PSTAT 174 Project Data Tidy

Mu Niu

2023-05-09

Data Preprocessing

(1) We first read in the csv file to get the raw data:

```r
raw <- read.csv('GlobalLandTemperaturesByMajorCity.csv')
head(raw) #Look at the raw data
```

```
##           dt AverageTemperature AverageTemperatureUncertainty    City
## 1 1849-01-01             26.704                         1.435 Abidjan
## 2 1849-02-01             27.434                         1.362 Abidjan
## 3 1849-03-01             28.101                         1.612 Abidjan
## 4 1849-04-01             26.140                         1.387 Abidjan
## 5 1849-05-01             25.427                         1.200 Abidjan
## 6 1849-06-01             24.844                         1.402 Abidjan
##         Country Latitude Longitude
## 1 Côte D’Ivoire    5.63N     3.23W
## 2 Côte D’Ivoire    5.63N     3.23W
## 3 Côte D’Ivoire    5.63N     3.23W
## 4 Côte D’Ivoire    5.63N     3.23W
## 5 Côte D’Ivoire    5.63N     3.23W
## 6 Côte D’Ivoire    5.63N     3.23W
```

(2) My goal is to forecast the temperature of Peking, the capital of China, so I have to filter the data set:

```r
pek <- raw[raw$City == "Peking",]
head(pek)
```

```
##                dt AverageTemperature AverageTemperatureUncertainty   City
## 176248 1820-08-01             22.822                         2.218 Peking
## 176249 1820-09-01             19.384                         1.706 Peking
## 176250 1820-10-01             11.029                         1.817 Peking
## 176251 1820-11-01              2.429                         1.961 Peking
## 176252 1820-12-01             -4.767                         2.237 Peking
## 176253 1821-01-01             -3.565                         2.213 Peking
##        Country Latitude Longitude
## 176248   China   39.38N   116.53E
## 176249   China   39.38N   116.53E
## 176250   China   39.38N   116.53E
## 176251   China   39.38N   116.53E
## 176252   China   39.38N   116.53E
## 176253   China   39.38N   116.53E
```

- Since we want to work on univariate time series, we only want to keep the variable that we want to forecast, which is the Average Temperature:

```
pek.temp <- pek[,1:2]
head(pek.temp)
```

```
##                 dt AverageTemperature
## 176248 1820-08-01             22.822
## 176249 1820-09-01             19.384
## 176250 1820-10-01             11.029
## 176251 1820-11-01              2.429
## 176252 1820-12-01             -4.767
## 176253 1821-01-01             -3.565
```

(3) Now, we want to check whether there is missing values:

```
sum(is.na(pek.temp))
```

```
## [1] 14
```

Knowing there are 14 missing values, we want to know how the missing values are distributed(index of NaN):

```
pek.temp[is.na(pek.temp$AverageTemperature) == TRUE,]
```

```
##                 dt AverageTemperature
## 176394 1832-10-01                 NA
## 176457 1838-01-01                 NA
## 176458 1838-02-01                 NA
## 176459 1838-03-01                 NA
## 176460 1838-04-01                 NA
## 176461 1838-05-01                 NA
## 176462 1838-06-01                 NA
## 176463 1838-07-01                 NA
## 176464 1838-08-01                 NA
## 176465 1838-09-01                 NA
## 176466 1838-10-01                 NA
## 176467 1838-11-01                 NA
## 176468 1838-12-01                 NA
## 178565 2013-09-01                 NA
```

We notice that we don't have observation for October 1832, all year of 1838, and September 2013.

(4) Fill the Missing Values:

- For October 1832: Take the mean value of the neighbors average temperature
- For September 2013: This is the last observation in our data set, so we can't take the mean value of the neighbors average temperature. What we want to do is take the mean value of the two previous months' average temperature, since July, August, and September are in the same quarter of the year.
- For all 12 months of 1838: Even the data set is monthly, I want to make it yearly data by taking the average temperature of 12 months. Hence, we can first fill all 12 months in 1838 by 0. After we get the yearly data, we can change it to the mean value of the 2 neighbor year average temperature.

We first fill the value of October 1832, September 2013, and assign 0 to 1838 all year:

```
pek.temp["176394","AverageTemperature"] <- (pek.temp["176395","AverageTemperature"]+pek.temp["176393","A
pek.temp["178565","AverageTemperature"] <- (pek.temp["178563","AverageTemperature"]+pek.temp["178564","A
for (index in as.character(seq(176457,176468))) {
  pek.temp[index, "AverageTemperature"] <- 0
}

# Check Missing Values
sum(is.na(pek.temp))
```

```
## [1] 0
```

(5) Convert to yearly data:

Looking at the data set, I noticed that both 1820 and 2013 don't have observations for all 12 months. Considering the lack of particular months might lead to mistake, I decided to drop these years.

```
pek.temp <- pek.temp[as.character(seq(176253,178556)),] # Monthly temperature data from 1821 to 2012
```

Then, we convert the monthly data to yearly data by taking average.(yearly data is more smooth than monthly data, and ideal to work with):

```
year.temp <- c()
for (jan in seq(1,2304,12)){
  dec <- jan+11
  year.temp <- c(year.temp,mean(pek.temp[jan:dec,2]))
}
```

Now, we can construct a new data frame by using the yearly data, and fill the value of 1838 by using the mean of its neighbors:

```
Year <- c(1821:2012)
data <- data.frame(Year = Year,
                   AverageTemperature = year.temp)
data[data$Year == 1838, "AverageTemperature"] <- (data[data$Year == 1837, "AverageTemperature"] +
                                                  data[data$Year ==1839,"AverageTemperature"])/2
head(data)
```

```
##   Year AverageTemperature
## 1 1821           11.50625
## 2 1822           11.50225
## 3 1823           10.96133
## 4 1824           11.96475
## 5 1825           11.73483
## 6 1826           11.65225
```

We need split the data into two parts: we use the first part to model the data, and the the second part to test our model

```
model.data <- data[Year <= 2010, ]
test.data <- data[Year > 2010, ]
```

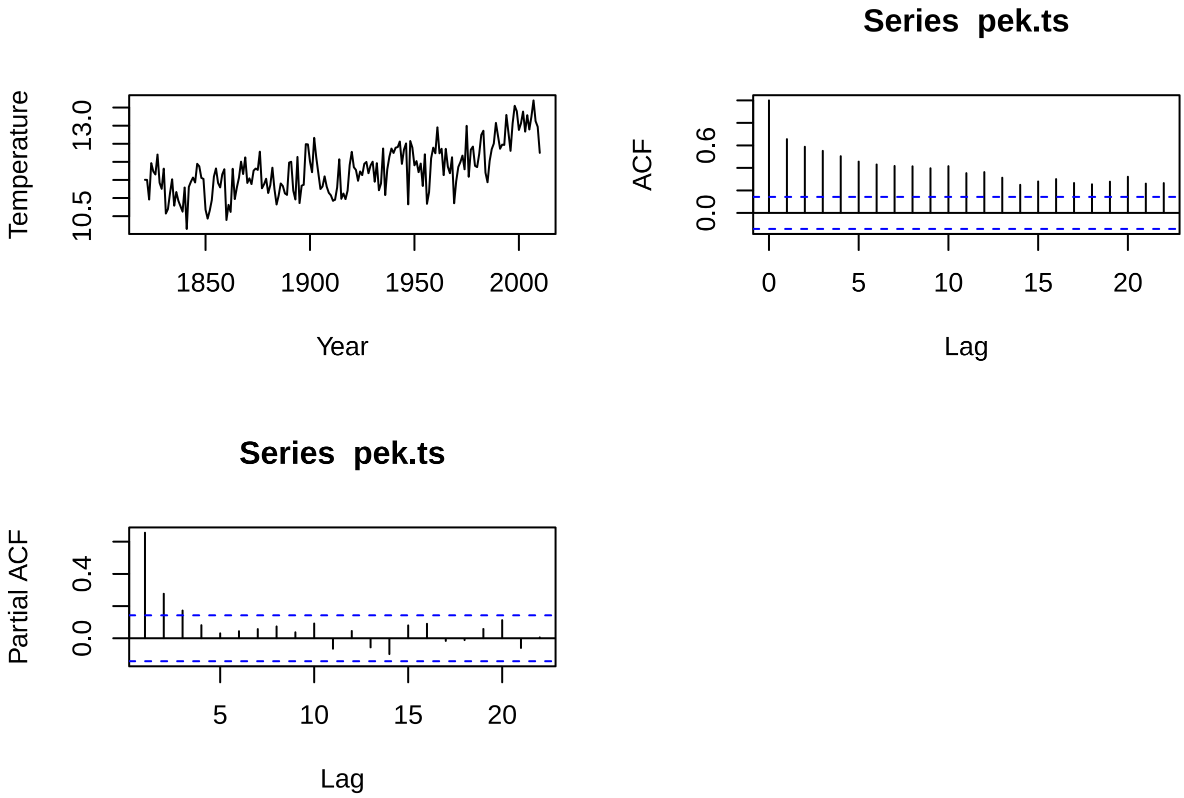
Now, we finished the process of data tidying, and can use this data for time series study.

```
# Export the data set
write.csv(data, "Peking.Temp.1821-2012.csv")
write.csv(model.data, "train.1821-2011.csv")
write.csv(test.data, "test.2012.csv")
```

Transformation and Differencing

Original Peking temperature data set from 1821-2011:

```
op <- par(mfrow=c(2,2))
pek.ts <- ts(model.data[,2], start = 1821, frequency = 1)
ts.plot(pek.ts, gpars=list(xlab="Year", ylab="Temperature"))
acf(pek.ts)
pacf(pek.ts)
par(op)
```

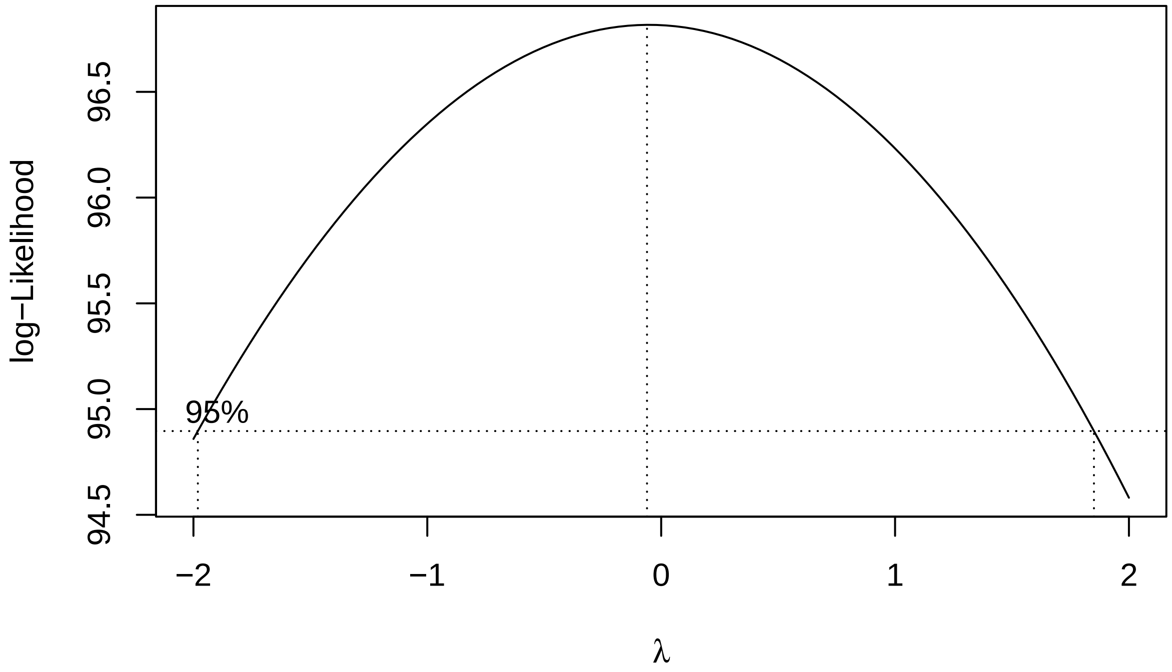


From the plots above, we can see that there is no seasonality but an upward trend. The variance is relatively stable, but we need to check the value of λ and the histograms to see if transformation is necessary.

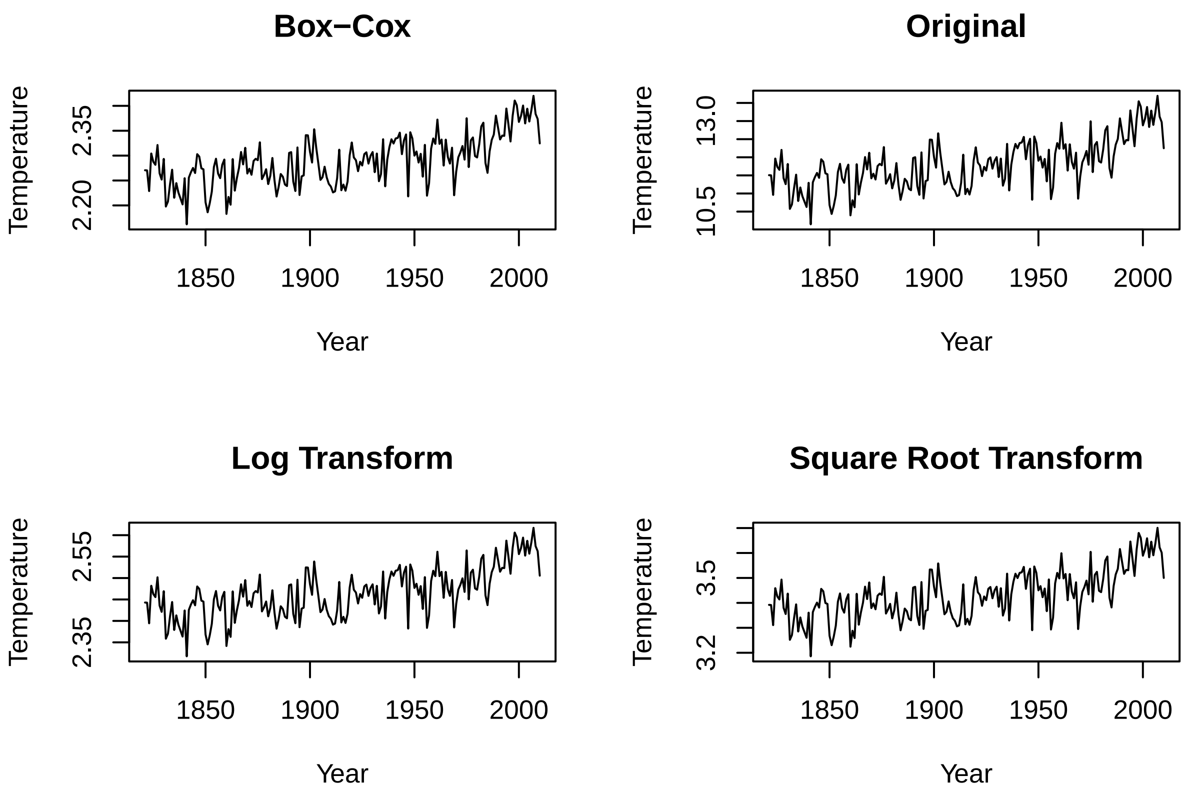
Transformation

Now, let's see if we need any transformations:

```
library(MASS)
t = 1:length(pek.ts)
fit = lm(pek.ts ~ t)
bcTransform = boxcox(pek.ts ~ t,plotit = T)
```

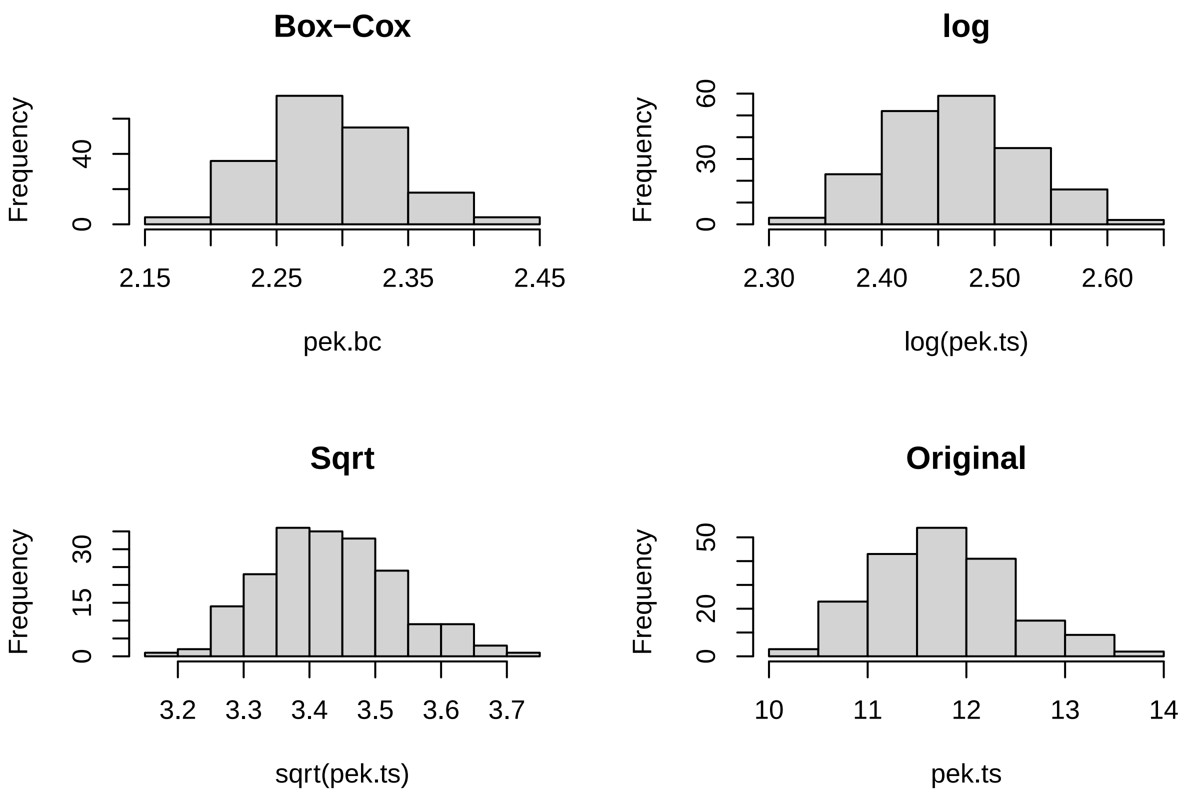


```
lambda = bcTransform$x[which(bcTransform$y == max(bcTransform$y))]
pek.bc = (1/lambda)*(pek.ts^lambda-1)
op <- par(mfrow=c(2,2))
ts.plot(pek.bc,main = "Box-Cox", xlab='Year',ylab='Temperature')
ts.plot(pek.ts,main = "Original", xlab='Year',ylab='Temperature')
ts.plot(log(pek.ts), main = "Log Transform", xlab='Year',ylab='Temperature')
ts.plot(sqrt(pek.ts), main = "Square Root Transform", xlab='Year',ylab='Temperature')
```



```
par(op)
```

```
op <- par(mfrow=c(2,2))
hist(pek.bc, main = 'Box-Cox')
hist(log(pek.ts),main='log')
hist(sqrt(pek.ts),main= 'Sqrt')
hist(pek.ts, main = 'Original')
```



```
par(op)
```

The Box-Cox plot shows that 1 is within the 95% Confidence Interval for the value of λ, also the histogram, which tells us the distribution of the data before transformation, appears to be normally distributed and has a bell shape. Hence, it's safe to conclude that our data doesn't need to be transformed.

Differencing

Since we noticed the upward trend, let's first difference the data once at lag 1 and compare its variance with the variance of data before differencing:

```
var(pek.ts) # Before Diff
```

```
## [1] 0.4770113
```

```
dt.pek.ts <- diff(pek.ts, 1)
(var(dt.pek.ts)) # Diff once at lag 1
```

```
## [1] 0.328638
```

Then, let's difference again at lag 1 and compare their variance:

```
dt2.pek.ts <- diff(dt.pek.ts, 1)
(var(dt2.pek.ts)) # Diff twice at lag 1
```

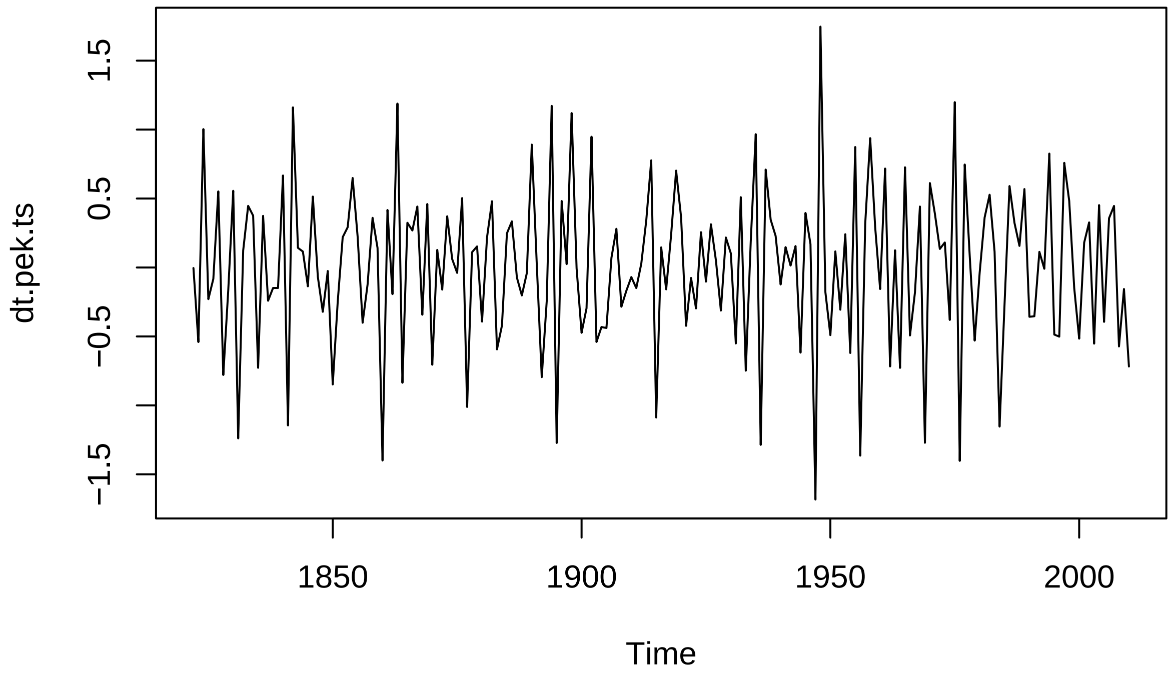
```
## [1] 0.9280505
```

We can see the variance increased after we differentiate twice, which means over differencing. Hence, we want to use the diff once data.

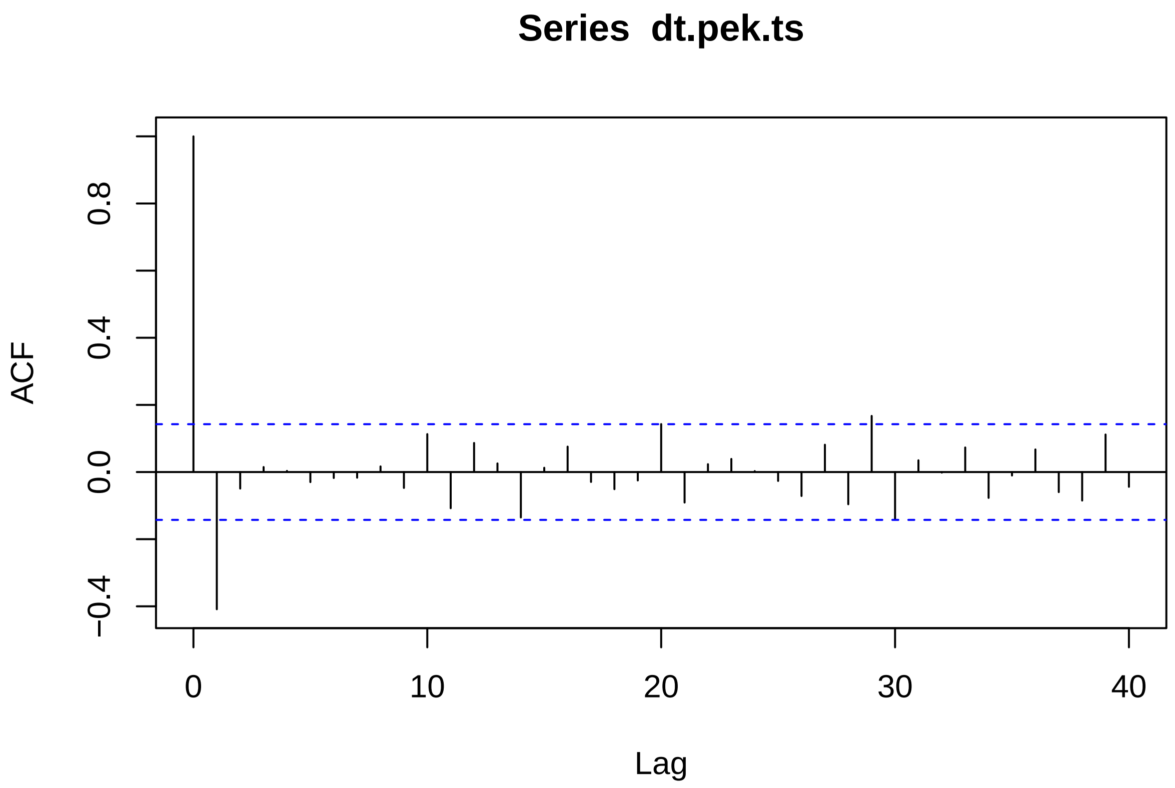
Model Identification

We first want to guess the model based on the ACF and PACF:

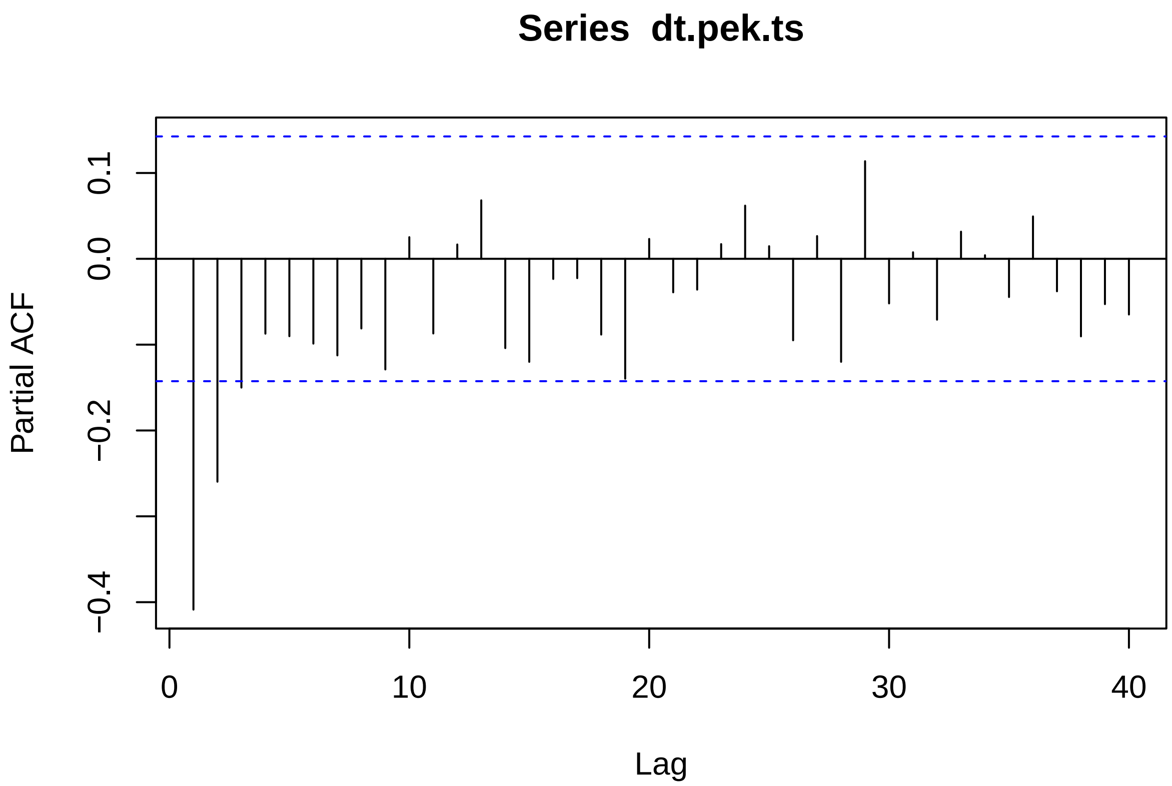
```
plot(dt.pek.ts)
```



```
acf(dt.pek.ts, lag.max = 40)
```



```
pacf(dt.pek.ts, lag.max = 40)
```



Now, we can fit different ARMA models using maximum likelihood estimation and compare the model fits using AICC:

- p: 0,1,2,3
- d: 1
- q: 0,1

```
library(qpcR)
```

```
## Loading required package: minpack.lm

## Loading required package: rgl

## Loading required package: robustbase

## Loading required package: Matrix
```

```
# Construct Matrix
aiccs <- matrix(NA, nr = 4, nc = 2)
dimnames(aiccs) = list(p=0:3, q=0:1)

# Use for loop to calculate the AICc matrix
for(p in 0:3) {
  for(q in 0:1) {
    aiccs[p+1,q+1] = AICc(arima(dt.pek.ts, order = c(p,0,q), method="ML"))
  } }
aiccs
```

```
##    q
## p          0        1
##   0 329.0586 270.5249
##   1 296.4201 265.7672
##   2 284.8366 265.8102
##   3 281.9829 266.3851
```

Another way to identify the model is to use the automatic arima fit function by Rob Hyndman:

```
library(forecast)
```

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

```
auto.arima(pek.ts)
```

```
## Series: pek.ts
## ARIMA(1,1,1) with drift
##
## Coefficients:
##          ar1      ma1   drift
##       0.2847  -0.9210  0.0088
## s.e.  0.0942   0.0512  0.0041
##
## sigma^2 = 0.2309:  log likelihood = -128.82
## AIC=265.64   AICc=265.85   BIC=278.6
```

We can choose the models that has low AICc from the table: $ARIMA(1,1,1)$ and $ARIMA(2,1,1)$

Coefficients Estimation and Diagnostic Checking

- Coefficients Estimation

Now, we can fit the data in the chosen models, which is $ARIMA(1,1,1)$ and $ARIMA(2,1,1)$:

- $ARIMA(1,1,1)$

```
(arima111 <- arima(pek.ts, order=c(1,1,1), method="ML"))
```

```
##
## Call:
## arima(x = pek.ts, order = c(1, 1, 1), method = "ML")
##
## Coefficients:
##          ar1      ma1
##       0.2373  -0.8656
## s.e.  0.0963   0.0547
##
## sigma^2 estimated as 0.2308:  log likelihood = -130.12,  aic = 266.24
```

- $ARIMA(2,1,1)$

```
(arima211 <- arima(pek.ts, order=c(2,1,1), method="ML"))
```

```
##
## Call:
## arima(x = pek.ts, order = c(2, 1, 1), method = "ML")
##
## Coefficients:
##          ar1     ar2      ma1
##       0.2519  0.0870  -0.8949
## s.e.  0.0890  0.0839   0.0479
##
## sigma^2 estimated as 0.2295:  log likelihood = -129.59,  aic = 267.18
```

Since the 95% Confidence Interval of ϕ_2 contains 0, we can set it to zero. But doing so will give us $ARIMA(1,1,1)$ model, which is exactly the same with the previous one. So we remove $ARIMA(2,1,1)$ and consider $MA(1)$ with AICc = 270.5249 and $AR(3)$ with AICc = 281.9829, because they have low AICc in comparison with other models:

```
(ma1 <- arima(pek.ts, order=c(0,1,1), method="ML"))
```

```
##
## Call:
## arima(x = pek.ts, order = c(0, 1, 1), method = "ML")
##
## Coefficients:
##           ma1
##       -0.7186
## s.e.   0.0742
##
## sigma^2 estimated as 0.2371:  log likelihood = -132.53,  aic = 269.07
```

```
(ar3 <- arima(pek.ts, order=c(3,1,0), method="ML"))
```

```
##
## Call:
## arima(x = pek.ts, order = c(3, 1, 0), method = "ML")
##
## Coefficients:
```

```
##           ar1      ar2      ar3
##       -0.5644  -0.3494  -0.1620
## s.e.   0.0724   0.0793   0.0728
##
## sigma^2 estimated as 0.2463:  log likelihood = -135.96,  aic = 279.91
```

Since $MA(1)$ has fewer parameters and lower AICc value(270.5249 $<$ 281.9829), it's better to only take the $MA(1)$ model and $ARIMA(1,1,1)$ to diagnostic checking.

- Diagnostic Checking

Let's first write out the two models:

Let X_t denotes our data,

(A) $ARIMA(1,1,1)$:

$(1-0.2373_{0.0963}B)\nabla_1 X_t = (1-0.8656_{0.0547}B)Z_t$ with $\hat{\sigma}_Z^2 = 0.2308$

(B) $MA(1)$:

$\nabla_1 X_t = (1-0.7186_{0.0742}B)Z_t$ with $\hat{\sigma}_Z^2 = 0.2371$

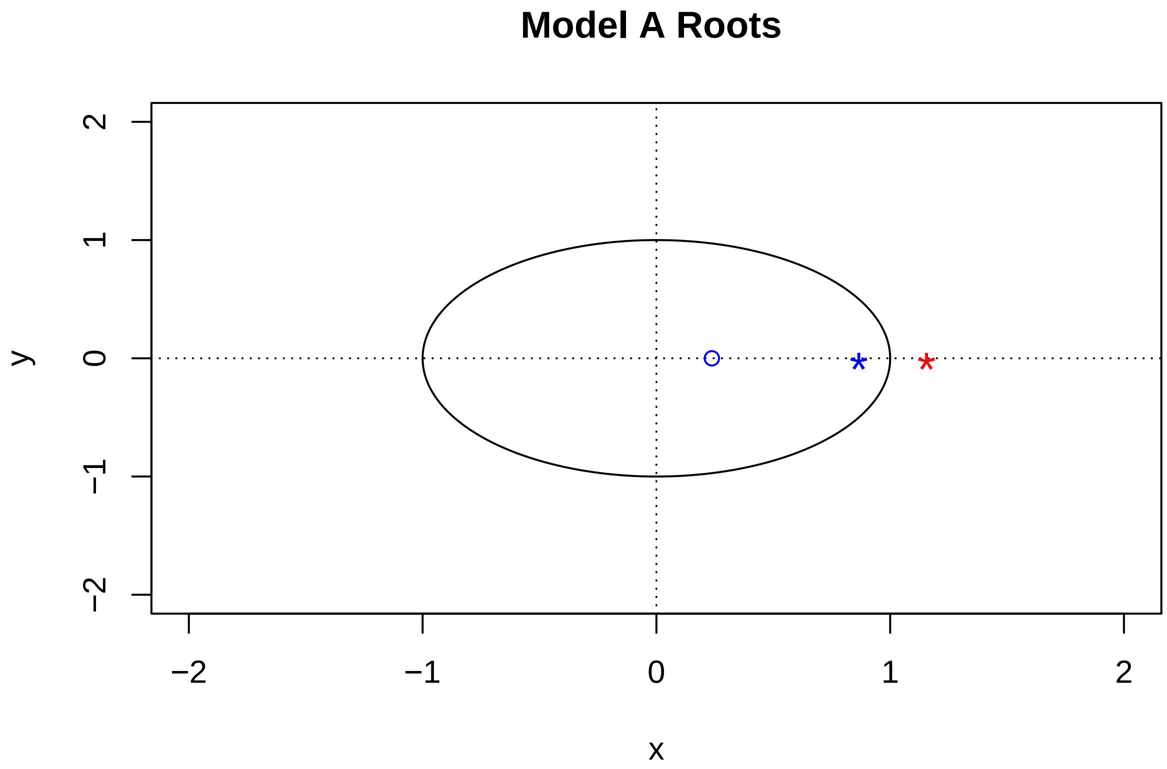
(1) Invertibility and Stationarity

For model (A):

$(1-0.2373_{0.0963}B)\nabla_1 X_t = (1-0.8656_{0.0547}B)Z_t$

Since $|\theta_1| < 1$ and $|\phi_1| < 1$, we can conclude that this model is both stationary and invertible. We can also plot the roots:

```
source('plot.roots.R')
plot.roots(polyroot(c(1, -0.2373)),polyroot(c(1, -0.8656)), main="Model A Roots ")
```



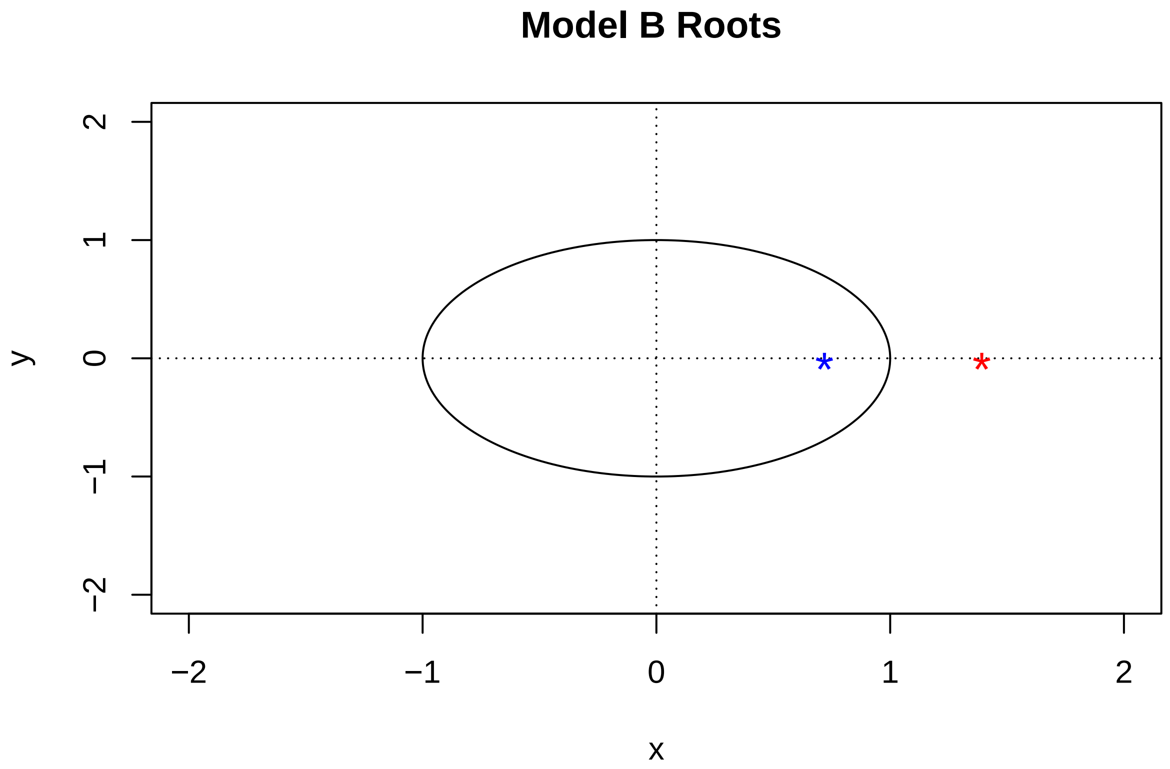
From, the plot above, we can see that both roots are within the unit circle, which confirms our conclusion.

For model (B):

$\nabla_1 X_t = (1 - 0.7186_{0.0742} B) Z_t$

Since this is a pure MA model, it's automatically stationary. Also, $|\theta_1| < 1$, so we can conclude that this model is both stationary and invertible. We can also plot the roots:

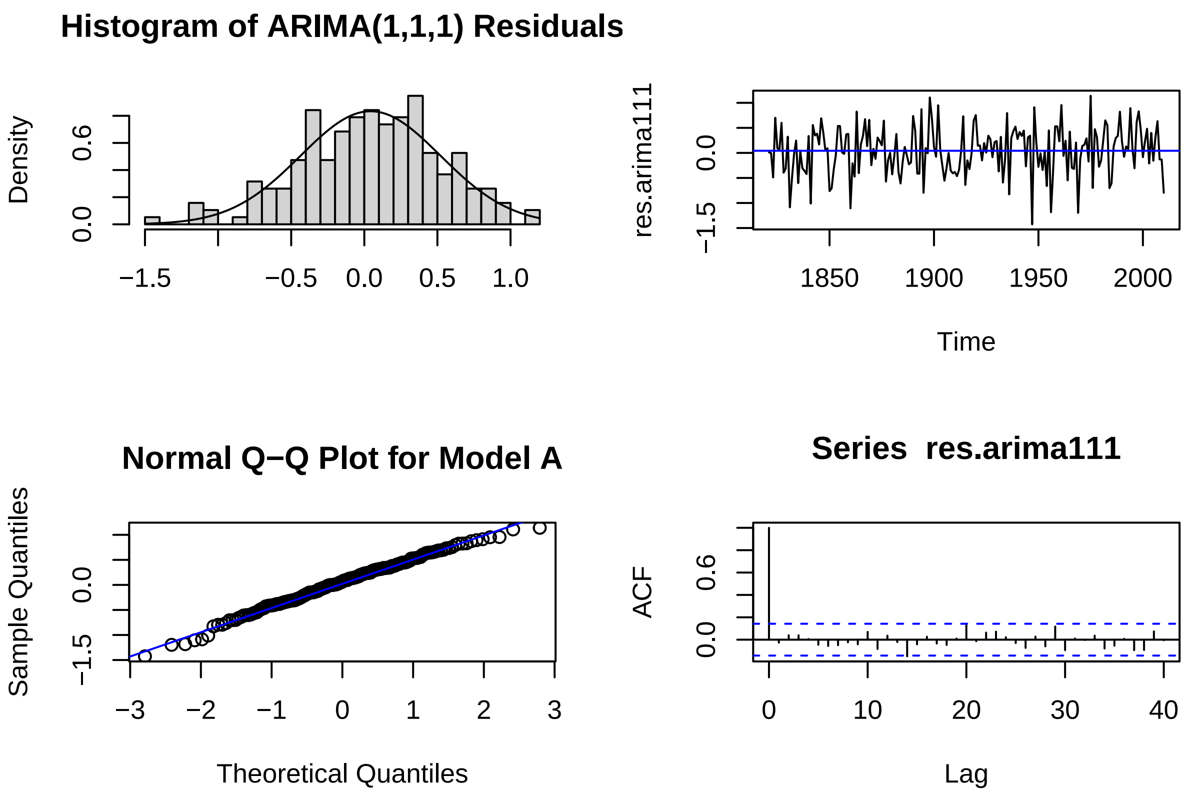
```
source('plot.roots.R')
plot.roots(NULL, polyroot(c(1, -0.7186)), main="Model B Roots ")
```



From, the plot above, we can see that the root is within the unit circle, which verifies our conclusion.

(2) Diagnostic Checking on Model (A): $ARIMA(1,1,1)$

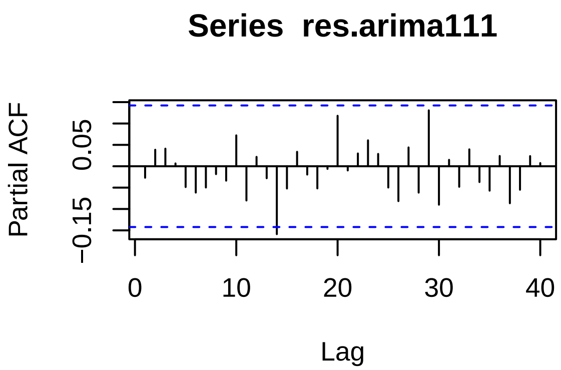
```
op <- par(mfrow = c(2,2))
res.arima111 <- residuals(arima111)
hist(res.arima111, breaks = 20, xlab="", prob=TRUE, main = 'Histogram of ARIMA(1,1,1) Residuals')
m.arima111 <- mean(res.arima111)
std.arima111 <- sqrt(var(res.arima111))
curve(dnorm(x,m.arima111,std.arima111), add=TRUE )
plot.ts(res.arima111)
abline(h=mean(res.arima111), col="blue")
qqnorm(res.arima111,main= "Normal Q-Q Plot for Model A")
qqline(res.arima111,col="blue")
acf(res.arima111, lag.max=40)
```


Histogram of ARIMA(1,1,1) Residuals

Normal Q−Q Plot for Model A

Series res.arima111

```
pacf(res.arima111, lag.max=40)
par(op)
```



From the plots above, we can say that there is no trend, no visible change of variance, no seasonality, sample mean is almost zero, histogram and Q-Q plot all look good.

Now, we can apply tests to residuals:

```
shapiro.test(res.arima111)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.arima111
## W = 0.99219, p-value = 0.4014
```

```
Box.test(res.arima111, type=c("Box-Pierce"), lag = 14, fitdf = 2)
```

```
##
##  Box-Pierce test
##
## data:  res.arima111
## X-squared = 9.3912, df = 12, p-value = 0.6692
```

```
Box.test(res.arima111, type=c("Ljung-Box"), lag = 14, fitdf = 2)
```

```
##
##  Box-Ljung test
##
## data:  res.arima111
## X-squared = 10.063, df = 12, p-value = 0.6104
```

```
Box.test((res.arima111)^2, type=c("Ljung-Box"), lag = 14, fitdf = 0)
```

```
##
##  Box-Ljung test
##
## data:  (res.arima111)^2
## X-squared = 15.916, df = 14, p-value = 0.3185
```

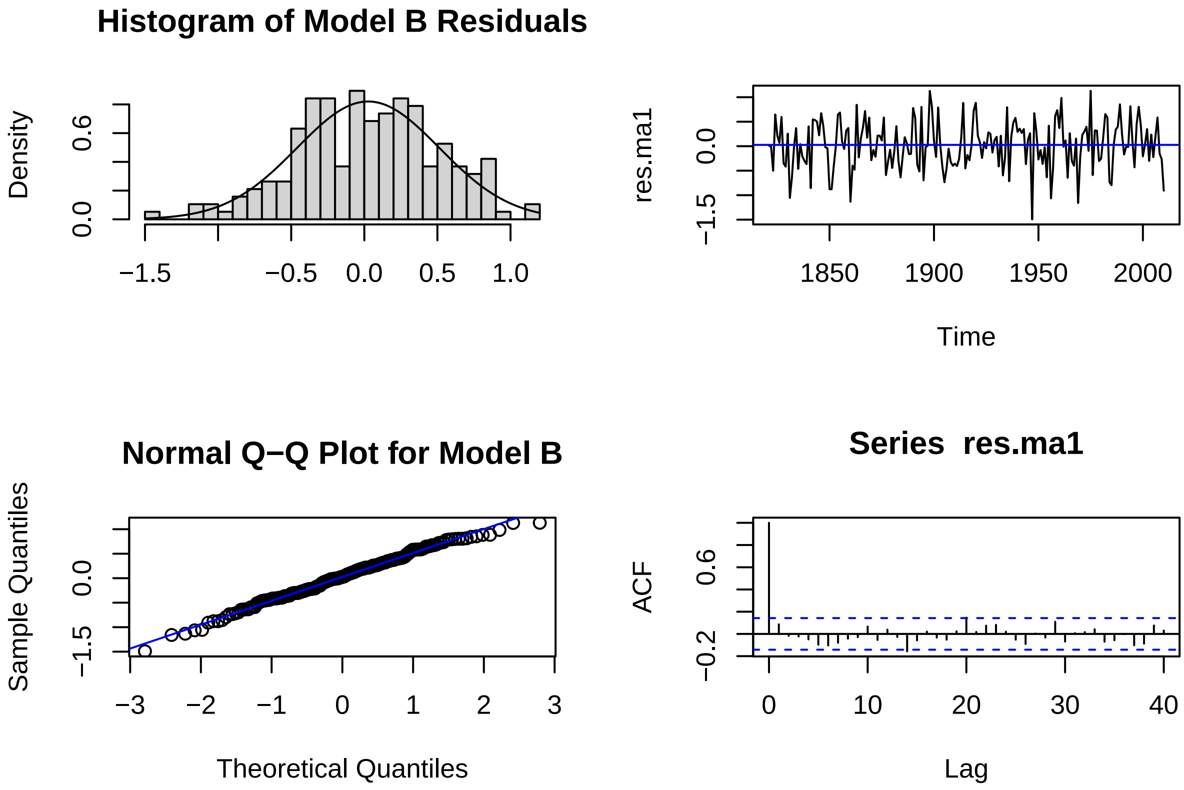
```
ar(res.arima111, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

```
##
## Call:
## ar(x = res.arima111, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.229
```

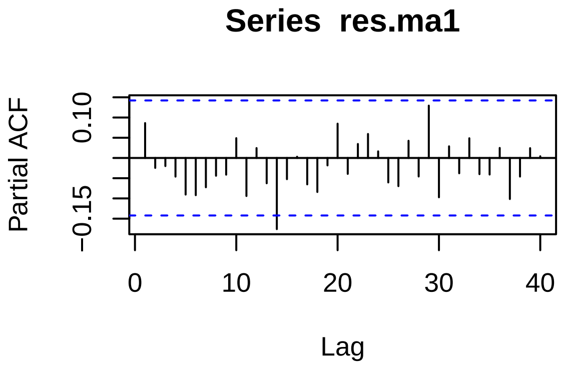
Since the residuals passed Shapiro test, Box-Pierce test, Ljung-Box test, Mcleod-Li test and AR order selects 0, it's safe to conclude that model A passed all the tests.

(3) Diagnostic Checking on Model (B): $MA(1)$

```
op <- par(mfrow = c(2,2))
res.ma1 <- residuals(ma1)
hist(res.ma1, breaks = 20, xlab="", prob=TRUE, main = 'Histogram of Model B Residuals')
m.ma1 <- mean(res.ma1)
std.ma1 <- sqrt(var(res.ma1))
curve(dnorm(x,m.ma1,std.ma1), add=TRUE )
plot.ts(res.ma1)
abline(h=mean(res.ma1), col="blue")
qqnorm(res.ma1,main= "Normal Q-Q Plot for Model B")
qqline(res.ma1,col="blue")
acf(res.ma1, lag.max=40)
```



```
pacf(res.ma1, lag.max=40)
par(op)
```



From the plots above, we can say that there is no trend, no visible change of variance, no seasonality, sample mean is almost zero, histogram and Q-Q plot all look good.

Now, we can apply tests to residuals:

```
shapiro.test(res.ma1)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.ma1
## W = 0.99291, p-value = 0.4891
```

```
Box.test(res.ma1, type=c("Box-Pierce"), lag = 14, fitdf = 1)
```

```
##
##  Box-Pierce test
##
## data:  res.ma1
## X-squared = 14.06, df = 13, p-value = 0.3696
```

```r
Box.test(res.ma1, type=c("Ljung-Box"), lag = 14, fitdf = 1)
```

```
##
##  Box-Ljung test
##
## data:  res.ma1
## X-squared = 14.9, df = 13, p-value = 0.3136
```

```r
Box.test((res.ma1)^2, type=c("Ljung-Box"), lag = 14, fitdf = 0)
```

```
##
##  Box-Ljung test
##
## data:  (res.ma1)^2
## X-squared = 14.397, df = 14, p-value = 0.4206
```

```r
ar(res.ma1, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

```
##
## Call:
## ar(x = res.ma1, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.2364
```

Since the residuals passed Shapiro test, Box-Pierce test, Ljung-Box test, Mcleod-Li test and AR order selects 0, it's safe to conclude that model B passed all the tests.

(4) Final Model Selection:

As we can see, both models passed all the test. According to the principle of parsimony, I should choose the one with the least coefficients, which is the $MA(1)$ model. Hence, the final model for de-trended data: X_t follows $MA(1)$ model: $\nabla_1 X_t = (1 - 0.7186_{0.0742} B) Z_t$ with $\hat{\sigma}_Z^2 = 0.2371$.

Forecasting

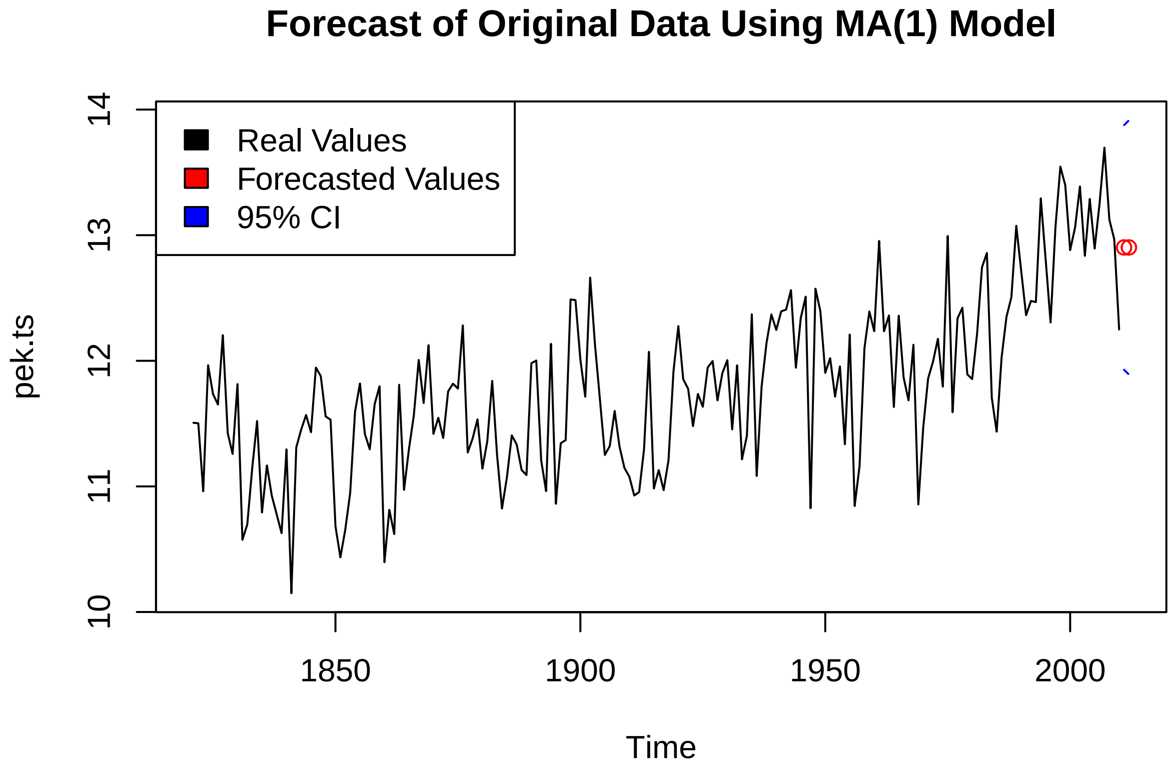
Now, we can forecast by using our model B:

```r
library(forecast)
# Fit into model and forecast
fit <- arima(pek.ts, order=c(0,1,1), method="ML")
forecast(fit)
```

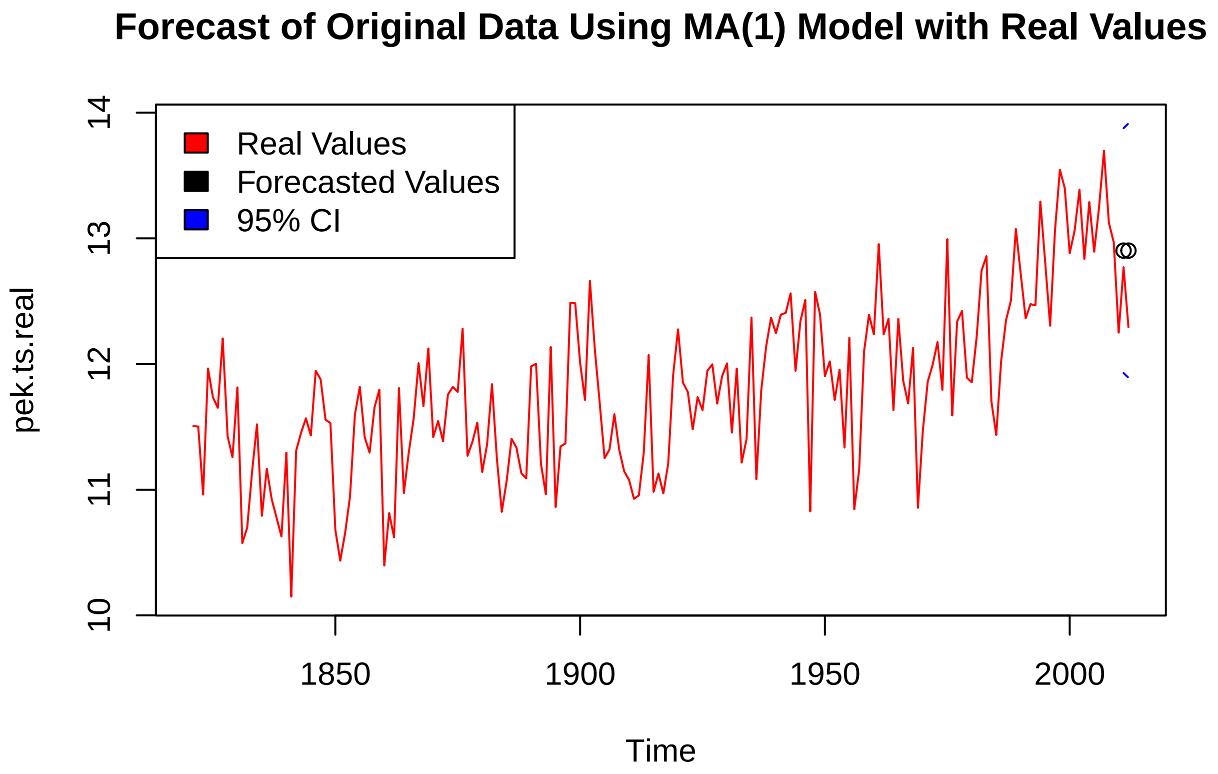
```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2011       12.90245 12.27842 13.52649 11.94807 13.85683
## 2012       12.90245 12.25417 13.55073 11.91099 13.89391
## 2013       12.90245 12.23080 13.57410 11.87525 13.92965
## 2014       12.90245 12.20822 13.59668 11.84072 13.96419
## 2015       12.90245 12.18635 13.61855 11.80727 13.99764
## 2016       12.90245 12.16513 13.63978 11.77481 14.03009
## 2017       12.90245 12.14450 13.66041 11.74326 14.06164
## 2018       12.90245 12.12441 13.68049 11.71255 14.09236
## 2019       12.90245 12.10484 13.70006 11.68261 14.12230
## 2020       12.90245 12.08573 13.71917 11.65339 14.15152
```

```
# To produce graph with 1 forecast on data:
pred<- predict(fit, n.ahead = 2)
U= pred$pred + 2*pred$se # Upper bound of prediction interval
L= pred$pred - 2*pred$se # Lower bound of prediction interval

# Plot the forecast value for 2011 and 2012 with 95% CI on original data
ts.plot(pek.ts, xlim=c(1821,2012), ylim = c(min(pek.ts),max(U)), main = 'Forecast of Original Data Usin
lines(2011:2012, y =  U, col="blue", lty=2)
lines(2011:2012, y =  L, col="blue", lty=2)
points(2011:2012, pred$pred, col="red")
legend("topleft",
       legend = c('Real Values', 'Forecasted Values', '95% CI'),
       fill = c('black','red','blue'),
       border = "black")
```

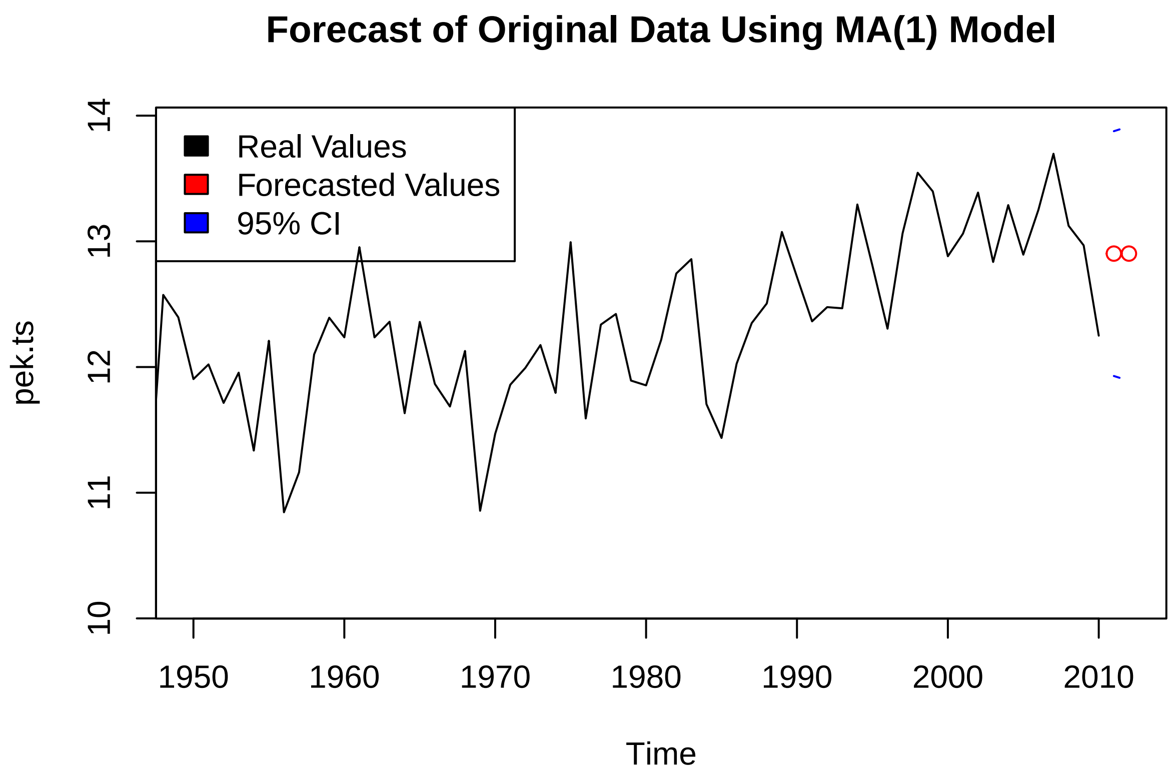


```
# Plot the forecast value for 2011 and 2012 with 95% CI on original data with real values
pek.ts.real <- ts(data[,2], start = 1821, frequency = 1)
ts.plot(pek.ts.real, xlim = c(1821,2012), ylim = c(min(pek.ts),max(U)), col="red", main = 'Forecast of 
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points(2011:2012, pred$pred, col="black")
legend("topleft",
       legend = c('Real Values', 'Forecasted Values', '95% CI'),
       fill = c('red','black','blue'),
       border = "black")
```

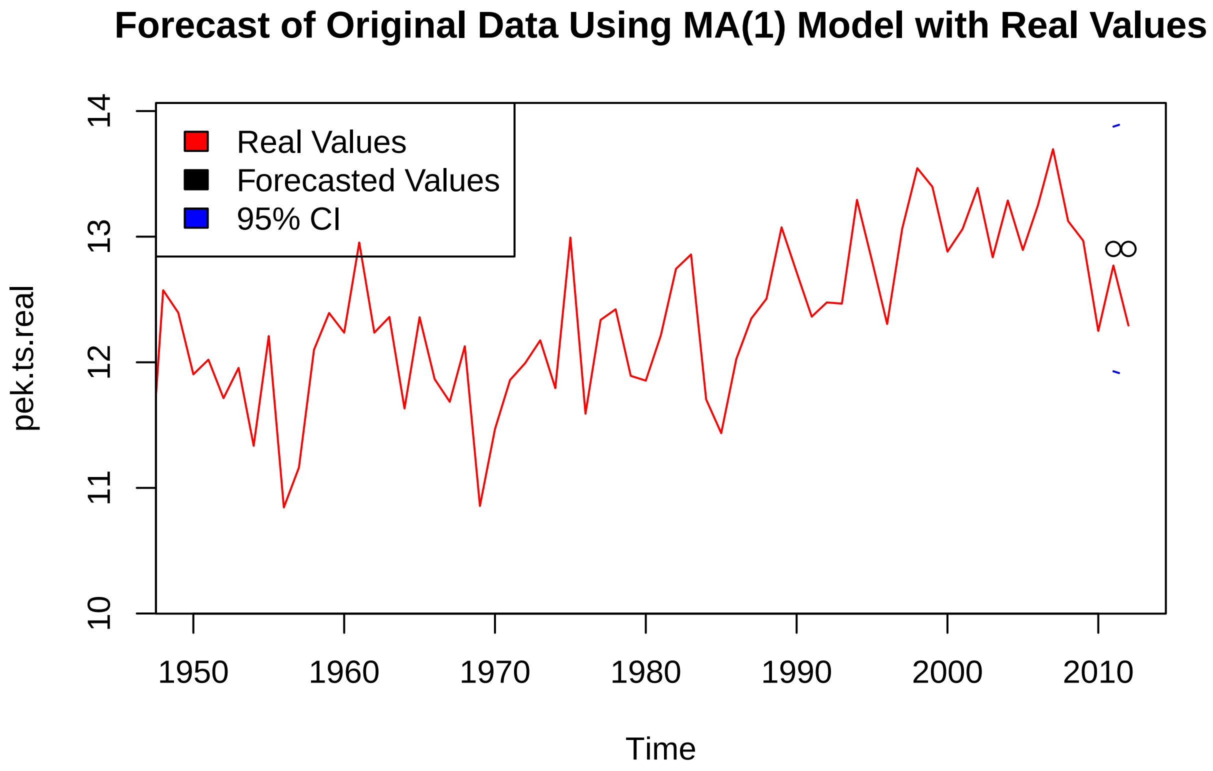


Look at the plots above, we can see that the real values from our test set is within the prediction interval. And we can plot the zoom-in graphs to see it more clearly:

```
# Plot the forecast value for 2011 and 2012 with 95% CI on original data: Zoom In
ts.plot(pek.ts, xlim=c(1950,2012), ylim = c(min(pek.ts),max(U)), main = 'Forecast of Original Data Usin
lines(2011:2012, y =  U, col="blue", lty=2)
lines(2011:2012, y =  L, col="blue", lty=2)
points(2011:2012, pred$pred, col="red")
legend("topleft",
       legend = c('Real Values', 'Forecasted Values', '95% CI'),
       fill = c('black','red','blue'),
       border = "black")
```



```
# Plot the forecast value for 2011 and 2012 with 95% CI on original data with real values: Zoom In
ts.plot(pek.ts.real, xlim = c(1950,2012), ylim = c(min(pek.ts),max(U)), col="red", main = 'Forecast of
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points(2011:2012, pred$pred, col="black")
legend("topleft",
       legend = c('Real Values', 'Forecasted Values', '95% CI'),
       fill = c('red','black','blue'),
       border = "black")
```


Forecast of Original Data Using MA(1) Model with Real Values
Real Values
Forecasted Values
95% CI
pek.ts.real
10
11
12
13
14
1950
1960
1970
1980
1990
2000
2010
Time