
```
acf(dat.subset$Humidity, main = "ACF for Humidity", lag = 200)
```



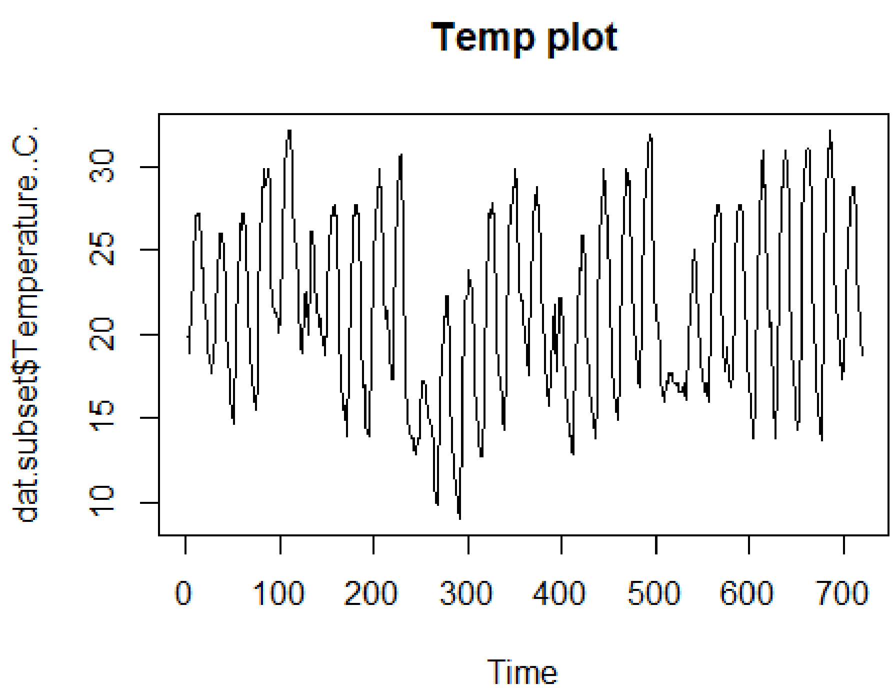

```
plot.ts(dat.subset$Humidity, main = "Humidity plot")
```



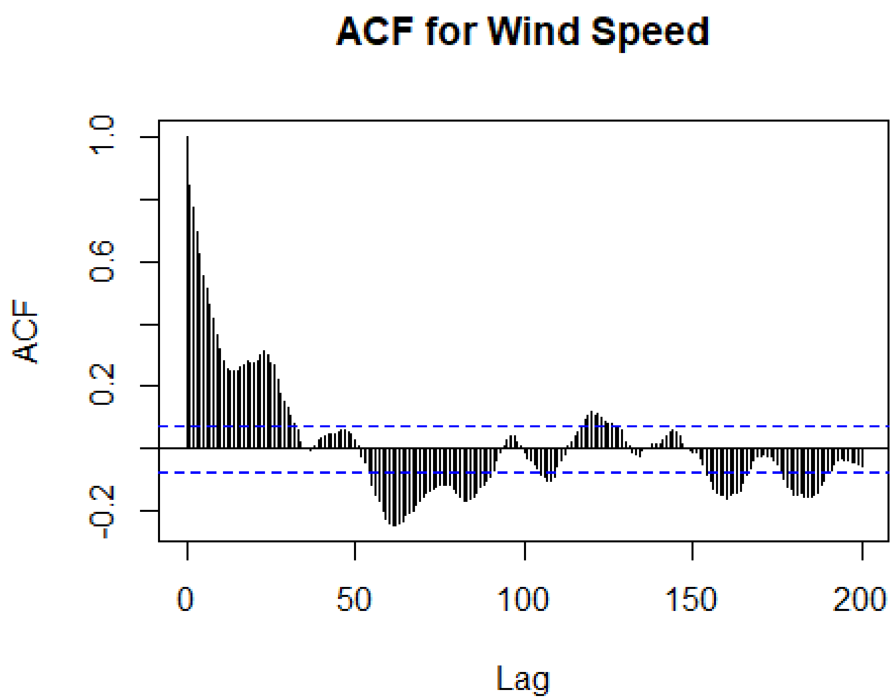

```
acf(dat.subset$Temperature..C., main = "ACF for Temp", lag = 200)
```



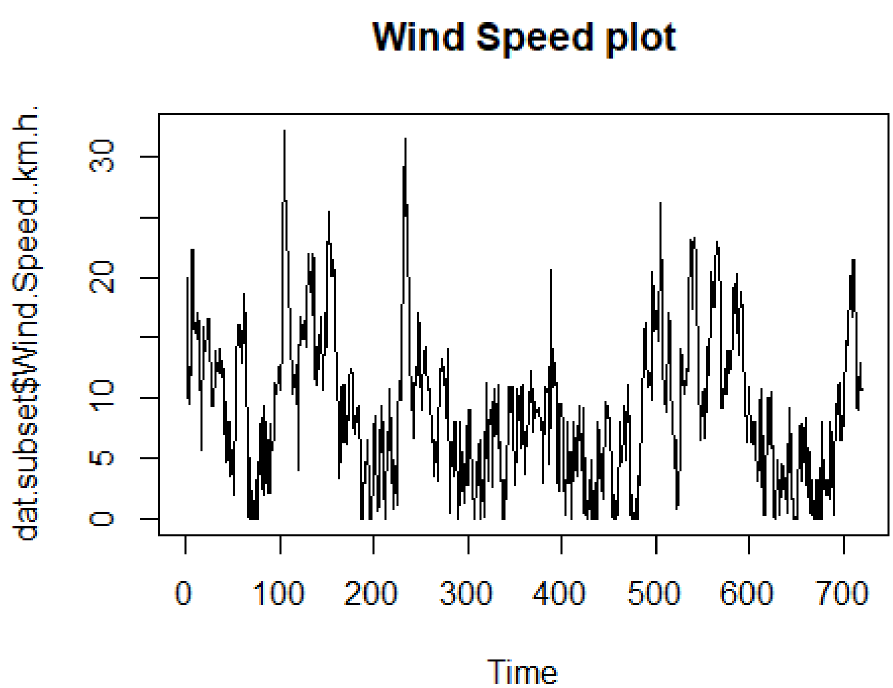

```r
plot.ts(dat.subset$Temperature..C., main = "Temp plot")
```




```r
acf(dat.subset$Wind.Speed..km.h., main = "ACF for Wind Speed", lag = 200)
```



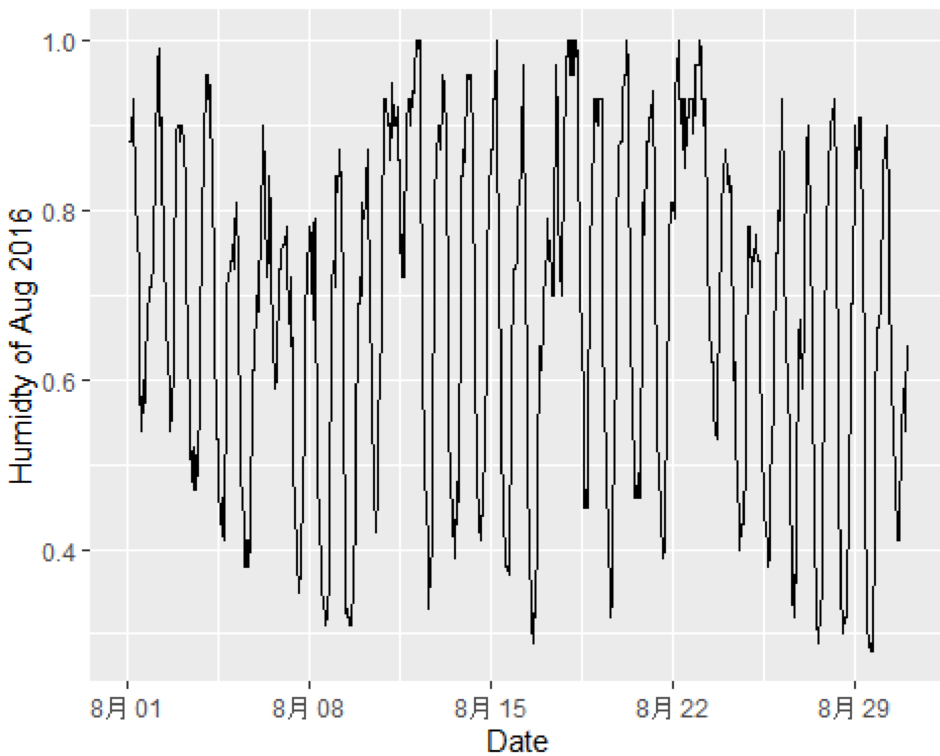

```r
plot.ts(dat.subset$Wind.Speed..km.h., main = "Wind Speed plot")
```



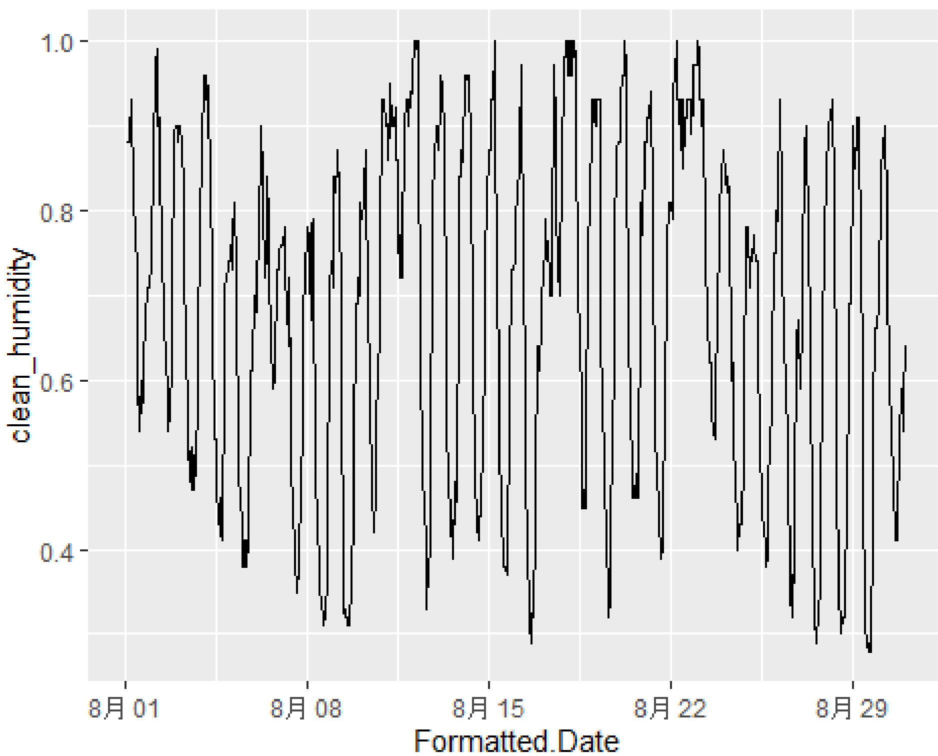
How to determine order of differencing needed?

step 2 Examine the data

```
# plot the humidity after cleaning the abnormal data if needed.

ggplot(dat.subset, aes(Formatted.Date, Humidity)) + geom_line() + xlab("Date")
 + ylab("Humidty of Aug 2016")
```




```
count_ts = ts(dat.subset[, c('Humidity')])

dat.subset$clean_humidity = tsclean(count_ts)

ggplot() +
  geom_line(data = dat.subset, aes(x = Formatted.Date, y = clean_humidity))

## Don't know how to automatically pick scale for object of type ts. Defaulti
ng to continuous.
```

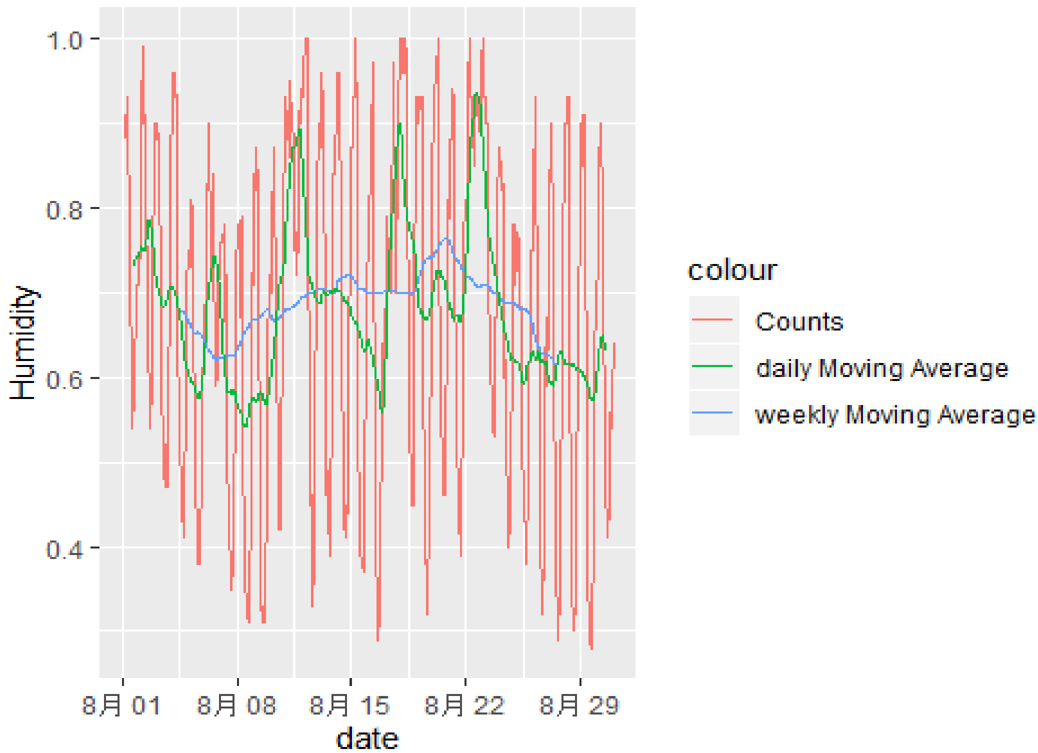


Smooth the data

Visually, we could a draw a line through the series tracing its bigger troughs and peaks while smoothing out noisy fluctuations. This line can be described by one of the simplest — but also very useful — concepts in time series analysis known as a moving average.

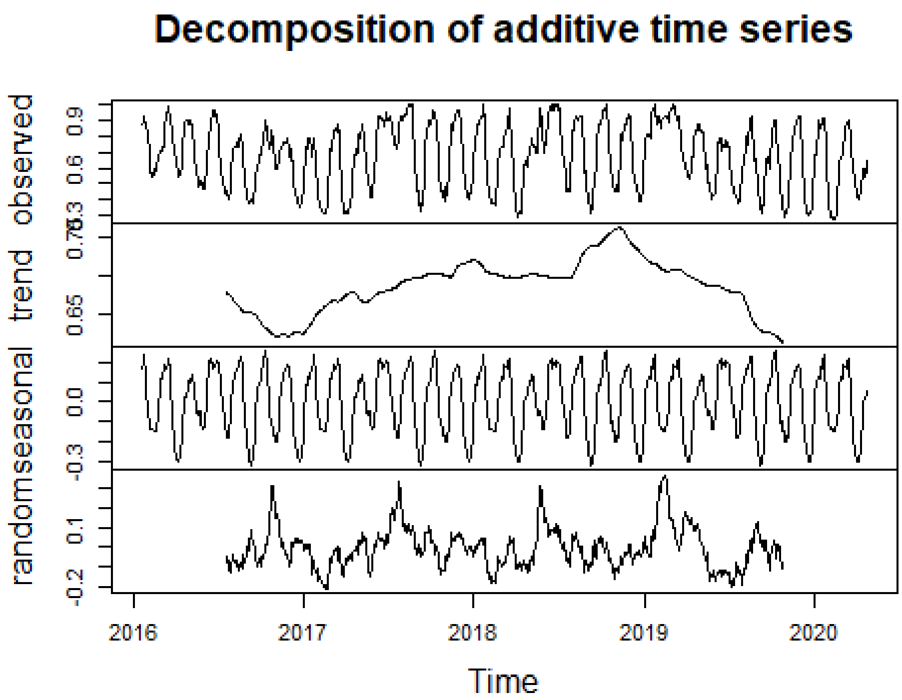
```
dat.subset$humid_ma_w = ma(dat.subset$clean_humidity, order=7*24) # using the
 clean count with no outliers
dat.subset$humid_ma_d = ma(dat.subset$clean_humidity, order=24)



ggplot() +
  geom_line(data = dat.subset, aes(x = Formatted.Date, y = clean_humidity, co
lour = "Counts")) +
  geom_line(data = dat.subset, aes(x = Formatted.Date, y = humid_ma_w,   colo
ur = "weekly Moving Average"))  +
  geom_line(data = dat.subset, aes(x = Formatted.Date, y = humid_ma_d, colour
 = "daily Moving Average"))  +
  ylab('Humidity')+
  xlab('date')

## Don't know how to automatically pick scale for object of type ts. Defaulti
ng to continuous.
```

```
## Warning: Removed 168 rows containing missing values (geom_path).

## Warning: Removed 24 rows containing missing values (geom_path).
```

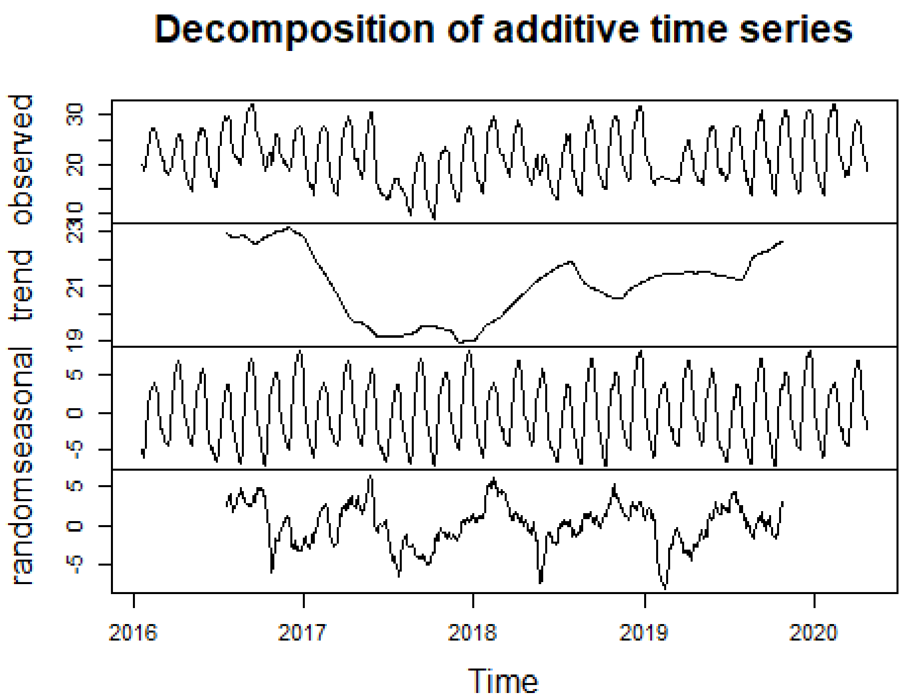


step 3: Decompose the data

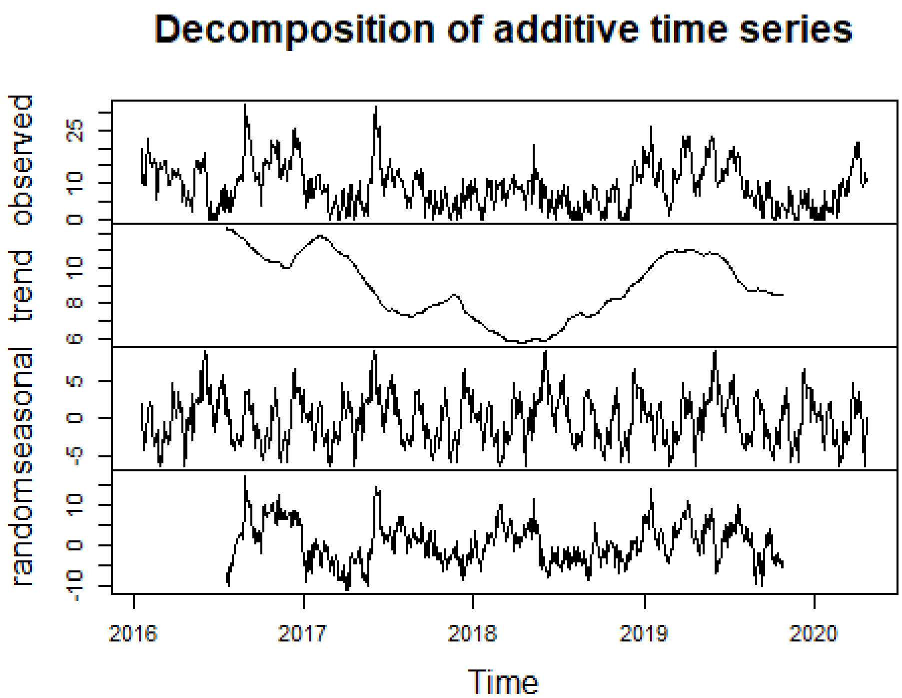
```
# Decompose the data

## Does the series appear to have trends or seasonality?


hum.ts = ts(dat.subset$Humidity,start = c(2016,8),frequency = 24*7)

d = decompose(hum.ts)
plot(d)
```




```
temp.ts = ts(dat.subset$Temperature..C.,start = c(2016,8),frequency = 24*7)

d.temp = decompose(temp.ts)
plot(d.temp)
```

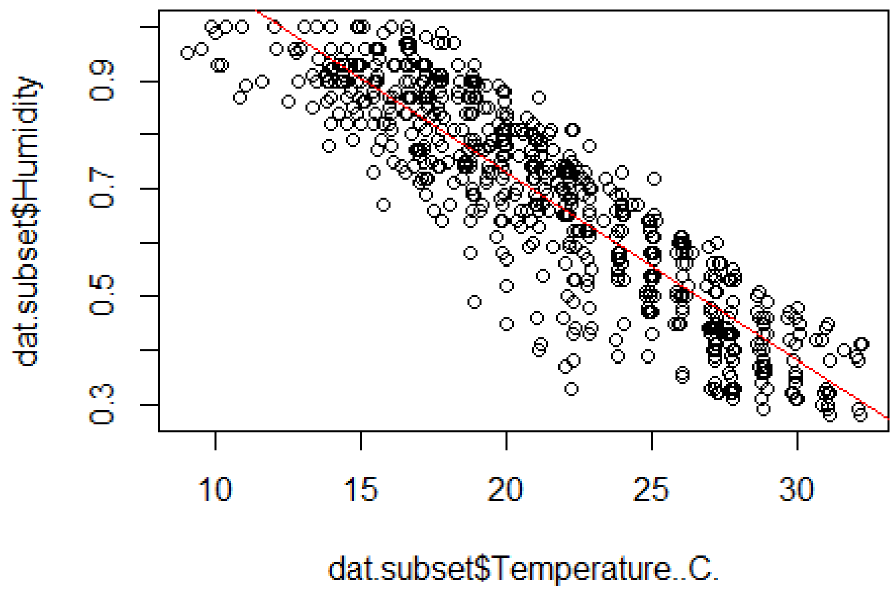



```
wind.ts = ts(dat.subset$Wind.Speed..km.h.,start = c(2016,8),frequency = 24*7)

d.wind = decompose(wind.ts)
plot(d.wind)
```

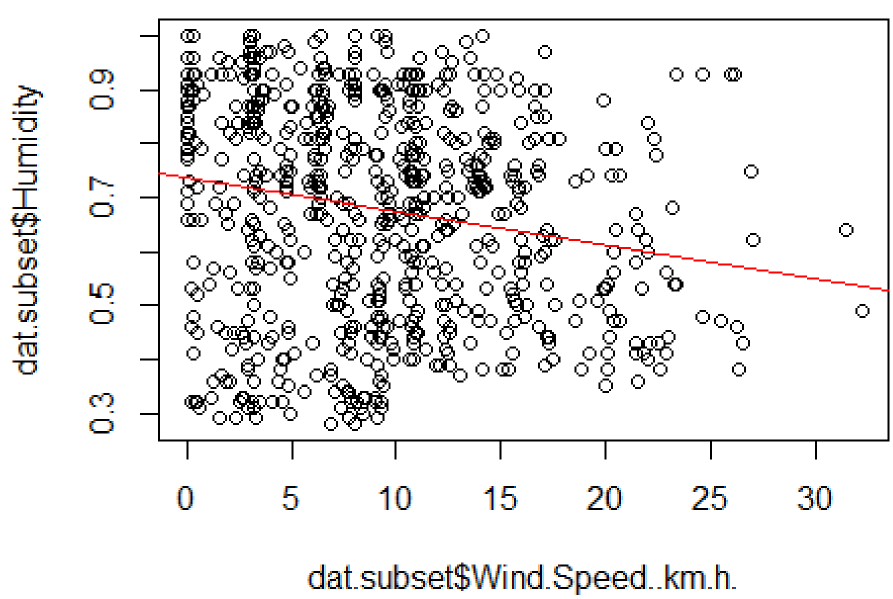


relationships

```
fit.temp = lm(dat.subset$Humidity ~ dat.subset$Temperature..C.)
plot(y = dat.subset$Humidity, x = dat.subset$Temperature..C., type = "p")
abline(fit.temp, col = "red")
```

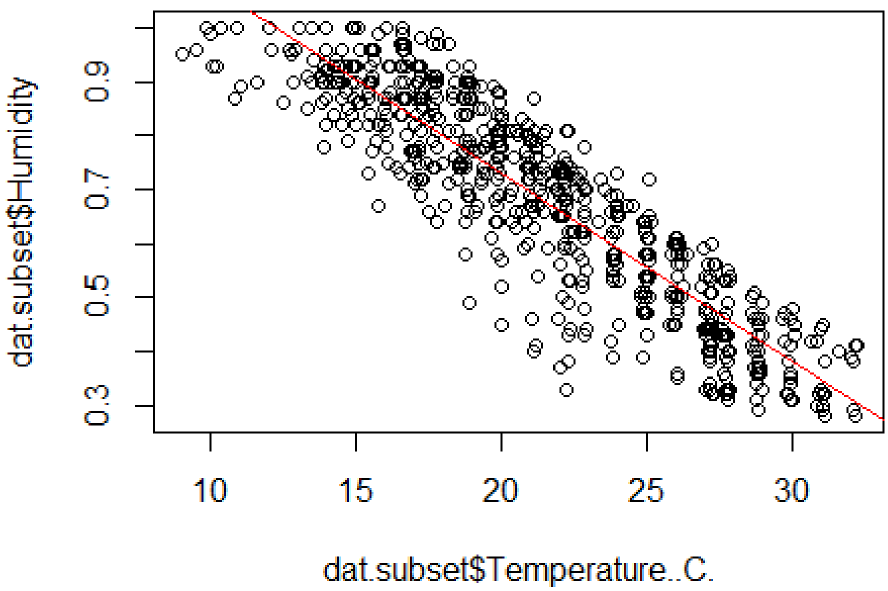



```
fit.wind = lm(dat.subset$Humidity ~ dat.subset$Wind.Speed..km.h.)
  plot(y = dat.subset$Humidity, x = dat.subset$Wind.Speed..km.h., type = "p")
abline(fit.wind, col = "red")
```



regression models

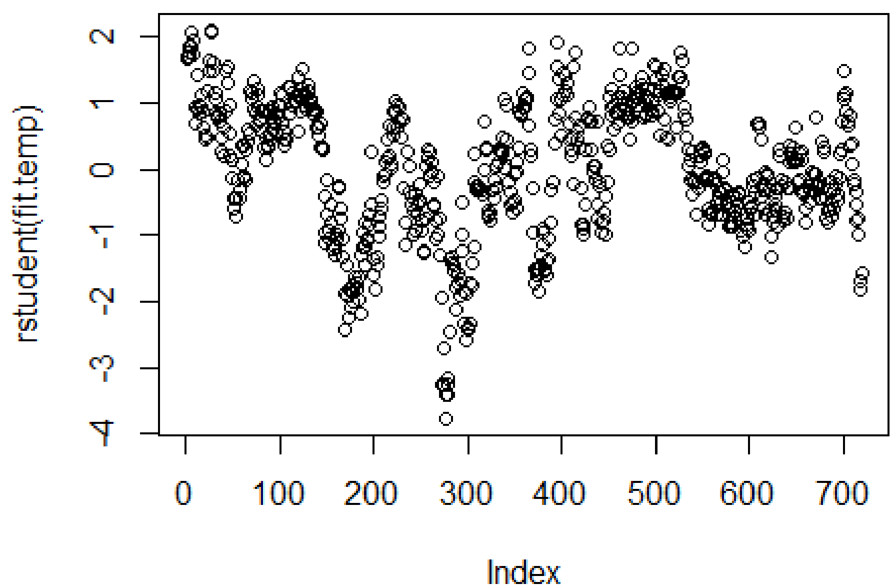
```r
library(lmtest)

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

fit.temp = lm(dat.subset$Humidity ~ dat.subset$Temperature..C.)
plot(y = dat.subset$Humidity, x = dat.subset$Temperature..C., type = "p")
abline(fit.temp, col = "red")
```




```r
summary(fit.temp)

##
## Call:
## lm(formula = dat.subset$Humidity ~ dat.subset$Temperature..C.)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32253 -0.05610  0.00483  0.07141  0.18057
##
## Coefficients:
```

```
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  1.4292855  0.0137976  103.59   <2e-16 ***
## dat.subset$Temperature..C. -0.0349542  0.0006258  -55.85   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08626 on 717 degrees of freedom
## Multiple R-squared:  0.8131, Adjusted R-squared:  0.8129
## F-statistic:  3120 on 1 and 717 DF,  p-value: < 2.2e-16
```

```
plot(rstudent(fit.temp))
```




```
dwtest(fit.temp) # < dL: reject H0
```

```
##
##  Durbin-Watson test
##
## data:  fit.temp
## DW = 0.13645, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is greater than 0
```

```
res.temp = fit.temp$residuals
rnum.temp = 0
rdenom.temp = 0
for (i in 2: 719){
  rnum.temp = rnum.temp + res.temp[i] * res.temp[i-1]
  rdenom.temp = rdenom.temp + res.temp[i-1]^2
```

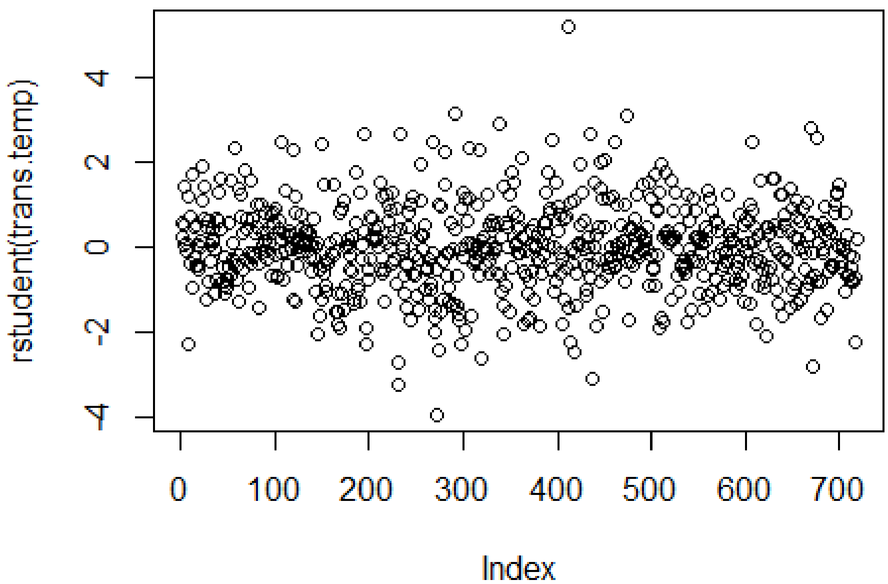
```
}
rhat.temp = rnum.temp / rdenom.temp
x.temp = rep(0, 718)
y.temp = rep(0, 718)
for (i in 1: 718){
  y.temp[i] = dat.subset$Humidity[i+1] - rhat.temp * dat.subset$Humidity[i]
  x.temp[i] = dat.subset$Temperature..C.[i+1] - rhat.temp * dat.subset$Temper
ature..C.[i]
}
trans.temp = lm(y.temp ~ x.temp)
summary(trans.temp)
```

```
##
## Call:
## lm(formula = y.temp ~ x.temp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12066 -0.01798 -0.00032  0.01674  0.15744
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.1021176  0.0016646   61.35   <2e-16 ***
## x.temp      -0.0378432  0.0008116  -46.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03106 on 716 degrees of freedom
## Multiple R-squared:  0.7523, Adjusted R-squared:  0.7519
## F-statistic:  2174 on 1 and 716 DF,  p-value: < 2.2e-16
```

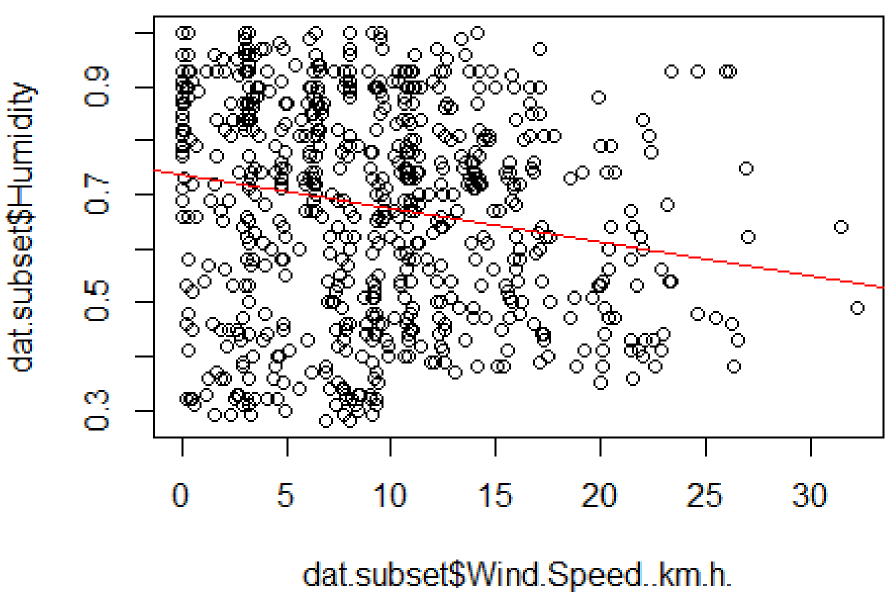
```
dwtest(trans.temp) # past DW test
```

```
##
##  Durbin-Watson test
##
## data:  trans.temp
## DW = 2.119, p-value = 0.942
## alternative hypothesis: true autocorrelation is greater than 0
```

```
plot(rstudent(trans.temp))
```



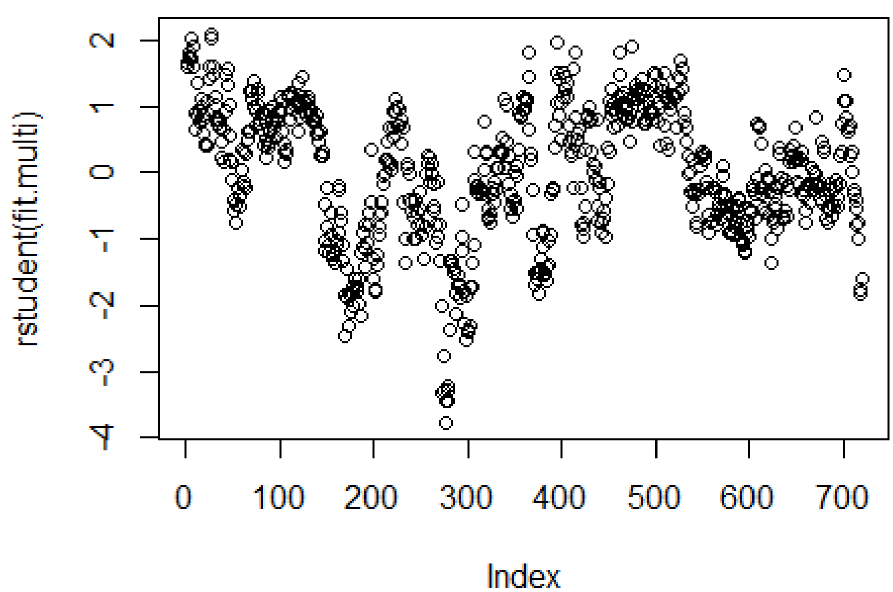

```
fit.wind = lm(dat.subset$Humidity ~ dat.subset$Wind.Speed..km.h.)
plot(y = dat.subset$Humidity, x = dat.subset$Wind.Speed..km.h., type = "p")
abline(fit.wind, col = "red")
```



```r
fit.multi = lm(dat.subset$Humidity ~ dat.subset$Temperature..C.+dat.subset$Wi
nd.Speed..km.h.)
summary(fit.multi)
```

```
##
## Call:
## lm(formula = dat.subset$Humidity ~ dat.subset$Temperature..C. +
##     dat.subset$Wind.Speed..km.h.)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32181 -0.05524  0.00394  0.07045  0.17935
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   1.4269926  0.0138635 102.932   <2e-16 ***
## dat.subset$Temperature..C.   -0.0351949  0.0006443 -54.629   <2e-16 ***
## dat.subset$Wind.Speed..km.h.  0.0008231  0.0005315   1.549    0.122
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08618 on 716 degrees of freedom
## Multiple R-squared:  0.8137, Adjusted R-squared:  0.8132
## F-statistic:  1564 on 2 and 716 DF,  p-value: < 2.2e-16
```

```r
plot(rstudent(fit.multi))
```



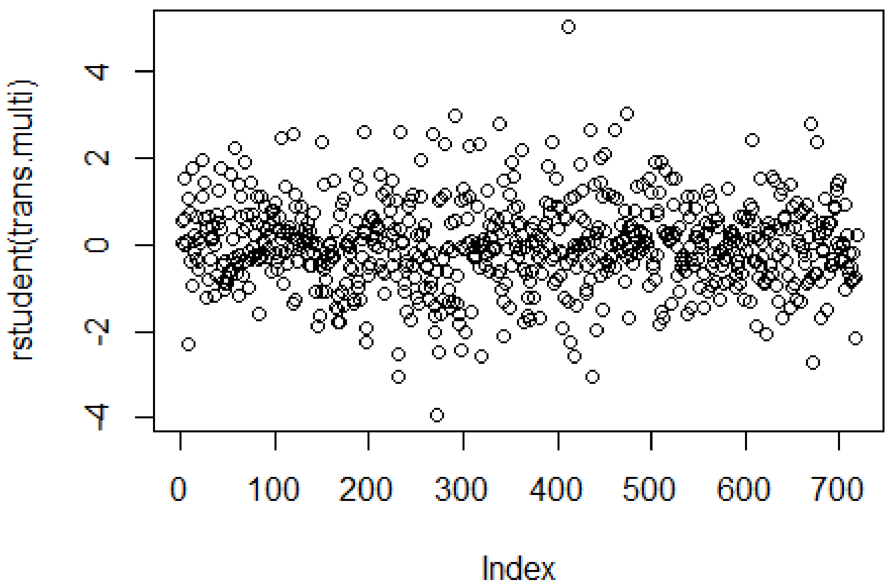
```r
res.multi = fit.multi$residuals
rnum.multi = 0
rdenom.multi = 0
for (i in 2: 719){
  rnum.multi = rnum.multi + res.multi[i] * res.multi[i-1]
  rdenom.multi = rdenom.multi + res.multi[i-1]^2
}
rhat.multi = rnum.multi / rdenom.multi
x.multi = rep(0, 718)
z.multi = rep(0, 718)
y.multi = rep(0, 718)
for (i in 1: 718){
  y.multi[i] = dat.subset$Humidity[i+1] - rhat.multi * dat.subset$Humidity[i]
  z.multi[i] = dat.subset$Wind.Speed..km.h.[i+1] - rhat.multi * dat.subset$Wi
nd.Speed..km.h.[i]
  x.multi[i] = dat.subset$Temperature..C.[i+1] - rhat.multi * dat.subset$Temp
erature..C.[i]
}
trans.multi = lm(y.multi ~ x.multi + z.multi)
summary(trans.multi)
```

```
##
## Call:
## lm(formula = y.multi ~ x.multi + z.multi)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.120251 -0.017728 -0.000327  0.016950  0.153022
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.1044699  0.0016796  62.198   <2e-16 ***
## x.multi     -0.0376190  0.0008147 -46.174   <2e-16 ***
## z.multi     -0.0007729  0.0003483  -2.219   0.0268 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03097 on 715 degrees of freedom
## Multiple R-squared:  0.7541, Adjusted R-squared:  0.7534
## F-statistic:  1096 on 2 and 715 DF,  p-value: < 2.2e-16
```

```r
dwtest(trans.multi) # past DW test
```

```
##
##  Durbin-Watson test
##
## data:  trans.multi
## DW = 2.1182, p-value = 0.9417
## alternative hypothesis: true autocorrelation is greater than 0
```

```r
plot(rstudent(trans.multi))
```

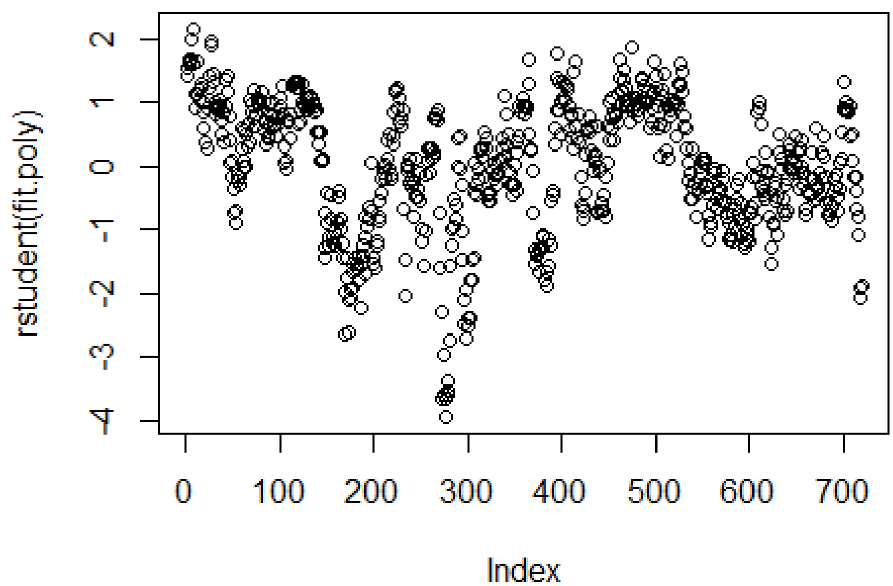



```r
library(ISLR)
fit.poly = lm(dat.subset$Humidity ~ poly(dat.subset$Temperature..C., degree =
 3) + poly(dat.subset$Wind.Speed..km.h., degree = 2))
summary(fit.poly)
```

```
##
## Call:
## lm(formula = dat.subset$Humidity ~ poly(dat.subset$Temperature..C.,
##     degree = 3) + poly(dat.subset$Wind.Speed..km.h., degree = 2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32331 -0.05084  0.00533  0.06992  0.17757
##
## Coefficients:
##                                                 Estimate Std. Error
## (Intercept)                                     0.679889   0.003095
## poly(dat.subset$Temperature..C., degree = 3)1   -4.847601   0.085613
## poly(dat.subset$Temperature..C., degree = 3)2   -0.340298   0.084281
## poly(dat.subset$Temperature..C., degree = 3)3    0.533649   0.083312
## poly(dat.subset$Wind.Speed..km.h., degree = 2)1  0.130584   0.086761
## poly(dat.subset$Wind.Speed..km.h., degree = 2)2  0.096205   0.083420
##                                                 t value Pr(>|t|)
## (Intercept)                                     219.671  < 2e-16 ***
## poly(dat.subset$Temperature..C., degree = 3)1   -56.622  < 2e-16 ***
## poly(dat.subset$Temperature..C., degree = 3)2    -4.038 5.98e-05 ***
## poly(dat.subset$Temperature..C., degree = 3)3     6.405 2.72e-10 ***
```

```
## poly(dat.subset$Wind.Speed..km.h., degree = 2)1   1.505    0.133
## poly(dat.subset$Wind.Speed..km.h., degree = 2)2   1.153    0.249
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08299 on 713 degrees of freedom
## Multiple R-squared:  0.828,  Adjusted R-squared:  0.8268
## F-statistic: 686.4 on 5 and 713 DF,  p-value: < 2.2e-16
```

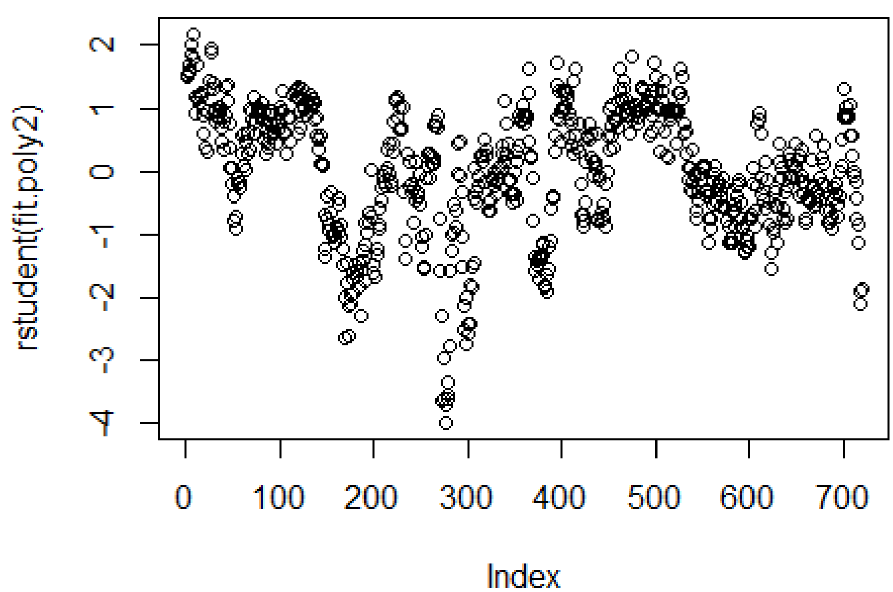
```
plot(rstudent(fit.poly))
```



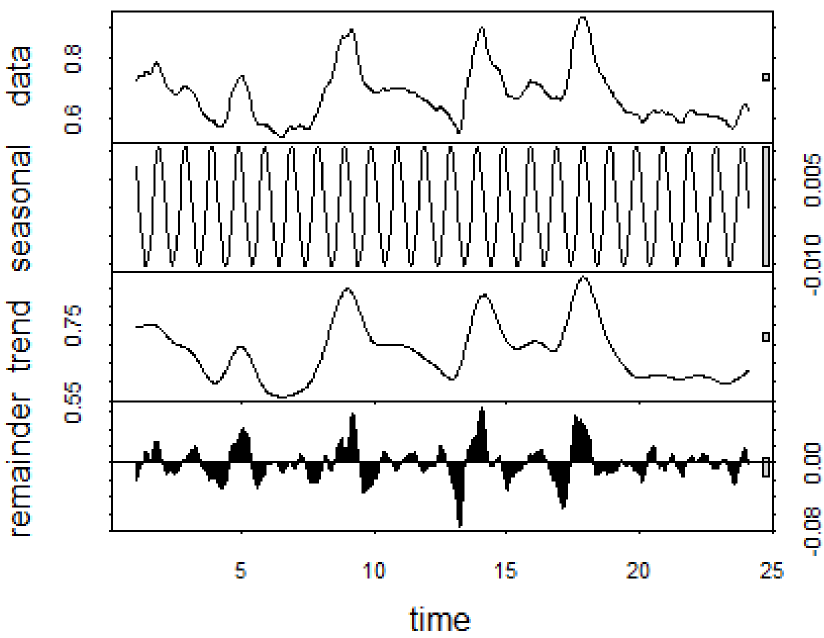

```
fit.poly2 = lm(dat.subset$Humidity ~ poly(dat.subset$Temperature..C., degree
= 3))
summary(fit.poly2)
```

```
##
## Call:
## lm(formula = dat.subset$Humidity ~ poly(dat.subset$Temperature..C.,
##     degree = 3))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32698 -0.05161  0.00645  0.06897  0.18020
##
## Coefficients:
##                                               Estimate Std. Error t value
## (Intercept)                                   0.679889   0.003099 219.415
```

```
## poly(dat.subset$Temperature..C., degree = 3)1 -4.817926   0.083087 -57.986
## poly(dat.subset$Temperature..C., degree = 3)2 -0.349889   0.083087  -4.211
## poly(dat.subset$Temperature..C., degree = 3)3  0.526063   0.083087   6.331
##                                               Pr(>|t|)
## (Intercept)                                    < 2e-16 ***
## poly(dat.subset$Temperature..C., degree = 3)1  < 2e-16 ***
## poly(dat.subset$Temperature..C., degree = 3)2 2.87e-05 ***
## poly(dat.subset$Temperature..C., degree = 3)3 4.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08309 on 715 degrees of freedom
## Multiple R-squared:  0.8271, Adjusted R-squared:  0.8264
## F-statistic:  1140 on 3 and 715 DF,  p-value: < 2.2e-16

plot(rstudent(fit.poly2))
```




```
count_humid_ma = ts(na.omit(dat.subset$humid_ma_d), frequency=30)
decomp = stl(count_humid_ma, s.window="periodic")
deseasonal_humid <- seasadj(decomp)
plot(decomp)
```




```
# we are specifying periodicity of the data, i.e., number of observations per
 period. Since we are using smoothed daily data, we have 30 observations per
month.

# We now have a de-seasonalized series
```

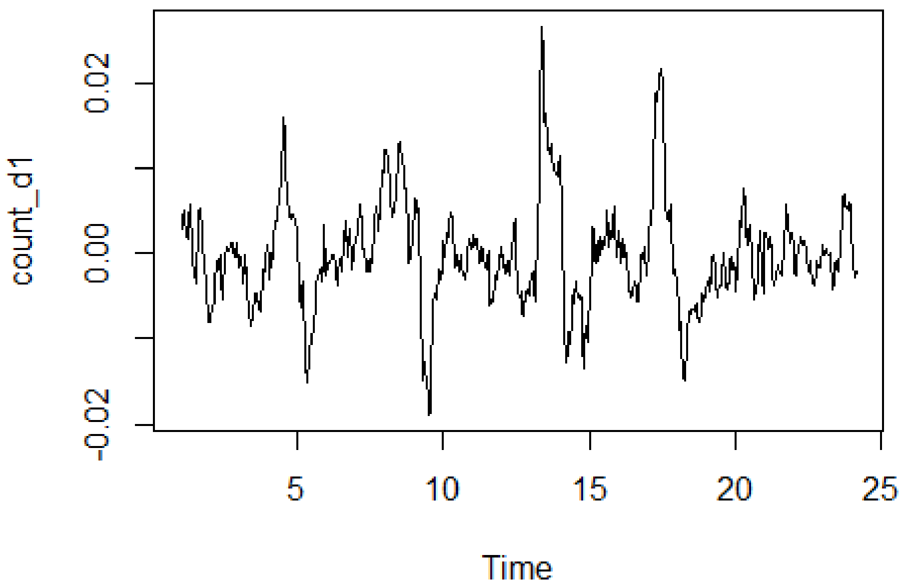
step 4: Stationary

Is the series stationary?

```
adf.test(count_humid_ma,alternative = "stationary")

##
##  Augmented Dickey-Fuller Test
##
## data:  count_humid_ma
## Dickey-Fuller = -3.658, Lag order = 8, p-value = 0.02711
## alternative hypothesis: stationary
```

step 5: Autocorrelations and Choosing Model Order

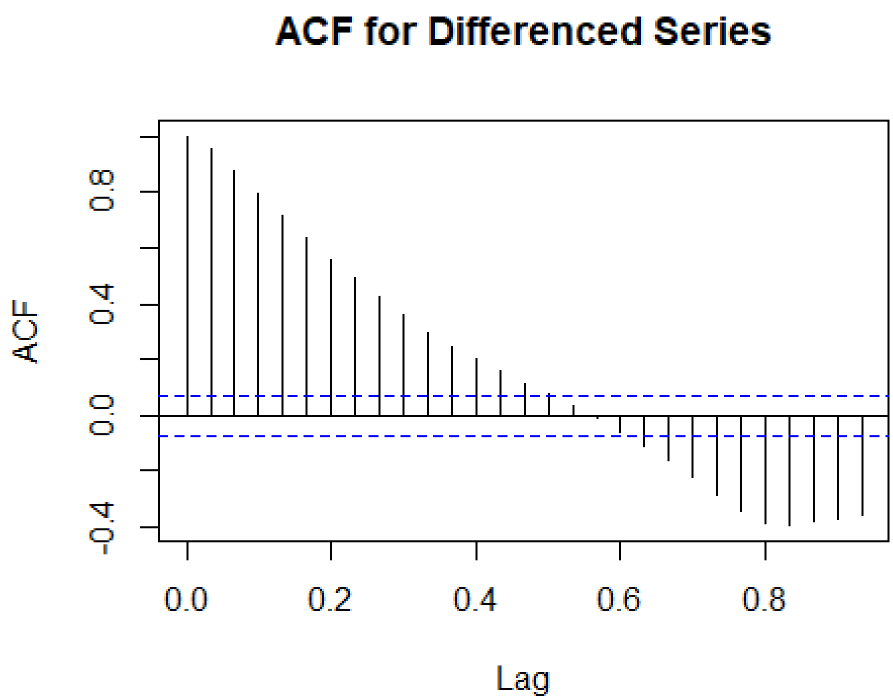
```
# We can start with the order of d = 1 and re-evaluate whether further differ
encing is needed.
count_d1 = diff(deseasonal_humid, differences = 1)
plot(count_d1)
```



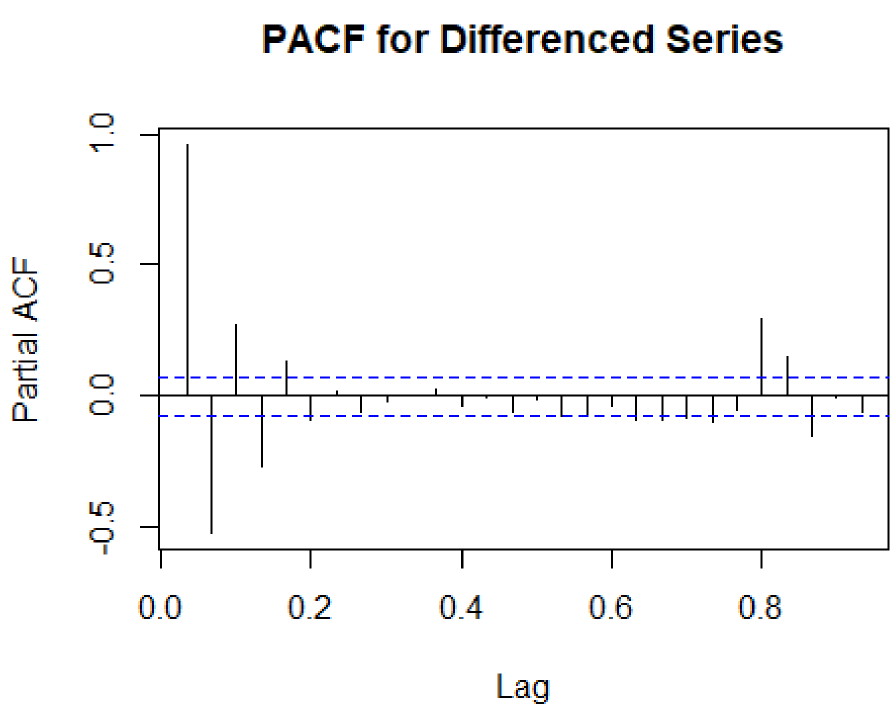

```
adf.test(count_d1, alternative = "stationary")

## Warning in adf.test(count_d1, alternative = "stationary"): p-value smaller
## than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  count_d1
## Dickey-Fuller = -5.7047, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

ACF and PACF

```
acf(count_d1, main='ACF for Differenced Series')
```




```r
pacf(count_d1, main='PACF for Differenced Series')
```



```
# There are significant and constantly decreasing auto correlations within one day from our hourly data and beyond.

# Partial correlation plots show a significant spike at lag 1 to 4.
```

Fitting an ARIMA model

```
# We can specify non-seasonal ARIMA structure and fit the model to de-seasonalize data. Parameters (1,1,2) suggested by the automated procedure are in line with our expectations based on the steps above
auto.arima(deseasonal_humid,seasonal = FALSE)
```

```
## Series: deseasonal_humid
## ARIMA(1,1,2)
##
## Coefficients:
##          ar1     ma1     ma2
##       0.9102  0.8232  0.0940
## s.e.  0.0170  0.0416  0.0412
##
## sigma^2 estimated as 1.836e-06:  log likelihood=3598.06
## AIC=-7188.12   AICc=-7188.06   BIC=-7169.95
```

Using the ARIMA notation introduced above, the fitted model can be written as

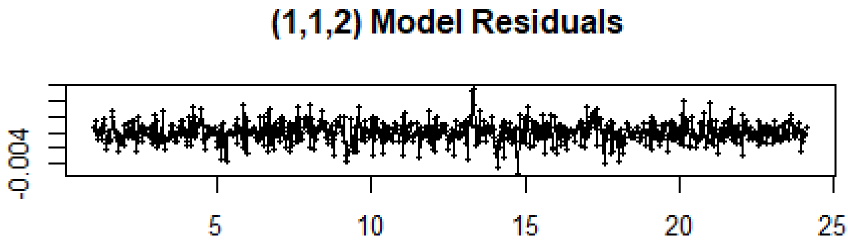
$$Y_{d_t} = 0.9102Y_{t-1} + 0.8232e_{t-1} + 0.0940e_{t-2} + \epsilon$$

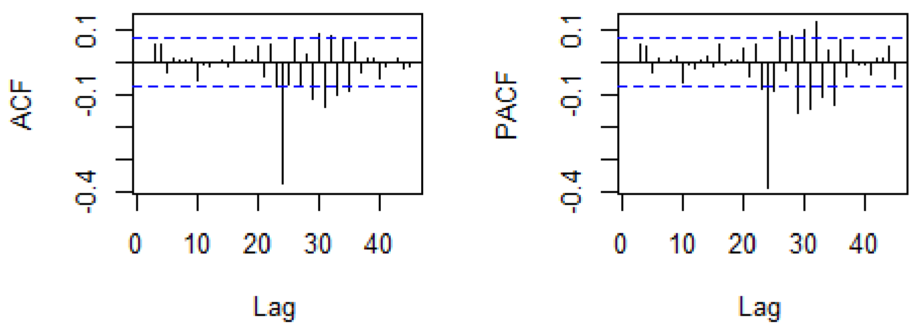
where ϵ is some error and the original series is differenced with order 1.

AR(1) coefficient p = 0.9102 tells us that the next value in the series is taken as a dampened previous value by a factor of 0.91 and depends on previous error lag.

step 7: Evaluate and Iterate

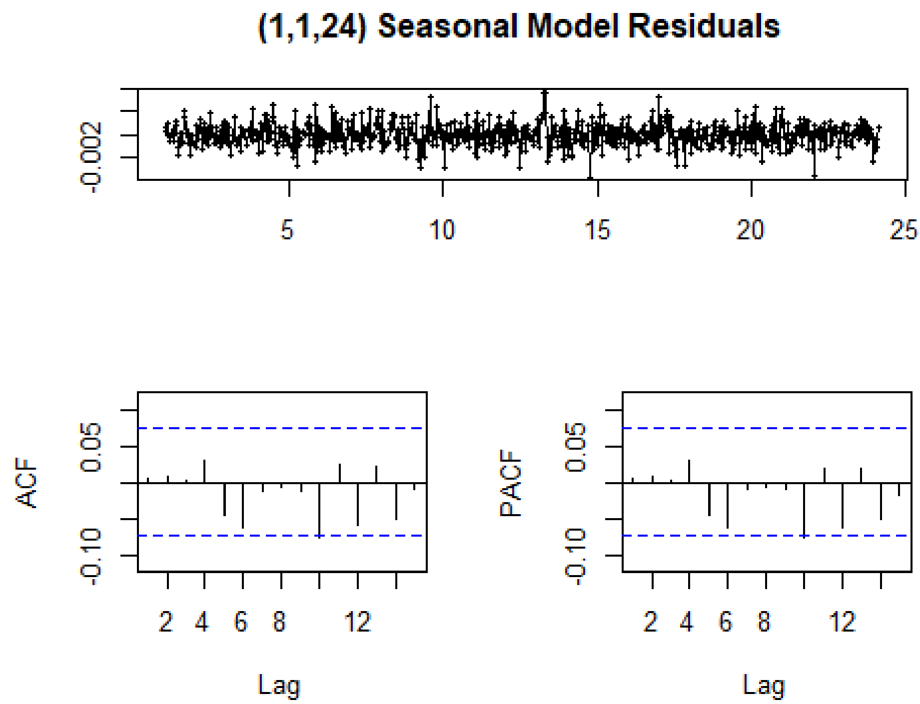
```
fit<-auto.arima(deseasonal_humid, seasonal=FALSE)
tsdisplay(residuals(fit), lag.max=45, main='(1,1,2) Model Residuals')
```



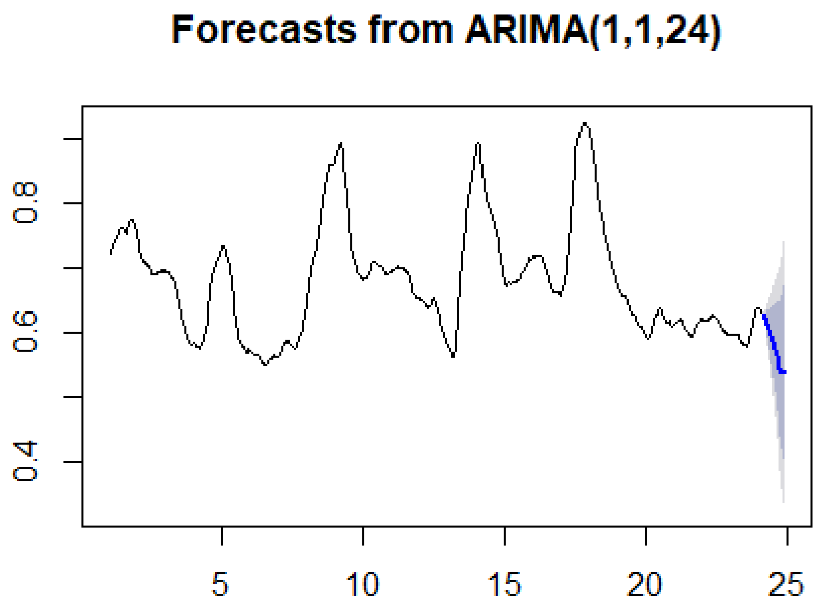



```r
# There is a clear pattern present in ACF/PACF and model residuals plots repe
ating at lag 24. This suggests that our model may be better off with a differ
ent specification, such as p = 24 or q = 24.

fit2 = arima(deseasonal_humid, order=c(1,1,24))
fit2

##
## Call:
## arima(x = deseasonal_humid, order = c(1, 1, 24))
##
## Coefficients:
##           ar1     ma1     ma2     ma3     ma4     ma5     ma6     ma7
##       -0.0291  1.7521  1.7750  1.7855  1.8096  1.7269  1.7984  1.8052
## s.e.   0.0853  0.0793  0.1274  0.1480  0.1592  0.1598  0.1485  0.1530
##          ma8     ma9    ma10    ma11    ma12    ma13    ma14    ma15
##       1.7254  1.6384  1.5578  1.4055  1.4101  1.3843  1.2958  1.2245
## s.e.  0.1517  0.1477  0.1486  0.1622  0.1901  0.2214  0.2512  0.2744
##         ma16    ma17    ma18    ma19    ma20    ma21    ma22    ma23
##       1.1809  1.1131  1.2015  1.2949  1.3277  1.3668  1.4518  1.4670
## s.e.  0.2841  0.2825  0.2725  0.2461  0.2147  0.1776  0.1308  0.0859
##         ma24
##       0.6333
## s.e.  0.0487
##
## sigma^2 estimated as 1.091e-06:  log likelihood = 3749.87,  aic = -7447.75
```

```r
tsdisplay(residuals(fit2), lag.max=15, main='(1,1,24) Seasonal Model Residuals')
```



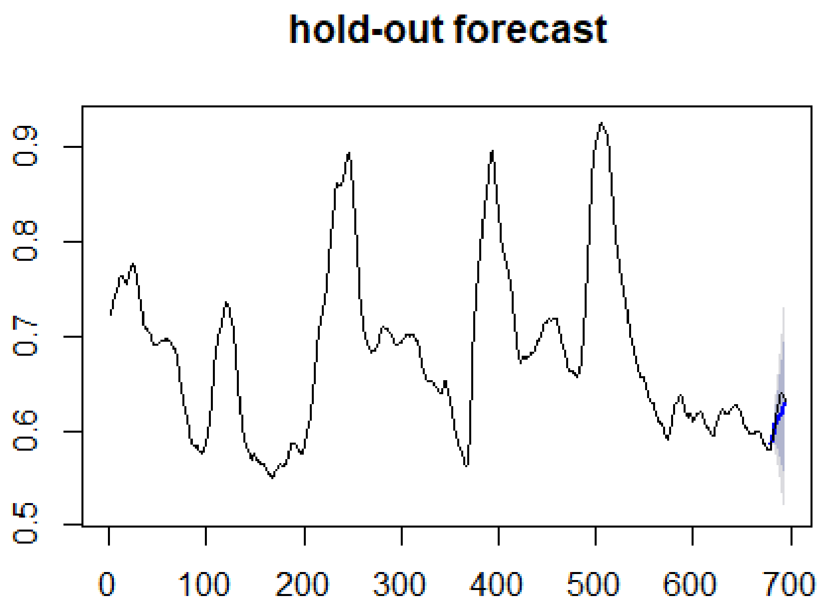

```r
fcast <- forecast(fit2, h=24)
plot(fcast)
```



holt-out forecast

The light blue line above shows the fit provided by the model, but what if we wanted to get a sense of how the model will perform in the future? One method is to reserve a portion of our data as a “hold-out” set, fit the model, and then compare the forecast to the actual observed values:

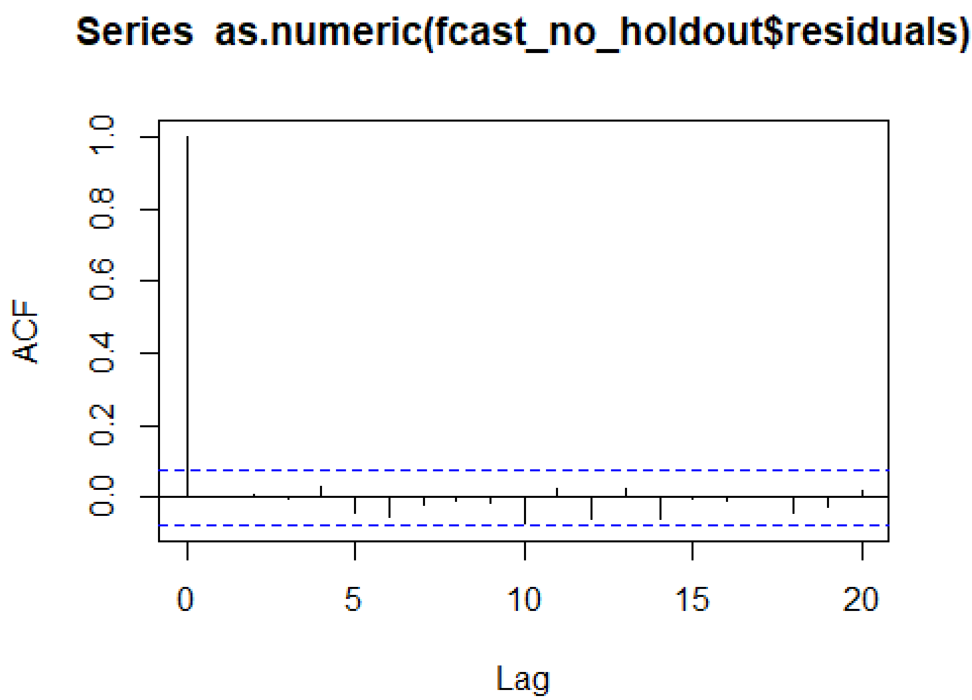
```
hold <- window(ts(deseasonal_humid), start=680)

fit_no_holdout = arima(ts(deseasonal_humid[-c(680:695)]), order=c(1,1,24))

fcast_no_holdout <- forecast(fit_no_holdout,h=15)

plot(fcast_no_holdout, main="hold-out forecast")
lines(ts(deseasonal_humid))
```



Testing the distribution of errors in your ARIMA model.

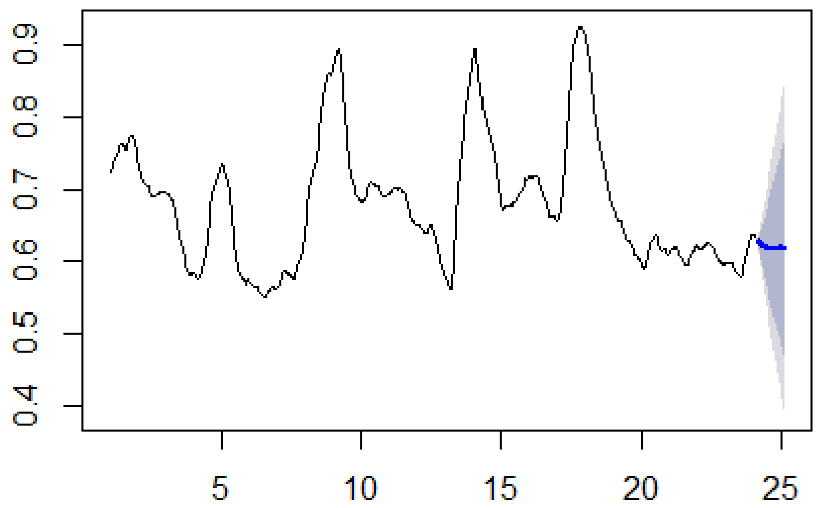
Are successive errors correlated?

```
acf(as.numeric(fcast_no_holdout$residuals), lag.max=20)
```

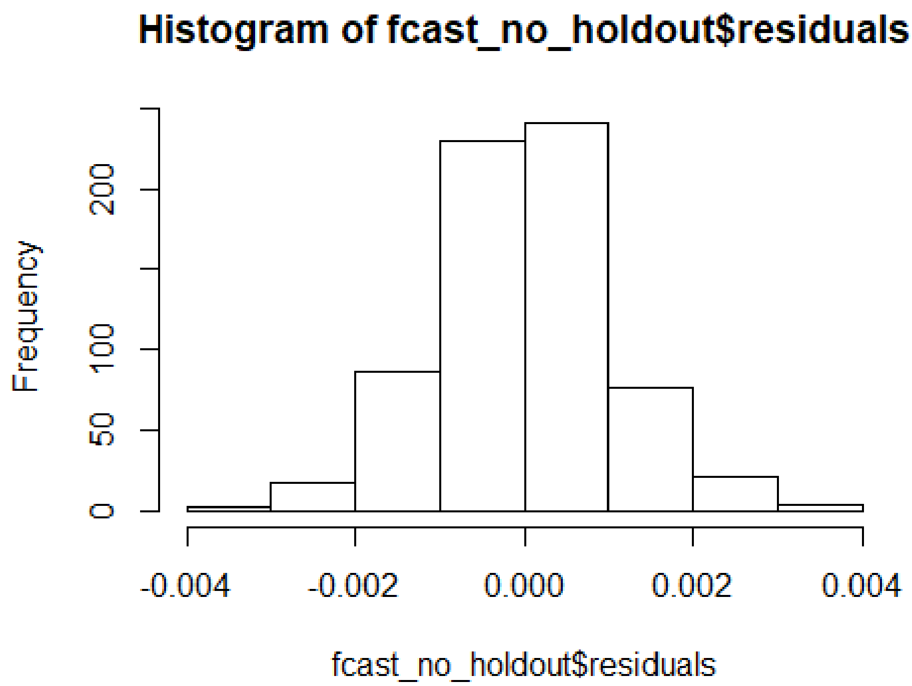



```
fit_w_seasonality = auto.arima(deseasonal_humid, seasonal=TRUE)

fit_w_seasonality

## Series: deseasonal_humid
## ARIMA(1,1,2)(0,0,1)[30]
##
## Coefficients:
##          ar1     ma1     ma2    sma1
##       0.9147  0.8401  0.0945  0.0853
## s.e.  0.0165  0.0421  0.0409  0.0382
##
## sigma^2 estimated as 1.825e-06:  log likelihood=3600.53
## AIC=-7191.05   AICc=-7190.97   BIC=-7168.34

seas_fcast <- forecast(fit_w_seasonality, h=30)
plot(seas_fcast)
```

Forecasts from ARIMA(1,1,2)(0,0,1)[30]

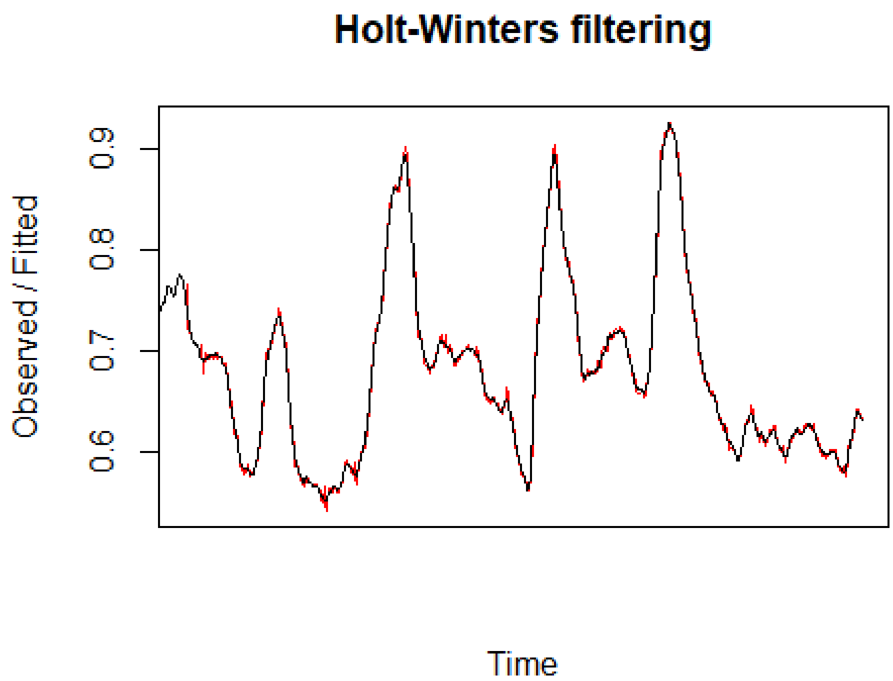



```
# ARIMA
Box.test(fcast_no_holdout$residuals,lag=20,type = "Ljung-Box")

##
##  Box-Ljung test
##
## data:  fcast_no_holdout$residuals
## X-squared = 15.917, df = 20, p-value = 0.7218

hist(fcast_no_holdout$residuals,breaks = 10)
```

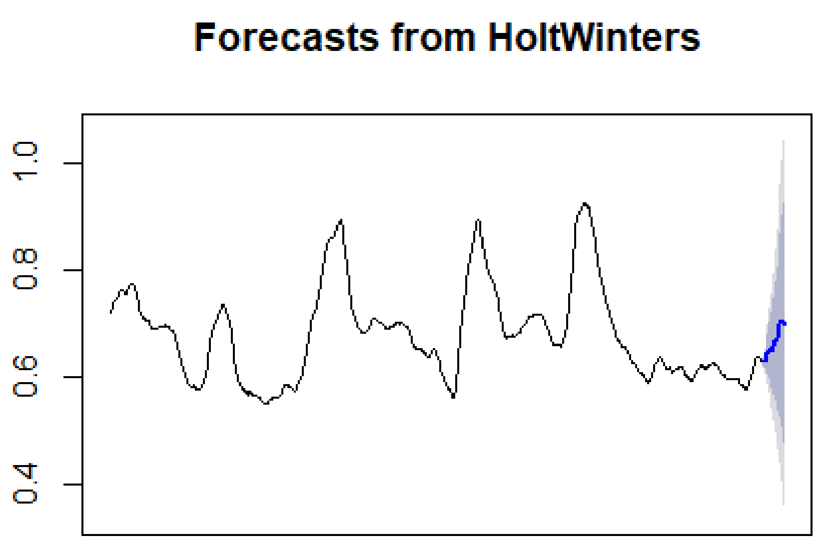


HoltWinters

```
hw = HoltWinters(deseasonal_humid)

plot(hw,xaxt='n')
```



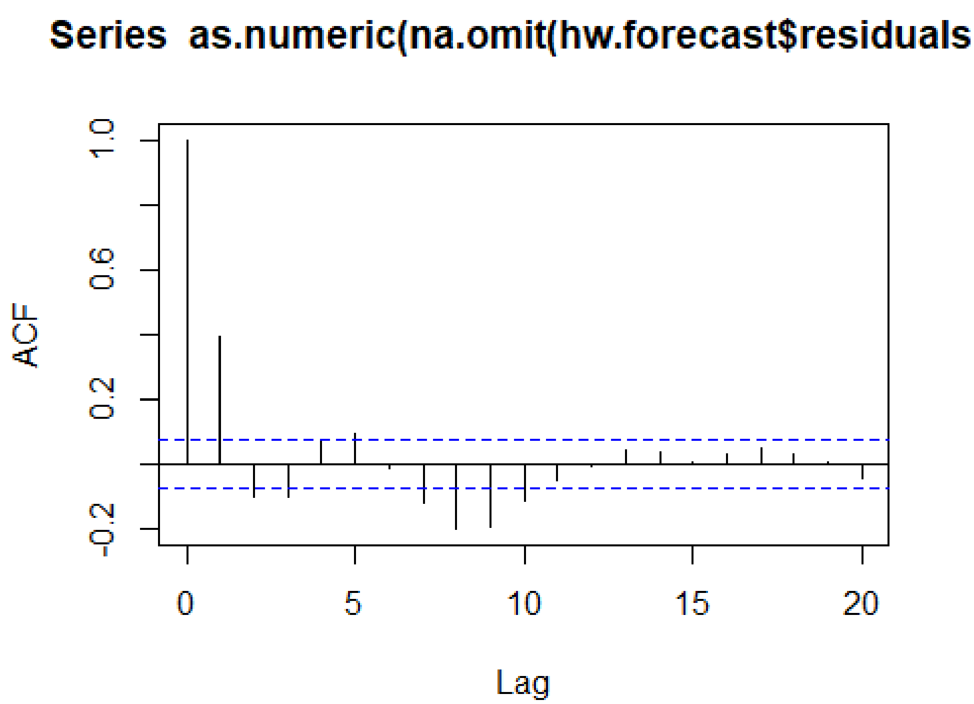

```r
hw.forecast = forecast(hw,h = 24)
plot(hw.forecast,xaxt='n')
```



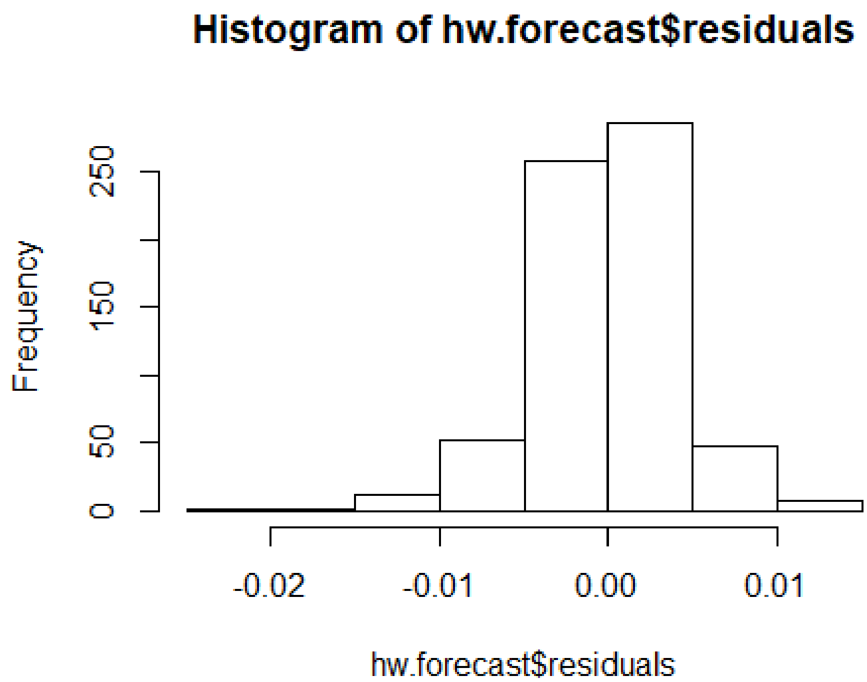
```
hw$SSE

## [1] 0.011527
```

Testing the distribution of errors in your Holt-Winters model.

```
acf(as.numeric(na.omit(hw.forecast$residuals)),lag.max = 20)
```




```
# Holtwinters
Box.test(hw.forecast$residuals,lag=20,type = "Ljung-Box")

##
##  Box-Ljung test
##
## data:  hw.forecast$residuals
## X-squared = 203.18, df = 20, p-value < 2.2e-16

hist(hw.forecast$residuals,breaks = 10)
```


Histogram of hw.forecast$residuals
Frequency
0
50
150
250
-0.02
-0.01
0.00
0.01
hw.forecast$residuals