StackOverFlow Python Questions Time Series Analysis

Kathy Wu

March 11, 2022

Contents

1. Abstract

In this project, I will use the time series method I learned in PSTAT174 to analyze the StackOverFlow Question Counts from Kaggle. I use the data from 2009 to 2018 as the training data to fit a SARIMA Model, and use the data of 2019 as the test data to check the model fit. First, I visualized the data and determine if any transformation is needed to obtain a equal variance. Then, I plot the sample ACF and PACF to choose the candidate models. By selecting the best model that has the lowest AICc, I checked the ACF and PACF, normality, independence, non-linear dependence, and characteristic roots of the residuals to ensure the it follows the white noise process. Finally, I forecasted the StackOverFlow Python Question Counts in 2019 and compare it with the original data using the 95% Prediction Interval.
However, when checking the McLeod-Li Test for testing if the residuals has non-linear dependence, the result shows that there exists non-linear dependence, and there might be a non-linear model that fit the data better. As a result, I fit the most appropriate linear model that would provide the best forecast of the Python data.

2. Introduction

The StackOverFlow data was collected from Kaggle, which includes the number of question counts of various coding languages, algorithm, and specific libraries. I find this data set interesting is because nowadays programming languages become more and more popular among students and industry companies. The specific type of question I would like to focus is python. From analyzing the forecasted number of questions on python, while understanding the behavior of future number of questions related to python will be asked on StackOverFlow, we can not only know that Python is becoming more and more popular, but also recruit or call upon enough people to help others understand the python questions and may be able to publish more teaching materials about python. So, my goal is to use the SARIMA to fit a linear model and predict the future trend and seasonality of the Questions Counts.
Although in the end I fitted a linear model, the residuals still show strong non-linear dependence. I still try my best to fit the most appropriate linear model and forecast the future value, and the true test data appears to be within the prediction interval except one Nov.2019 observation. As a result, I think this model might have a better fit with the non-linear model.

3. Analyzing the Data Set

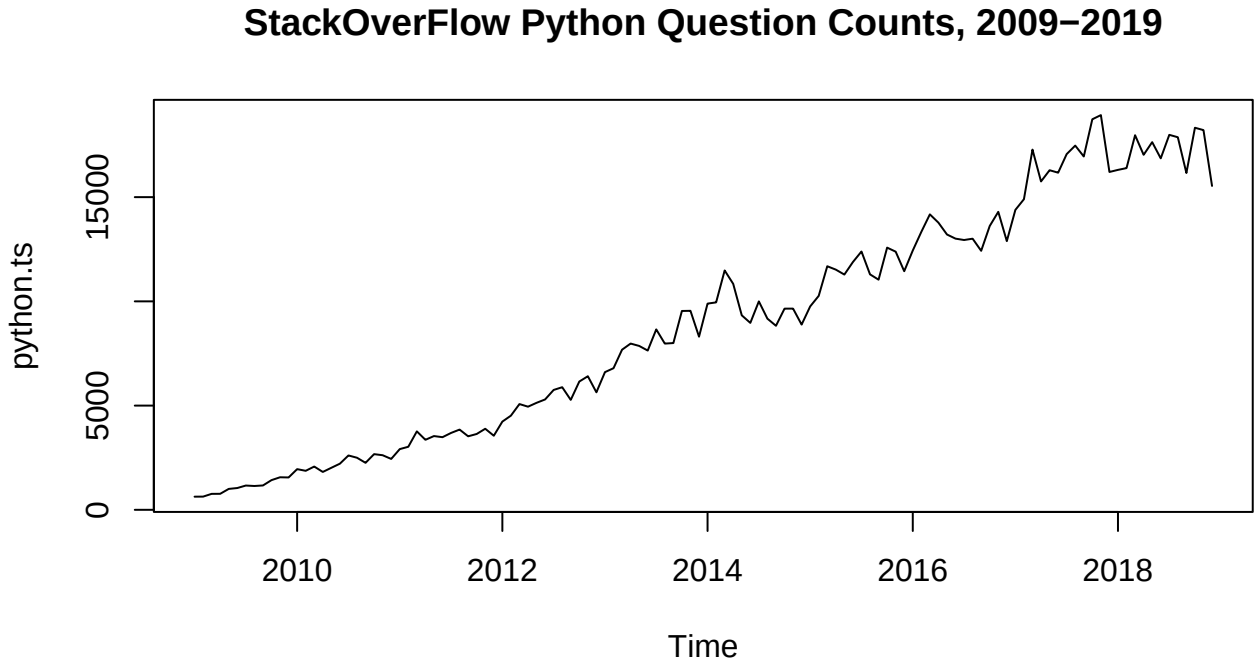
3.1 Preview of the Python Data

The data ranges from Jan.2009 to Dec.2019. I then split the data into training and test sets, which are Jan.2009 to Dec.2018 as the training set, and Jan.2019 to Dec.2019 as the test set. The partial data is shown below:

```
##    Jan Feb Mar Apr  May  Jun  Jul  Aug Sept  Oct  Nov  Dec
## py 631 633 766 768 1003 1046 1165 1143 1169 1424 1562 1552
```

3.2 Plotting the Python Data

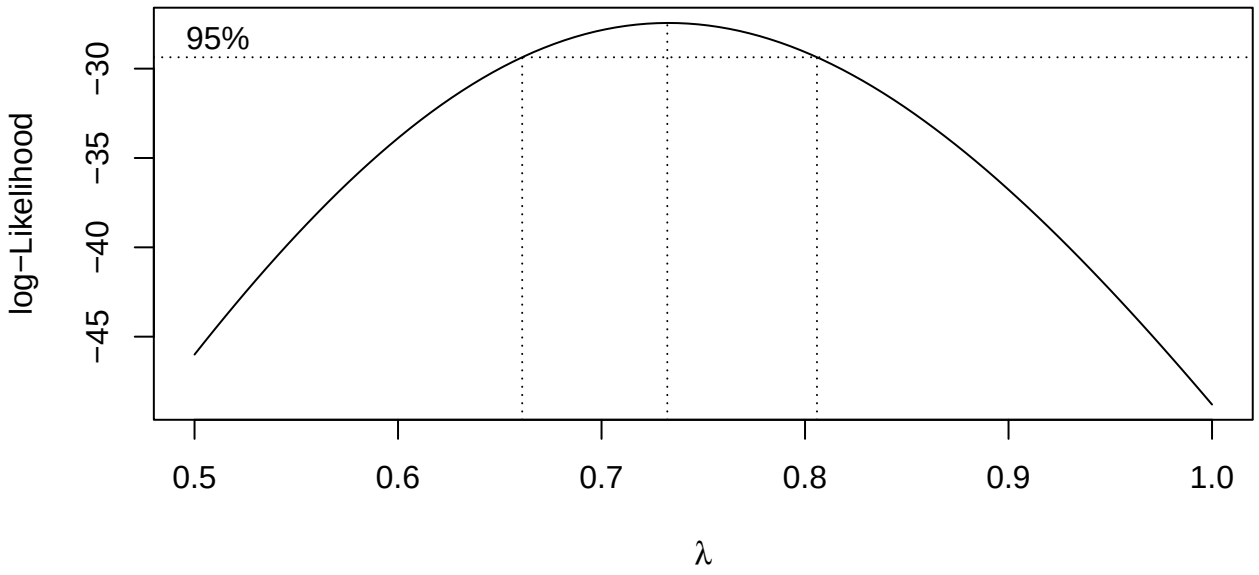
The time series plot of the Python Data is:



From the time series plotting above, we can see that the time series involves an upward trend and seasonality, and between the year 2018 and 2019, we can observe that there is relatively a small sharp increase in the time series. Furthermore, this time series may has unequal variance. So I choose to consider to apply the Box-Cox Transformation to check whether we need transformation to conduct a equal variance.

3.3 Possible Transformation of the Python Data

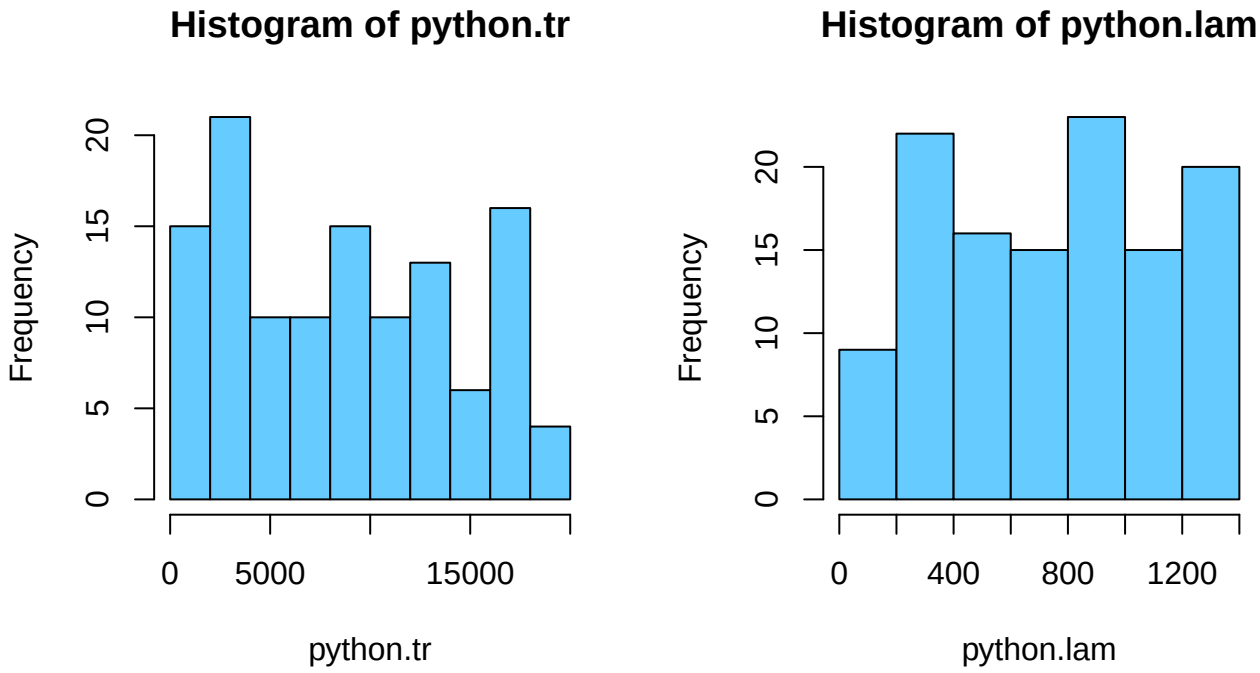
The Box-Cox Transformation is shown below:



Since the 95% Confidence Interval of Box-Cox Transformation does not include 0 or 1, which means we would not choose log transformation or keep the original data set. Hence, I choose to take the power of λ to

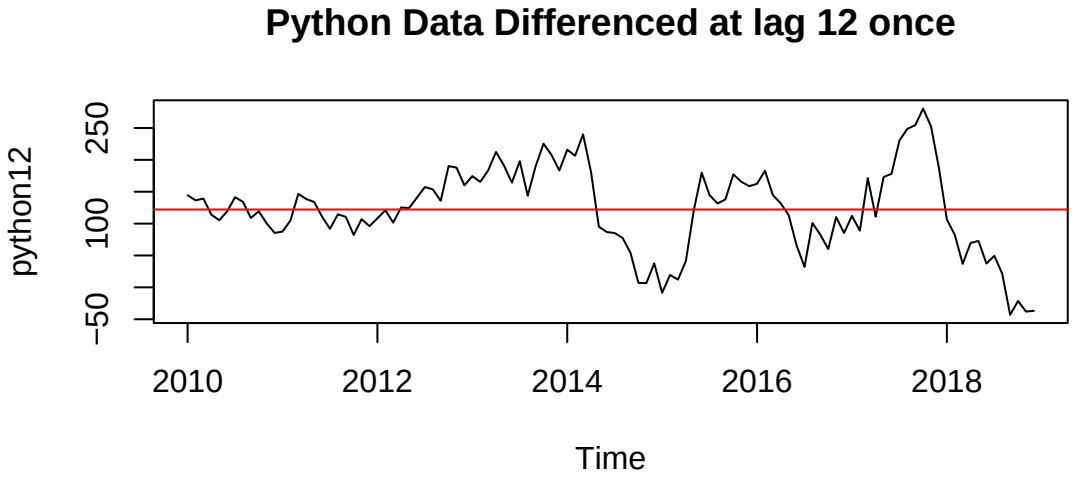
the data set as the transformation.

The histogram on the right-hand-side indicates that the variance is getting more consistent after transformation.

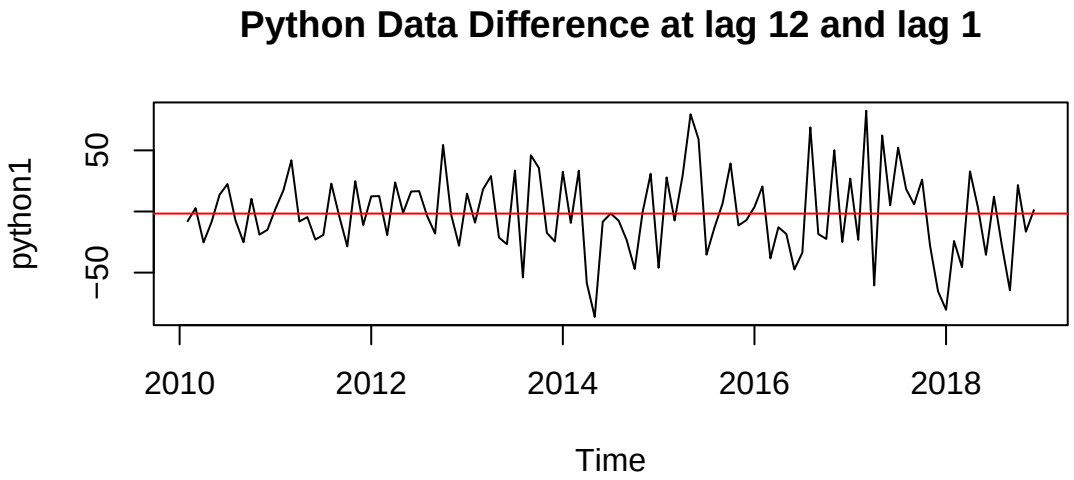


3.4 Difference the Python Data

Since the previous time series plot shows seasonality, so I first choose to difference once at lag 12 to reduce the seasonality of the Python Data.



However, the plot still appears non-stationary. Then, I choose to difference the Python Data at lag 1 once to remove the trend.



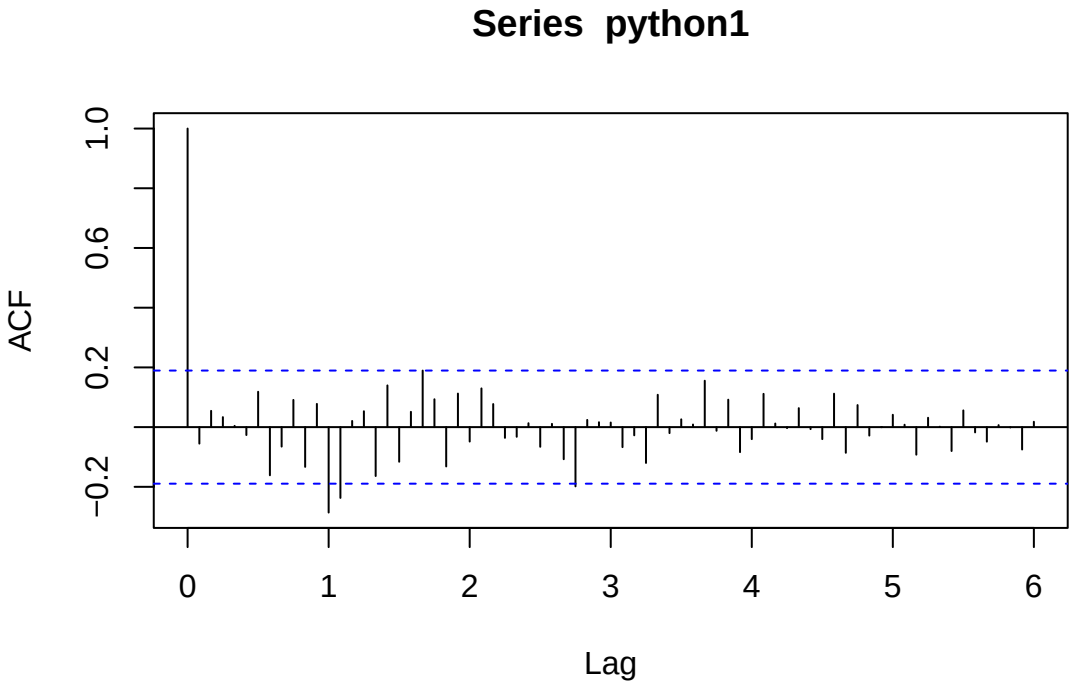
Now, the plot seems stationary, but to further ensure that Python Data is stationary, I apply the Dickey-Fuller Test to check whether the Python Data is stationary. The null hypothesis of the Dickey-Fuller Test is that the series is not stationary. So in order to seek stationary, I want to have a p-value smaller than 0.05.

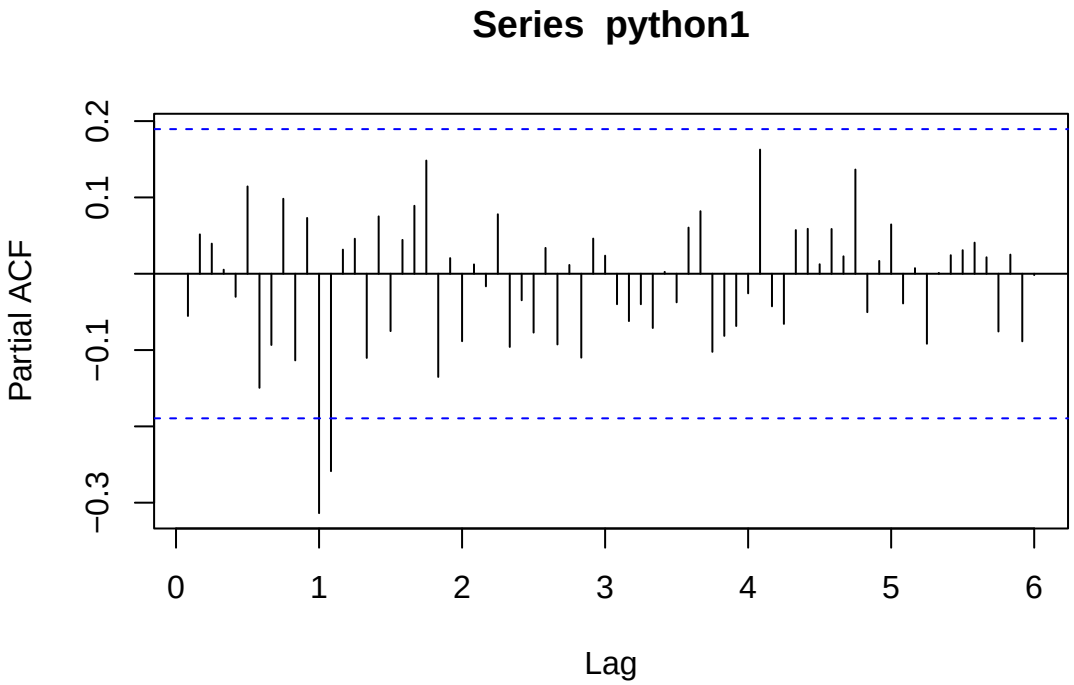
```
##
##  Augmented Dickey-Fuller Test
##
## data:  python1
## Dickey-Fuller = -4.379, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

Since the p-value results in 0.01 which is smaller than 0.05. Hence I reject the null hypothesis and conclude that the differenced Python Data is Stationary.

3.5 Sample ACF and PACF Plot of Python Data

I plot the ACF and PACF to prelimiary idenfy the candidates of the model.





The sample ACF plot does not have statistical significance between lag 1 and lag 11, but it appears to have statistical significance at lag 12 and lag 13. The sample PACF plot does not have statistical significance between lag 1 and lag 11 as well, but it also appears to have statistical significance at lag 12 and lag 13. And the sample ACF and PACF plots appears to be very similar to each other. Hence, I suggest the model to have p=7,12,13, q=7,12,13 with no SMA and SAR or SARIMA model with p=7,13, q=1,7,13, P=1, Q=1, then check if any these obtain a lower AICc. I also choose p=7 and q=7 here, because lags 7 seems to be statistically significant since it is around the confidence Interval.
Furthermore, since lag 13 is a quite big number, so I choose to fit the whole model as p=13 and q=13 first, then select the only coefficients that the confidence interval does not include 0.

4. Model Selection

The following sections will present the AICc of the candidates model and choose the one model with the lowest AICc as the model.

4.1 Checking AICc of the Candidates Model

AICc of the Candidates Models:

Table 1: AICc of the Candidates Models

	AICc
1. SARIMA(1,1,1)(1,1,1)_12	1046.16
2. SARIMA(13,1,13)(0,1,0)_12	1068.94
3. SARIMA(13,1,13)(0,1,0)_12 with Coef. ar7,12,13, ma7,12,13	1046.84
4. SARIMA(13,1,12)(0,1,0)_12 with Coef. ar7,12,13, ma7,12	1043.95
5. SARIMA(13,1,13)(0,1,1)_12 with Coef. ar7,13, ma13, sma1	1046.54
6. SARIMA(13,1,12)(1,1,1)_12 with Coef. ar7,13, ma13, sar1, sma1	1049.87
7. SARIMA(13,1,1)(0,1,1)_12 with Coef. ar7,13, ma1, sma1	1036.44
8. SARIMA(13,1,13)(0,1,1)_12 with Coef. ar13, ma13, sma1	1050.61
9. SARIMA(13,1,1)(0,1,1)_12 with Coef. ar13, ma1, sma1	1037.28

	AICc
10. SARIMA(13,1,13)(0,1,0)_12 with Coef. ar12,13, ma12,13	1050.57
11. SARIMA(13,1,0)(0,1,1)_12 with Coef. ar13, sma1	1037.12
12. SARIMA(13,1,0)(0,1,1)_12 with Coef. ar7,13, sma1	1036.26
13. SARIMA(13,1,0)(0,1,0)_12 with Coef. ar7,12,13	1038.73

By comparing the AICc of the candidates models, I choose the 7th model with AICc=1036.43509 as **Model A** and 12th model with AICc=1036.262255 as **Model B**, since these two models both have AICc=1036.

The fitted **model A** is displyed below:

```
##
## Call:
## arima(x = python.lam, order = c(13, 1, 1), seasonal = list(order = c(0, 1, 1),
##     period = 12), fixed = c(rep(0, 6), NA, rep(0, 5), NA, NA, NA), method = "ML")
##
## Coefficients:
##       ar1  ar2  ar3  ar4  ar5  ar6     ar7  ar8  ar9  ar10  ar11  ar12    ar13
##         0    0    0    0    0    0  -0.154    0    0     0     0     0  -0.333
## s.e.    0    0    0    0    0    0   0.091    0    0     0     0     0   0.098
##          ma1    sma1
##       -0.151  -0.430
## s.e.   0.096   0.091
##
## sigma^2 estimated as 790:  log likelihood = -510.91,  aic = 1031.82
```

Hence, the **selected model A** is:

$$(1 + 0.154B^7 + 0.333B^{13})(1 - B)(1 - B^{12})X_t = (1 - 0.151B)(1 - 0.430B^{12})Z_t$$

with $\hat{\sigma}_Z = 790$.

The fitted **model B** is displyed below:

```
##
## Call:
## arima(x = python.lam, order = c(13, 1, 0), seasonal = list(order = c(0, 1, 1),
##     period = 12), fixed = c(rep(0, 6), NA, rep(0, 5), NA, NA), method = "ML")
##
## Coefficients:
##       ar1  ar2  ar3  ar4  ar5  ar6     ar7  ar8  ar9  ar10  ar11  ar12    ar13
##         0    0    0    0    0    0  -0.157    0    0     0     0     0  -0.309
## s.e.    0    0    0    0    0    0   0.092    0    0     0     0     0   0.097
##         sma1
##       -0.431
## s.e.   0.092
##
## sigma^2 estimated as 810:  log likelihood = -512.13,  aic = 1032.26
```

Hence, the **selected model B** is:

$$(1 + 0.157B^7 + 0.309B^{13})(1 - B)(1 - B^{12})X_t = (1 - 0.431B^{12})Z_t$$

with $\hat{\sigma}_Z = 810$.

4.2 Checking the Model Roots

In this part, I will check if the MA and SMA parts are invertible for both **Model A** and **Model B**.

Model A:
Checking the Root for **MA** and **SMA** Part:
$1 - 0.151B = 0$ and $1 - 0.43B^{12} = 0$
Since the coefficients of MA1 $|\theta_1| = |-0.151| = 0.151 < 1$ and SMA1 $|\Theta_1| = |-0.430| = 0.430 < 1$. And according to the definition of invertibility for MA(1) and SMA(1), I can conclude that this model A is **Invertible**.

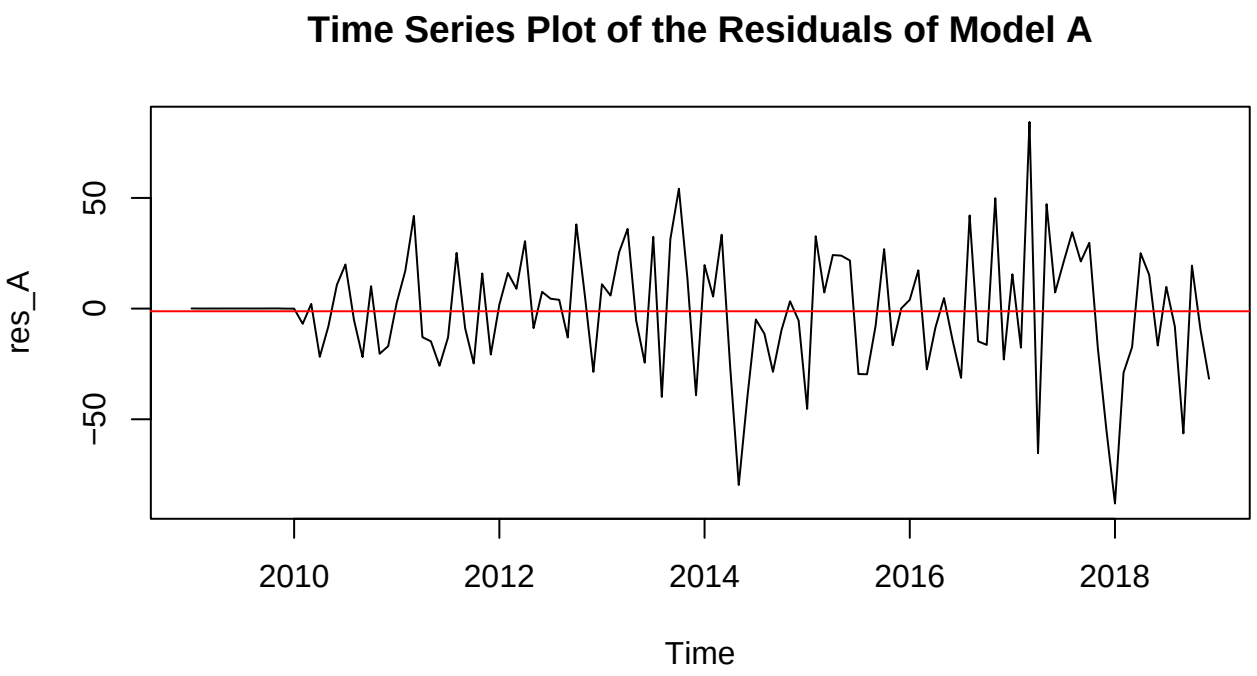
Model B:
Checking the Root for **SMA** Part:
$1 - 0.431B^{12} = 0$
Since the coefficient of SMA1 $|\Theta_1| = |-0.431| = 0.431 < 1$. And according to the definition of invertibility for SMA(1), I can conclude that this model B is **Invertible**.

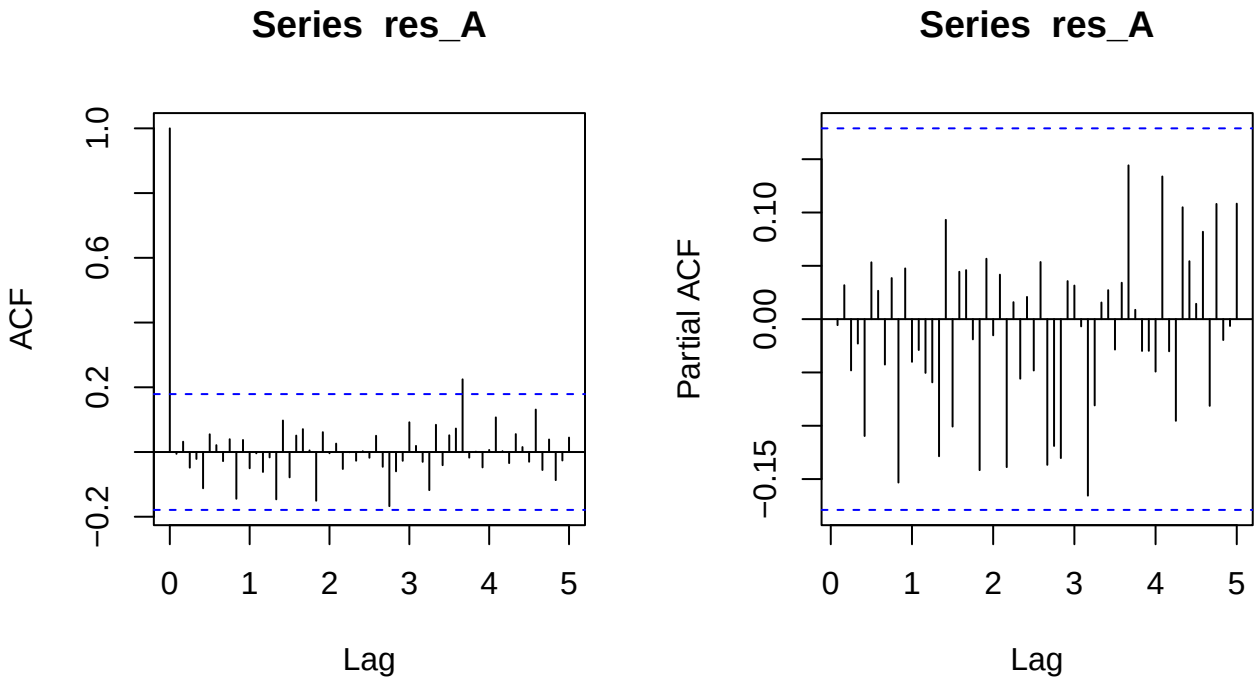
5. Model Diagnostics

After the selecting the best model that would fit the Python Data, I continue to check the residuals of the fitted model to make sure the model works well.

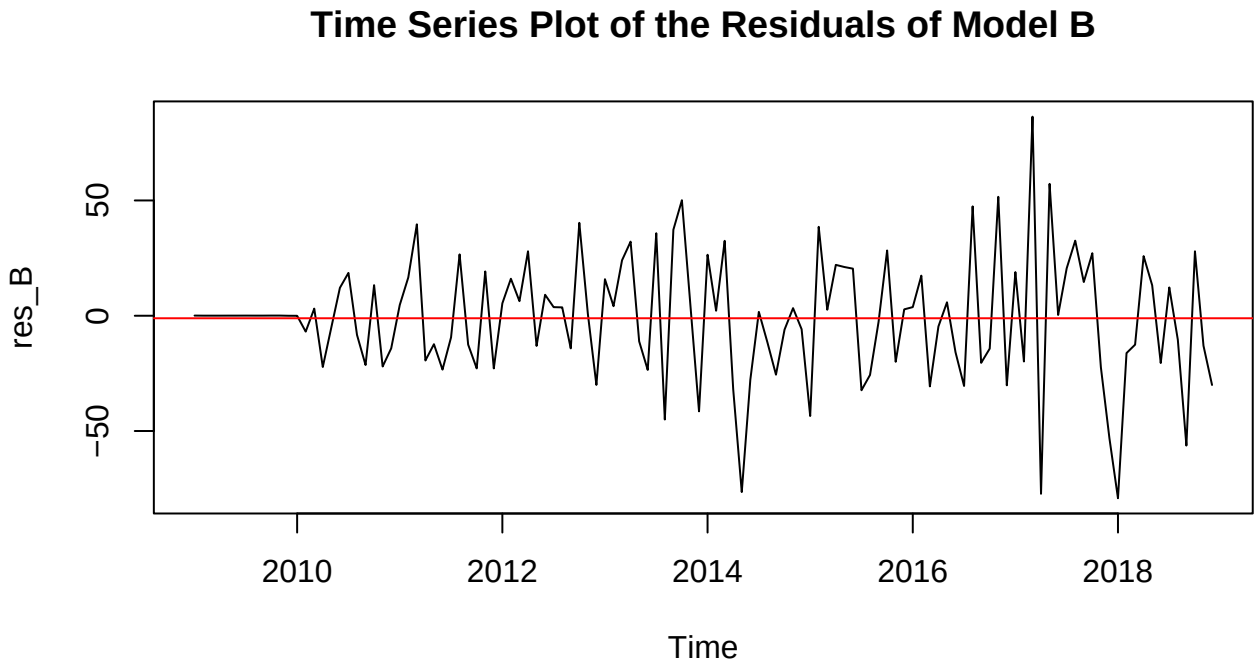
5.1 Checking the Time Series Plot, ACF, and PACF of the Residuals

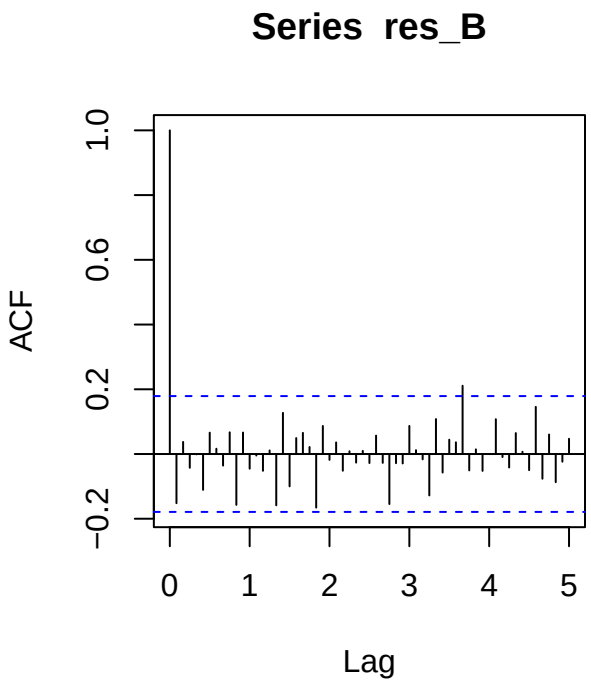
The time series plot, ACF, PACF of residuals of **Model A**:

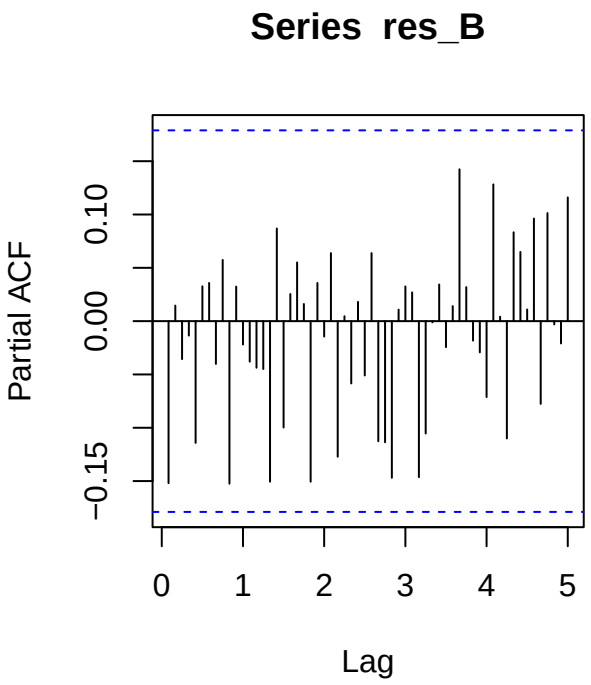




The time series plot, ACF, PACF of residuals of **Model B**:







From all the plots of these two models, I can see that he Time Series Plots of the Residuals have no trend, no visible change of variance, and no seasonality,and looks like white noise process, and the PACF of the residuals are all within the Confidence Interval and can be counted as zero. However, from both ACF plots of the residuals, $\hat{\rho}(44)$ are outside the Confidence Interval. Since the blue dash lines represents the 95% Confidence Interval, I would accept the spike at $\hat{\rho}(44)$ and consider the residuals of the fitted model might perform white noise process.

Then I continue to check if the residuals of these model would fit an AR(0) model.

For **Model A**, I can see that by plugging the fitted model residuals to the Yuler-Walker's method to see if the residual could fit in a AR(0) model (white nosie).

```
##
## Call:
## ar(x = res_A, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  709
```

Since automatically selected order is 0, which means the residuals of **Model A** follows a white noise process.

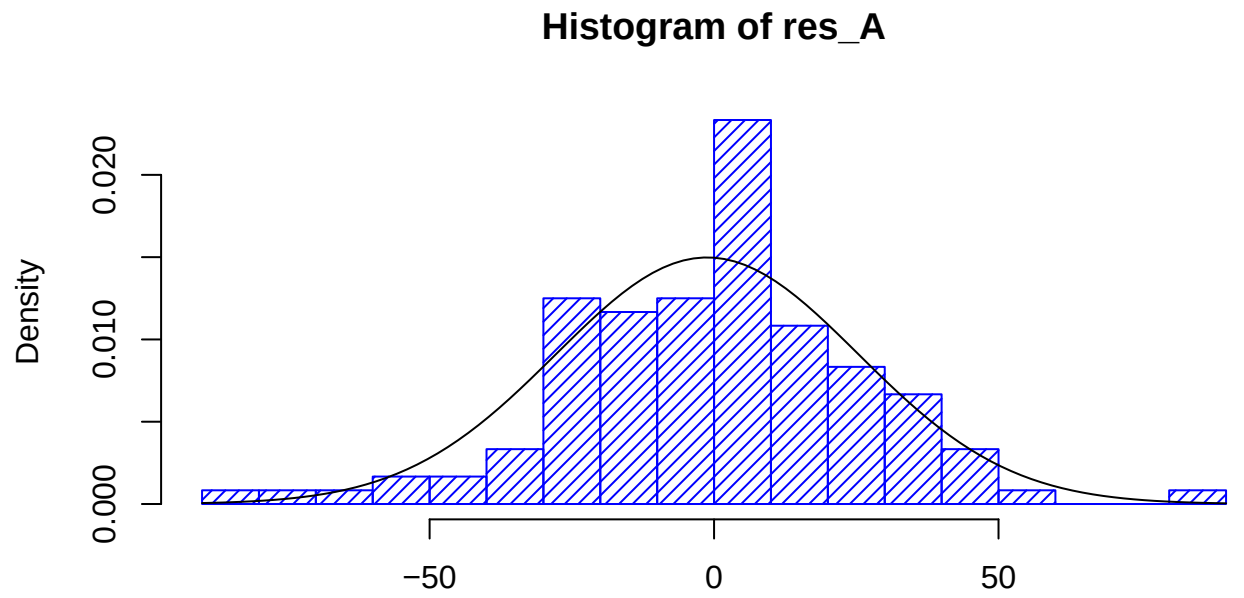
For **Model B**, I can see that by plugging the fitted model residuals to the Yuler-Walker's method to see if the residual could fit in a AR(0) model (white nosie).

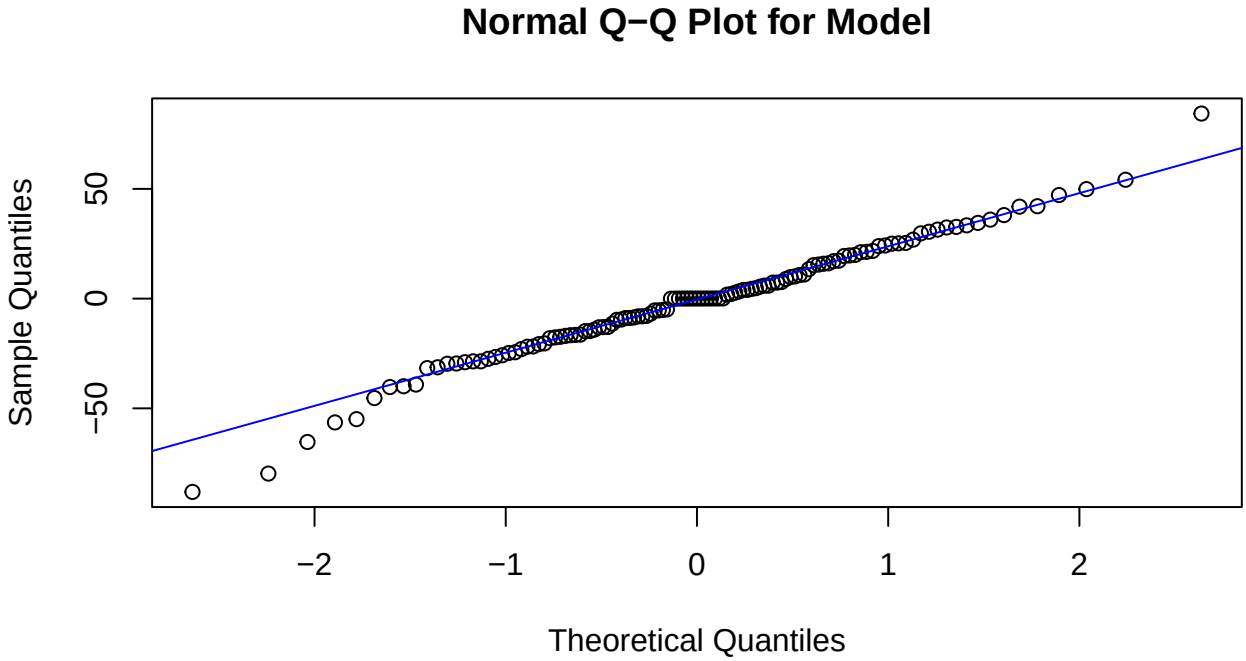
```
##
## Call:
## ar(x = res_B, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
## Coefficients:
##      1
## -0.152
##
## Order selected 1  sigma^2 estimated as  716
```

Since automatically selected order is 0, which means the residuals of **Model B** does not fit into AR(0) and it does now shows white noise process.
As a result, I will continue the diagnostic process with only **Model A**.

5.2 Normality

Checking if the residuals of **model A** are distributed noramlly.





Shapiro-Wilk Normality Test
Null Hypothesis: the variable is normally distributed is some population.

Table 2: Shapiro-Wilk Normality Test of Model A

	P-value
Shapiro-Wilk Test	0.127179

Since the p-value of the Shapiro test is larger than the significant value 0.05, then I do not need to reject the null hypothesis and conclude that the residuals are normally distributed.

5.3 Residual Independence

The number of observations in the Python Data is 120, I would consider $h = \sqrt{120} \approx 11$.

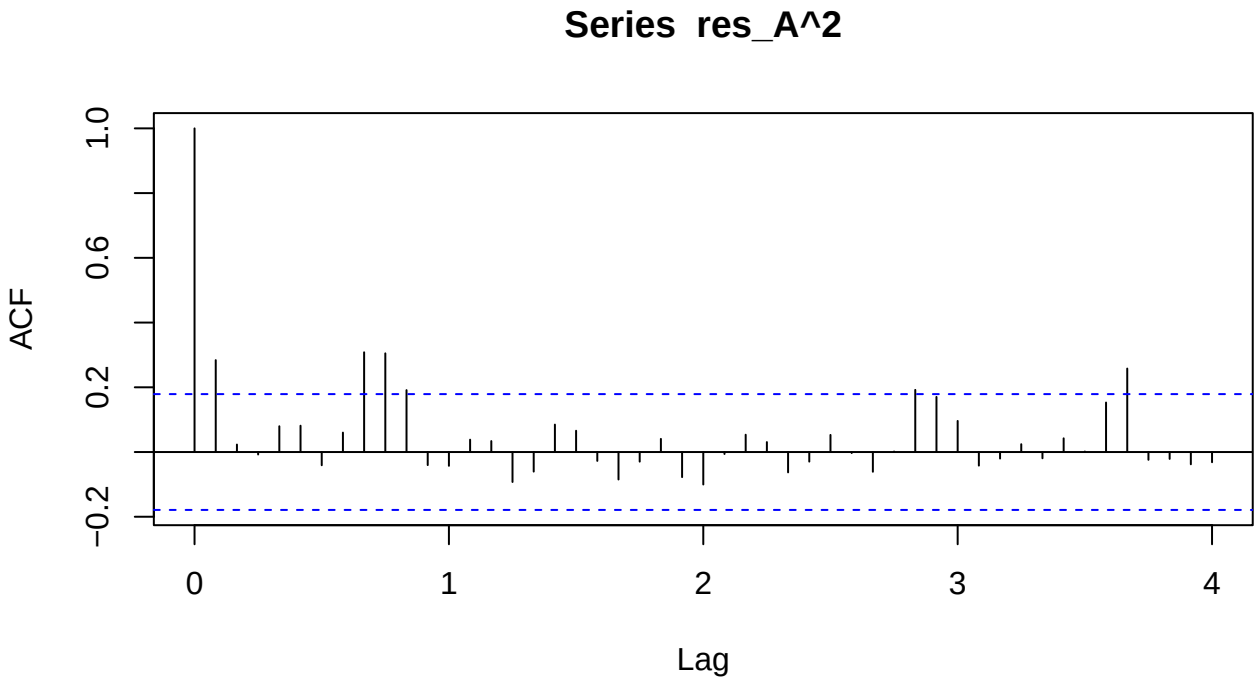
Box-Pierce: With degrees of freedom of 7.
Ljung-Box: With degrees of freedom of 7.
Mc-Leod Li: With degrees of freedom of 11.

Table 3: Box Test of Model A

	P-value
Box-Pierce	0.619524
Ljung-Box	0.564221
Mc-Leod Li	0.000016

The above table indicates that the **Box-Pierce Test** and **Ljung-Box Test** pass becasue the p-value of these two tests is larger than the significant value 0.05. However, the **Mc-Leod Li Test** does not seem passing which shows non-linear dependence of the residuals.

The following graph is the ACF plot of the residuals of the fitted model:

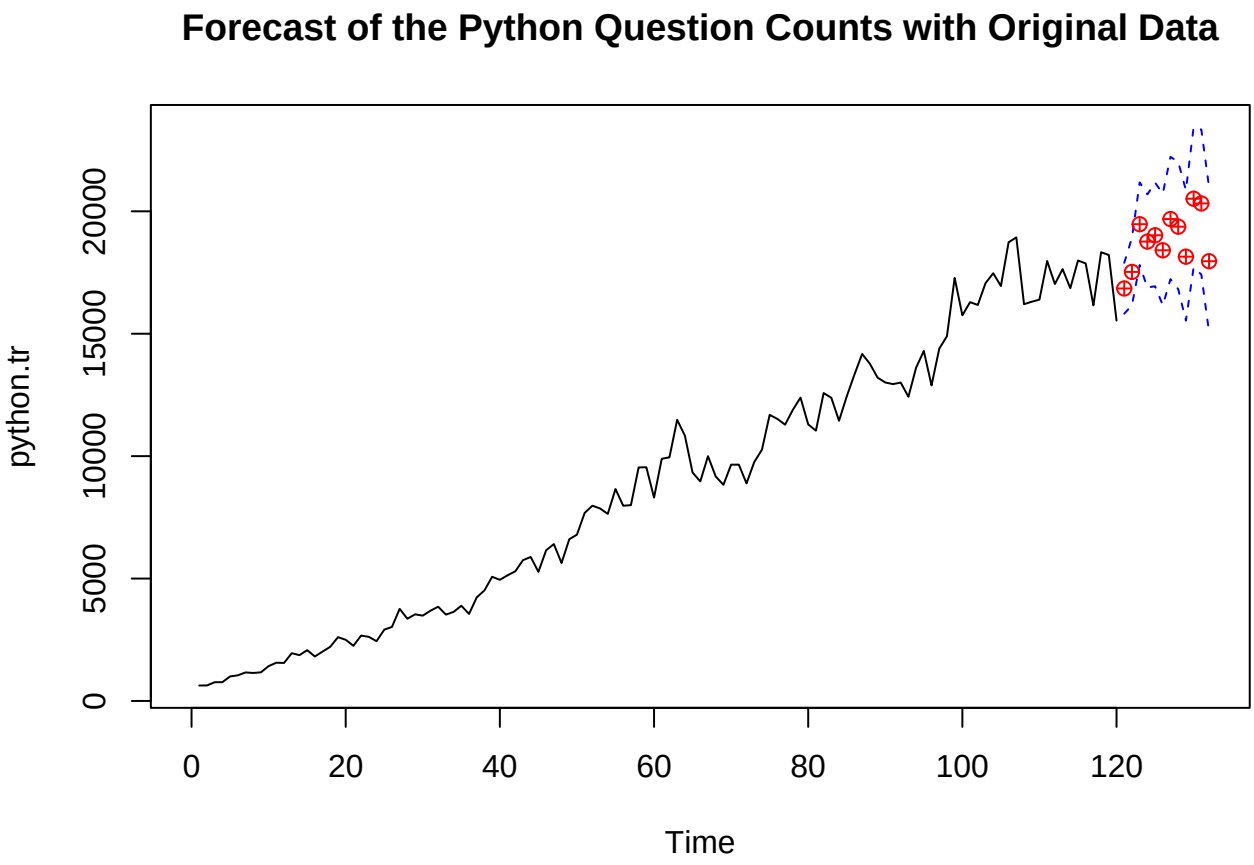


The ACF plot of the residuals demonstrates that the residual^2 has non-linear dependence, while the Mcleod-Li Test also shows that the p-value of the test is smaller than the significant value. However, I've tried other models but all of them shows non-linear dependence of the residuals. Hence, some non-linear models might needed to provide a better fit of this model.
To give some suggestions to solve the Mcleod-Li Test problem, I would suggest that choosing to add the arch-garch model or other non-linear models in the time series.

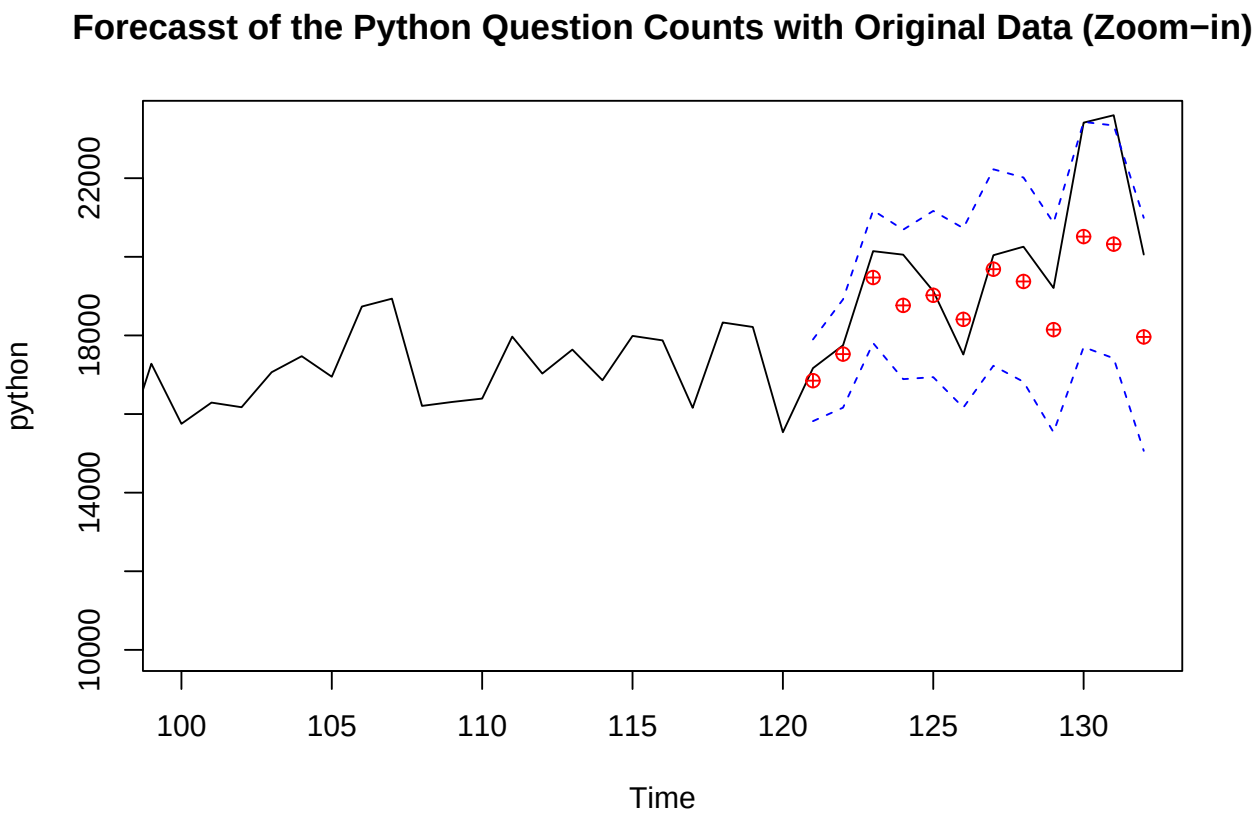
6. Forecast

Even though the the residuals of the fitted model shows non-linear dependence and non-linear models might be required, I try to forecast the currently best-selected linear model.

The following plot displays the 12-month forecast value and its 95% prediction interval:



Then I add the test data set and compared with the predicted values, and its 95% prediction interval:



The above plot displays the whole Python data, including both the training and testing sets. Comparing with the 95% prediction interval, since all the test data are within the interval except the Nov.2019 observation. Since this is a 95% prediction interval, I can conclude that although there exists non-linear dependence in the fitted model residuals, the model can be considered as an appropriate linear model.

7. Conclusion

By doing this time series forecast, I achieve my goal by fitting an appropriate linear model and forecast the future Python Question Counts. Throughout this project, I utilize the time series methods that I learned in PSTAT174 to fit an appropriate linear model, applying the Box-Cox Transformation lambda to set the transformation method, checking the ACF and PACF to consider candidates models. The two possible models are:

Model A:

$$(1 + 0.154B^7 + 0.33B^{13})(1 - B)(1 - B^{12})X_t = (1 - 0.151B)(1 - 0.430B^{12})Z_t$$

Model B:

$$(1 + 0.157B^7 + 0.309B^{13})(1 - B)(1 - B^{12})X_t = (1 - 0.431B^{12})Z_t$$

Then I apply the model diagnostics to check if the residuals of these two models behave as what I expected. And I find out that **Model A** appears to be a better model. However, the Python Data results in some necessary non-linear model might be needed to obtain a better model. Finally, I proceed to the forecast process and plot the 95% prediction interval of the 2019 Python Question Counts.

Last but not least, I want to thank professor Feldman for taking time helping me with the model fit and dealing with the non-linear dependence of the residuals. And TA Youhong Lee and TA Sunpeng Duan who helped me lot when I had problems fitting the model.

8. References

1. StackOverflow Questions Count Time Series, Kaggle. https://www.kaggle.com/aishu200023/stackindex
2. PSTAT 174 Lecture slides, Lecture Notes, Labs
3. Special thanks to Professor Feldman, TA Youhong Lee and TA Sunpeng Duan.

Appendix: R code

```
library(knitr)
knitr::opts_chunk$set(echo=FALSE,warning=FALSE, fig.width=7, fig.height=4, fig.align='center')
options(digits = 6)
library(tidyr)
library(dplyr)
library(lubridate)
library(qpcR)
library(forecast)
library(MASS)
library(tseries)
source("plot.roots.R")
# loading the data
counts = read.csv("MLTollsStackOverflow.csv")
python = counts[,5]
name=c('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sept', 'Oct', 'Nov', 'Dec')
py=python[1:12]
df=data.frame(rbind(name, py))
colnames(df)=c('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sept', 'Oct', 'Nov', 'Dec')
df[2,]
# split into training and test data
python.tr=python[1:120]
python.te=python[121:132]
# plot of the times series
python.ts = ts(python.tr, start=c(2009,1), frequency=12)
ts.plot(python.ts)
title("StackOverFlow Python Question Counts, 2009-2019", font= 2)
t = 1:length(python.ts)
fit = lm(python.ts ~ t)
bcTransform = boxcox(python.ts ~ t,plotit = TRUE, lambda = seq(0.5, 1, 0.01))

lambda = bcTransform$x[which(bcTransform$y == max(bcTransform$y))]
python.lam=(python.ts)^(lambda)
# compare the original and transform
par(mfrow=c(1,2))
hist(python.tr, col="#66CCFF")
hist(python.lam, col="#66CCFF")
# difference at lag 12 once
python12=diff(python.lam, 12)
plot(python12)
abline(h=mean(python12), col="red")
title("Python Data Differenced at lag 12 once")
# de-trend the data once
python1 = diff(python12,1)
plot(python1)
abline(h=mean(python1), col="red")
```

```
title("Python Data Difference at lag 12 and lag 1")
adf.test(python1)
# acf
acf(python1, lag.max=72)
# pacf
pacf(python1, lag.max=72)
# fit all possible values
fit1=arima(python.lam, order=c(13,1,13),
               seasonal = list(order = c(0,1,0), period = 12), method="ML")
fit2=arima(python.lam, order=c(13,1,13),
               fixed=c(rep(0,6),NA,rep(0,4),NA,NA,rep(0,6),NA,rep(0,4),NA,NA),
               seasonal = list(order = c(0,1,0), period = 12), method="ML")
fit3=arima(python.lam, order=c(13,1,12),
               fixed=c(rep(0,6),NA,rep(0,4),NA,NA,rep(0,6),NA,rep(0,4),NA),
               seasonal = list(order = c(0,1,0), period = 12), method="ML")
fit4=arima(python.lam, order=c(13,1,13),
               fixed=c(rep(0,6),NA,rep(0,5),NA,rep(0,12),NA,NA),
               seasonal = list(order = c(0,1,1), period = 12), method="ML")
fit5=arima(python.lam, order=c(13,1,13),
               fixed=c(rep(0,6),NA,rep(0,5),NA,rep(0,12),NA,NA,NA),
               seasonal = list(order = c(1,1,1), period = 12), method="ML")
fit6=arima(python.lam, order=c(13,1,1),
               fixed=c(rep(0,6),NA,rep(0,5),NA,NA,NA),
               seasonal = list(order = c(0,1,1), period = 12), method="ML")
fit7=arima(python.lam, order=c(13,1,13),
               fixed=c(rep(0,12),NA,rep(0,12),NA,NA),
               seasonal = list(order = c(0,1,1), period = 12), method="ML")
fit8=arima(python.lam, order=c(13,1,1),
               fixed=c(rep(0,12),NA,NA,NA),
               seasonal = list(order = c(0,1,1), period = 12), method="ML")
fit=arima(python.lam, order=c(1,1,1),
               fixed=c(NA,NA,NA,NA),
               seasonal = list(order = c(1,1,1), period = 12), method="ML")
fit9=arima(python.lam, order=c(13,1,13),
               fixed=c(rep(0,11),NA,NA,rep(0,11),NA,NA),
               seasonal = list(order = c(0,1,0), period = 12), method="ML")
fit10=arima(python.lam, order=c(13,1,0),
            fixed=c(rep(0,12),NA,NA),
            seasonal = list(order = c(0,1,1), period = 12), method="ML")
fit11=arima(python.lam, order=c(13,1,0),
            fixed=c(rep(0,6),NA,rep(0,5),NA,NA),
            seasonal = list(order = c(0,1,1), period = 12), method="ML")
fit12=arima(python.lam, order=c(13,1,0),
            fixed=c(rep(0,6),NA,rep(0,4),NA,NA),
            seasonal = list(order = c(0,1,0), period = 12), method="ML")
model.name=c('1. SARIMA(1,1,1)(1,1,1)_12',
             '2. SARIMA(13,1,13)(0,1,0)_12',
             '3. SARIMA(13,1,13)(0,1,0)_12 with Coef. ar7,12,13, ma7,12,13',
             '4. SARIMA(13,1,12)(0,1,0)_12 with Coef. ar7,12,13, ma7,12',
             '5. SARIMA(13,1,13)(0,1,1)_12 with Coef. ar7,13, ma13, sma1',
             '6. SARIMA(13,1,12)(1,1,1)_12 with Coef. ar7,13, ma13, sar1, sma1',
             '7. SARIMA(13,1,1)(0,1,1)_12 with Coef. ar7,13, ma1, sma1',
             '8. SARIMA(13,1,13)(0,1,1)_12 with Coef. ar13, ma13, sma1',
```

```
              '9. SARIMA(13,1,1)(0,1,1)_12 with Coef. ar13, ma1, sma1',
              '10. SARIMA(13,1,13)(0,1,0)_12 with Coef. ar12,13, ma12,13',
              '11. SARIMA(13,1,0)(0,1,1)_12 with Coef. ar13, sma1',
              '12. SARIMA(13,1,0)(0,1,1)_12 with Coef. ar7,13, sma1',
              '13. SARIMA(13,1,0)(0,1,0)_12 with Coef. ar7,12,13')
AICc=c(AICc(fit),AICc(fit1), AICc(fit2), AICc(fit3), AICc(fit4),
       AICc(fit5), AICc(fit6), AICc(fit7) ,AICc(fit8), AICc(fit9),
       AICc(fit10), AICc(fit11),AICc(fit12))
aicc.table=matrix(AICc, nrow=13, ncol=1)
rownames(aicc.table)=model.name
colnames(aicc.table)=c('AICc')
# table
knitr::kable(aicc.table, caption = "AICc of the Candidates Models")
fit6
fit11
# acf/pacf
res_A=residuals(fit6)
plot.ts(res_A)
abline(h=mean(res_A), col="red")
title('Time Series Plot of the Residuals of Model A')
par(mfrow=c(1,2))
acf(res_A, lag.max=60)
pacf(res_A, lag.max=60)
# acf/pacf
res_B=residuals(fit11)
plot.ts(res_B)
abline(h=mean(res_B), col="red")
title('Time Series Plot of the Residuals of Model B')
par(mfrow=c(1,2))
acf(res_B, lag.max=60)
pacf(res_B, lag.max=60)
# check is the residual fit the ar model, white noise
ar(res_A, aic = TRUE, order.max = NULL, method = c("yule-walker"))
ar(res_B, aic = TRUE, order.max = NULL, method = c("yule-walker"))
# normality, model A
hist(res_A,density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m=mean(res_A)
std=sqrt(var(res_A))
curve(dnorm(x,m,std), add=TRUE)
# Q-Q plot
qqnorm(res_A,main= "Normal Q-Q Plot for Model")
qqline(res_A,col="blue")
# test if residual is normally distributed
shapiro=shapiro.test(res_A)
shapiro.tb=matrix(c(shapiro$p.value),nrow=1,ncol=1)
rownames(shapiro.tb)=c('Shapiro-Wilk Test')
colnames(shapiro.tb)=c('P-value')
knitr::kable(shapiro.tb, caption = "Shapiro-Wilk Normality Test of Model A")
# residual independence
# Box test
# lag= sqrt(number of observations)
box.pierce=Box.test(res_A, lag = 11, type = c("Box-Pierce"), fitdf = 4)
ljung.box=Box.test(res_A, lag = 11, type = c("Ljung-Box"), fitdf = 4)
```

```
mcleod.li=Box.test(res_A^2, lag = 11, type = c("Ljung-Box"), fitdf = 0)
boxnames=c('Box-Pierce','Ljung-Box','Mc-Leod Li')
boxtest=matrix(c(box.pierce$p.value,ljung.box$p.value,mcleod.li$p.value),nrow=3, ncol=1)
rownames(boxtest)=boxnames
colnames(boxtest)=c('P-value')
# table
knitr::kable(boxtest, caption = "Box Test of Model A")
# acf plot of res to check the non-linear dependence
acf(res_A^2, lag.max=48)
forecast(fit6)
pred.tr = predict(fit6, n.ahead = 12)
python.ld=python.tr^(lambda)
# CI
U.tr= pred.tr$pred + 2*pred.tr$se
L.tr= pred.tr$pred - 2*pred.tr$se
# change to the original data
pred.orig = (pred.tr$pred)^(1/lambda)
U= (U.tr)^(1/lambda)
L= (L.tr)^(1/lambda)
# plot
ts.plot(python.tr, xlim=c(0,132), ylim = c(min(python.tr),max(U)))
lines((length(python.tr)+1):(length(python.tr)+12), U, col="blue", lty="dashed")
lines((length(python.tr)+1):(length(python.tr)+12), L, col="blue", lty="dashed")
points((length(python.tr)+1):(length(python.tr)+12), pred.orig, col="red", pch=10)
title("Forecast of the Python Question Counts with Original Data")
# To zoom the graph, starting from entry 100:
ts.plot(python, xlim = c(100,length(python.tr)+12), ylim = c(10000,max(U)))
lines((length(python.tr)+1):(length(python.tr)+12), U, col="blue", lty="dashed")
lines((length(python.tr)+1):(length(python.tr)+12), L, col="blue", lty="dashed")
points((length(python.tr)+1):(length(python.tr)+12), pred.orig, col="red", pch=10)
title("Forecasst of the Python Question Counts with Original Data (Zoom-in)")
```