Google search for 'Python' programming language

Kushal Kharel

11/28/2021

Suppose we have a sequence of random variables $X_t : t = 0, 1, 2, ...$ which are stochastic in nature then the probabilisitic structure of such a process can be determined by set of distributions of all finite collections of X which will serve as our model for time series analysis.In order to make any kind of statistical inference on the structure of such stochastic process which are observed over time, we make an assumption of stationarity that is the probability law do not change over time.When $n = 1$, the univariate distribution of X_t is same as X_{t-k}, $\forall t$ and k. The mean function and variance are both constant over time.

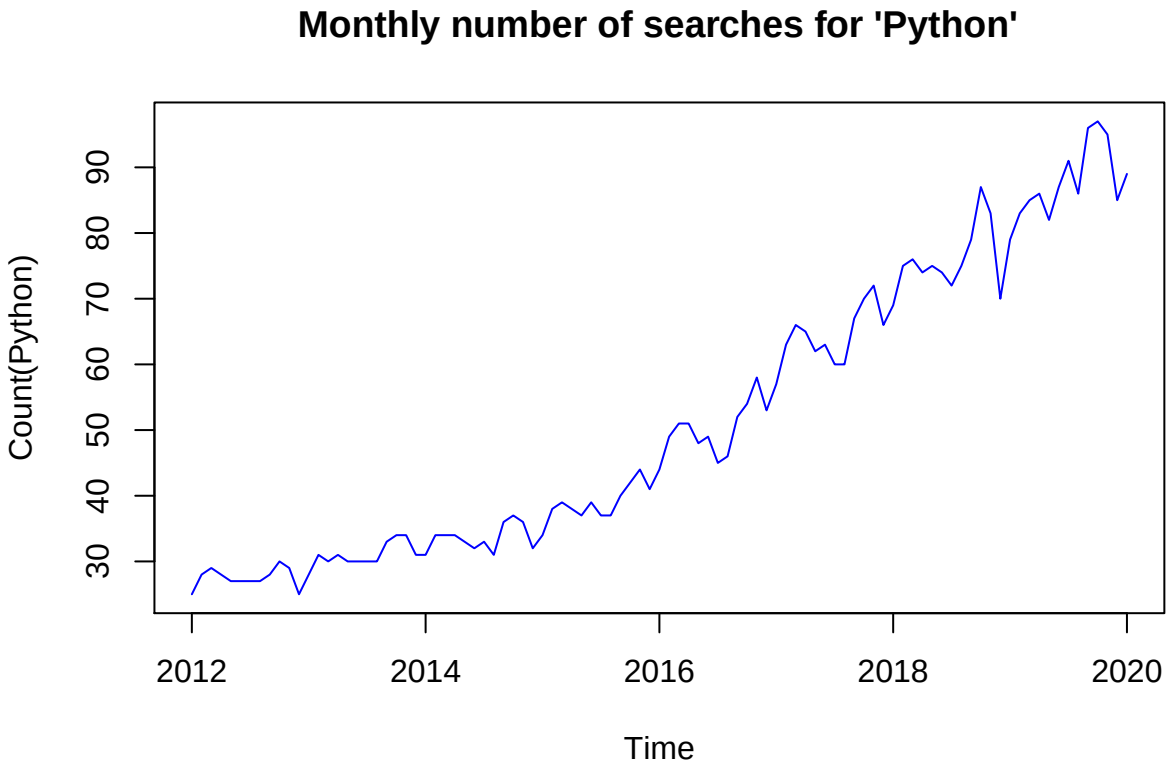
```
ts = read.csv("multiTimeline.csv", header = TRUE)

# Converting into time series object

ts = ts(ts)

# Checking the structure of the data
str(ts)
```

```
##  Time-Series [1:215, 1:2] from 1 to 215: 1 2 3 4 5 6 7 8 9 10 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:2] "Month" "Python"
```

```
# Creating sequence of dates
dates = seq(from=as.Date("2004-01-01"), to = as.Date("2021-11-28"), by = 'month')

# Extracting the values "Number of searches for Python Keyword"
x = ts[,2]

# Combining both and converting into time series object ordered by dates
Python = xts(x=x,order.by = dates)


# Monthly number of search terms for Python
# Log transforming the values
Python_F <- ts(Python, start = c(2004,1), end = c(2021,11), frequency = 12,)

# Subsetting Python series (Dec 2004 to Dec 2020)

Python_Series = window(Python_F, start = c(2012,1), end = c(2020,1))

# Quick look at the final series

head(Python_Series,50)
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 2012  25  28  29  28  27  27  27  27  28  30  29  25
## 2013  28  31  30  31  30  30  30  30  33  34  34  31
## 2014  31  34  34  34  33  32  33  31  36  37  36  32
## 2015  34  38  39  38  37  39  37  37  40  42  44  41
## 2016  44  49
```

```
# Plotting the series
plot(Python_Series, ylab = "Count(Python)", main = "Monthly number of searches for 'Python'", type = "l
```



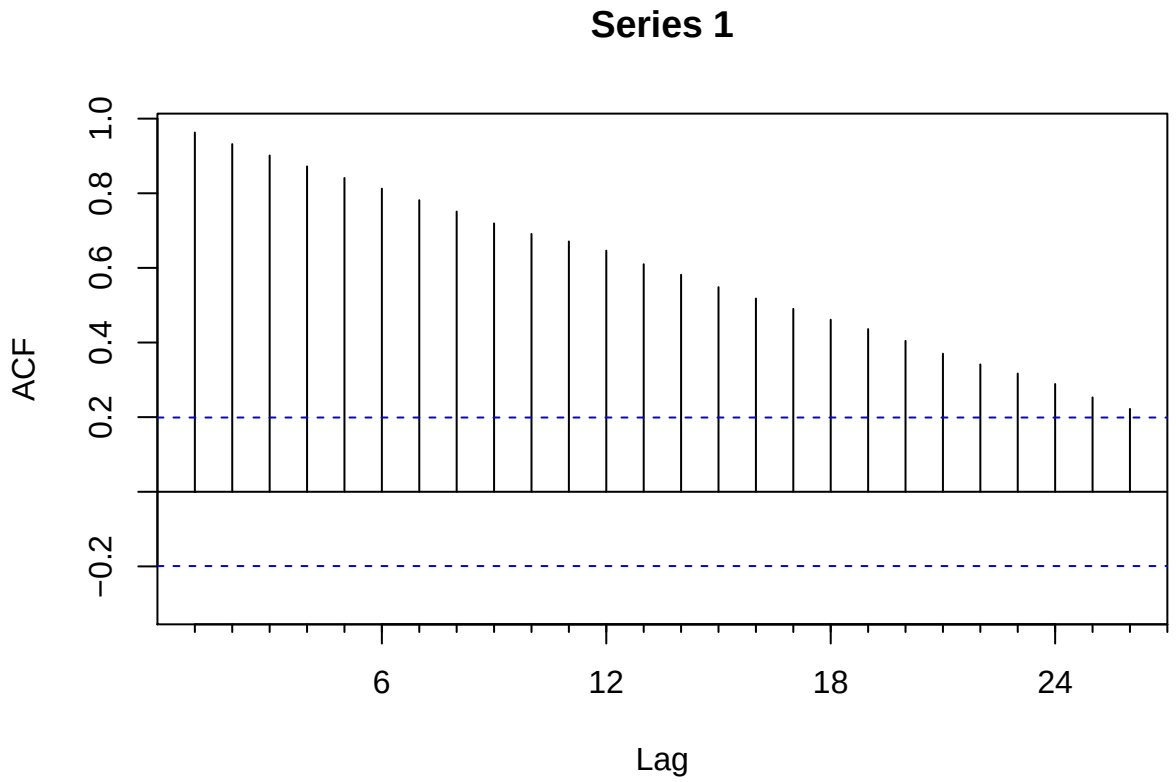
From the plot above, we can see that the plot is clearly trending upwards. We can also see there are some seasonal patterns in the times series. We have a upward trending data which suggests that the data is not stationary. To test for stationarity, let us perform Dickey-Fuller Test. Dickey-Fuller tests the null hypothesis of nonstationarity by performing unit root tests on autoregressive model and visualize the ACF and PACF plots. ACF and PACF helps us choose the best p and q values. ACF identifies how correlated are our time series values with each other. It plots the correlation coefficients against lags measured in number of periods. Lag represents a point in time after which we observe the first time series value.

```
# Dickey-Fuller Test for stationarity
adf.test(Python_Series)
```

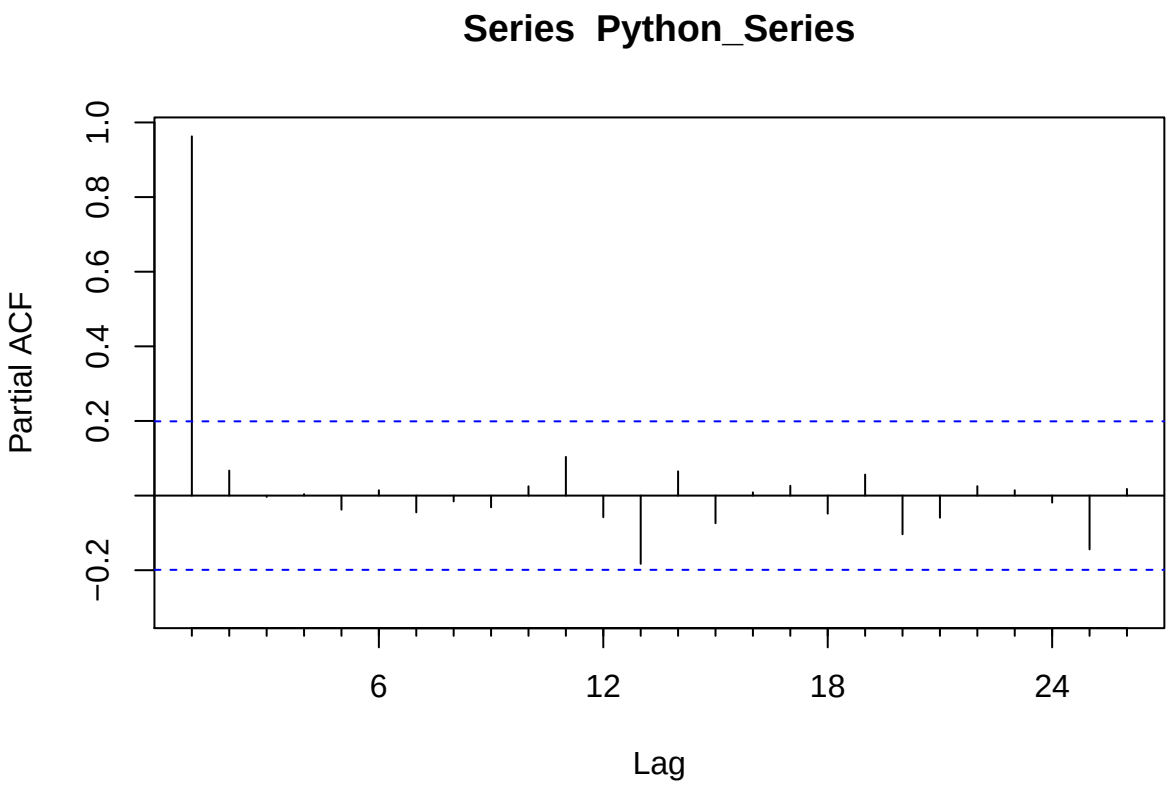
```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag  ADF p.value
```

```
## [1,]   0 1.56   0.969
## [2,]   1 1.64   0.975
## [3,]   2 2.70   0.990
## [4,]   3 3.72   0.990
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0 -0.328   0.913
## [2,]   1 -0.161   0.936
## [3,]   2  0.515   0.985
## [4,]   3  1.120   0.990
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -3.23  0.0875
## [2,]   1 -3.30  0.0760
## [3,]   2 -2.55  0.3458
## [4,]   3 -2.12  0.5199
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
Acf(Python_Series, lag.max = 26)
```



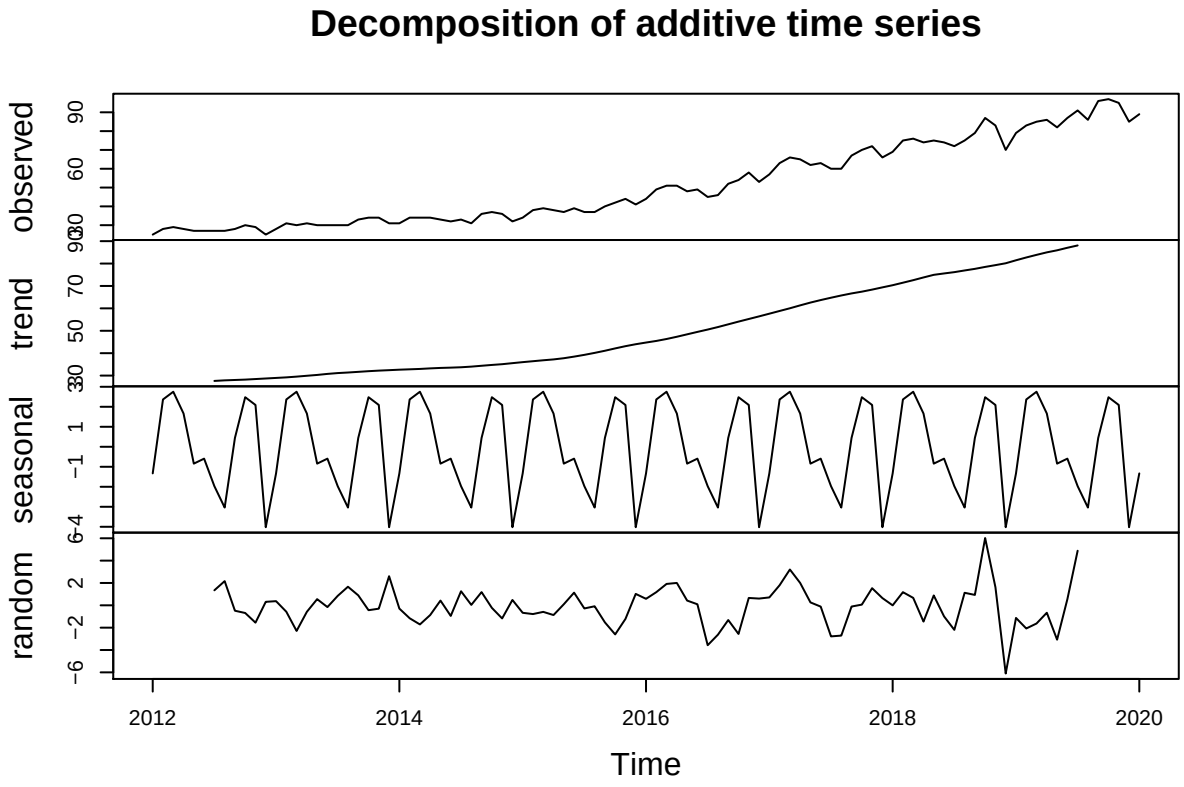

```
Pacf(Python_Series, lag.max = 26)
```



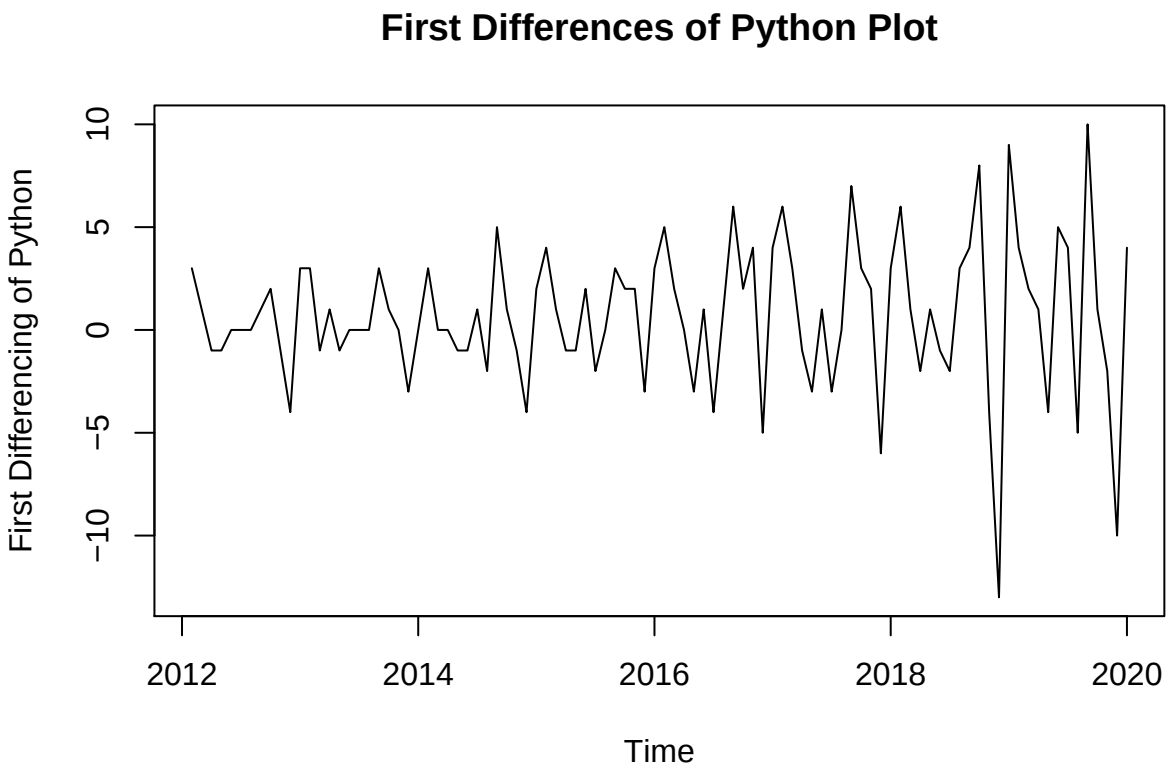
We can see that there are three types of model from the results, (Type 1, Type 2, Type 3).P-value is significantly higher than expected for all the models which supports the null hypothesis that our data is not stationary. The ACF declines more slowly and does not decay to zero which further suggests that the series is non-stationary and needs to be differenced.

Now, let us break up the trend, seasonality and residual(noise) components separately using decomposition method.

```
# Decomposing time series component
timeseriescomponents <- decompose(Python_Series)
plot(timeseriescomponents)
```




```
# Let us make the series stationary by removing the trend using first differencing
first_differenced_series = diff(Python_Series)
plot(first_differenced_series,ylab="First Differencing of Python", xlab="Time",main="First Differences
```



We can see from the plot that taking the first difference of the series resulted in constant mean and variance around zero. Note that the variance is increasing with time.

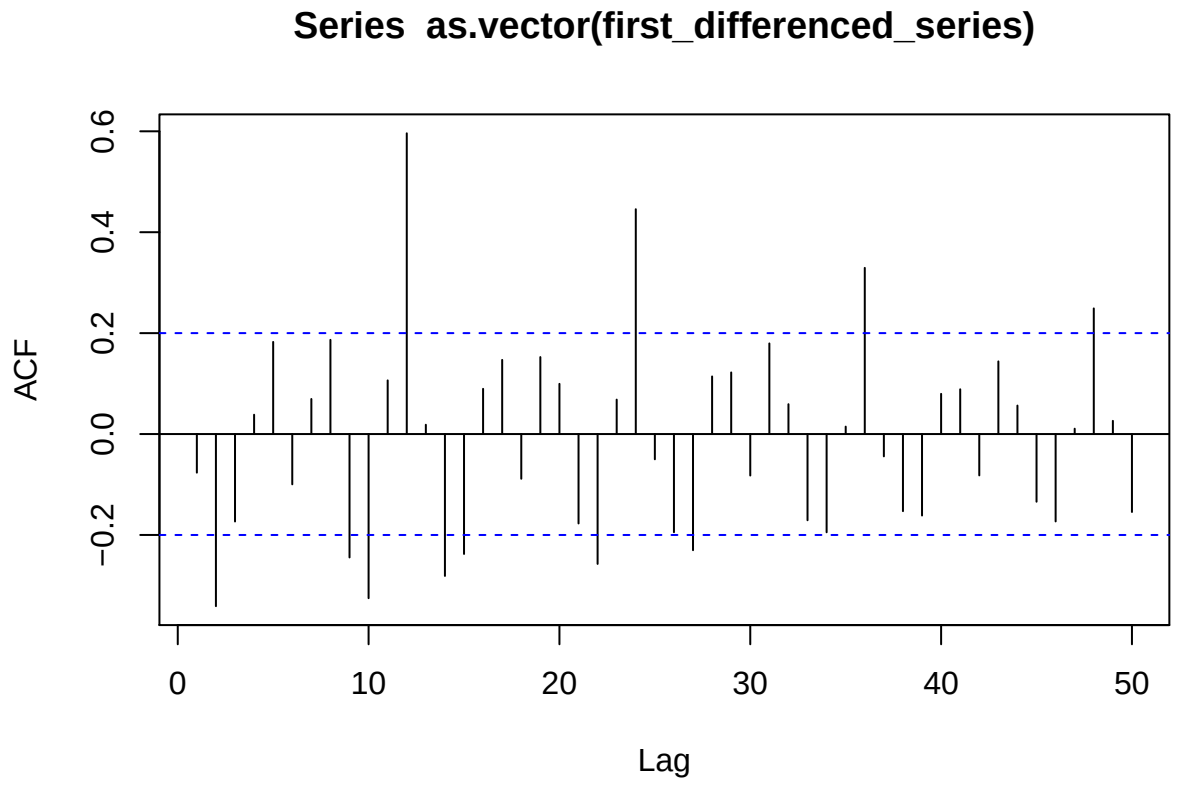
```
# Let us try DF test again on differenced data
adf.test(diff(Python_Series))
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -10.10    0.01
## [2,]   1  -9.51    0.01
## [3,]   2  -7.96    0.01
## [4,]   3  -6.25    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0 -10.40    0.01
## [2,]   1 -10.21    0.01
## [3,]   2  -9.17    0.01
## [4,]   3  -7.93    0.01
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0 -10.39    0.01
## [2,]   1 -10.30    0.01
## [3,]   2  -9.43    0.01
```

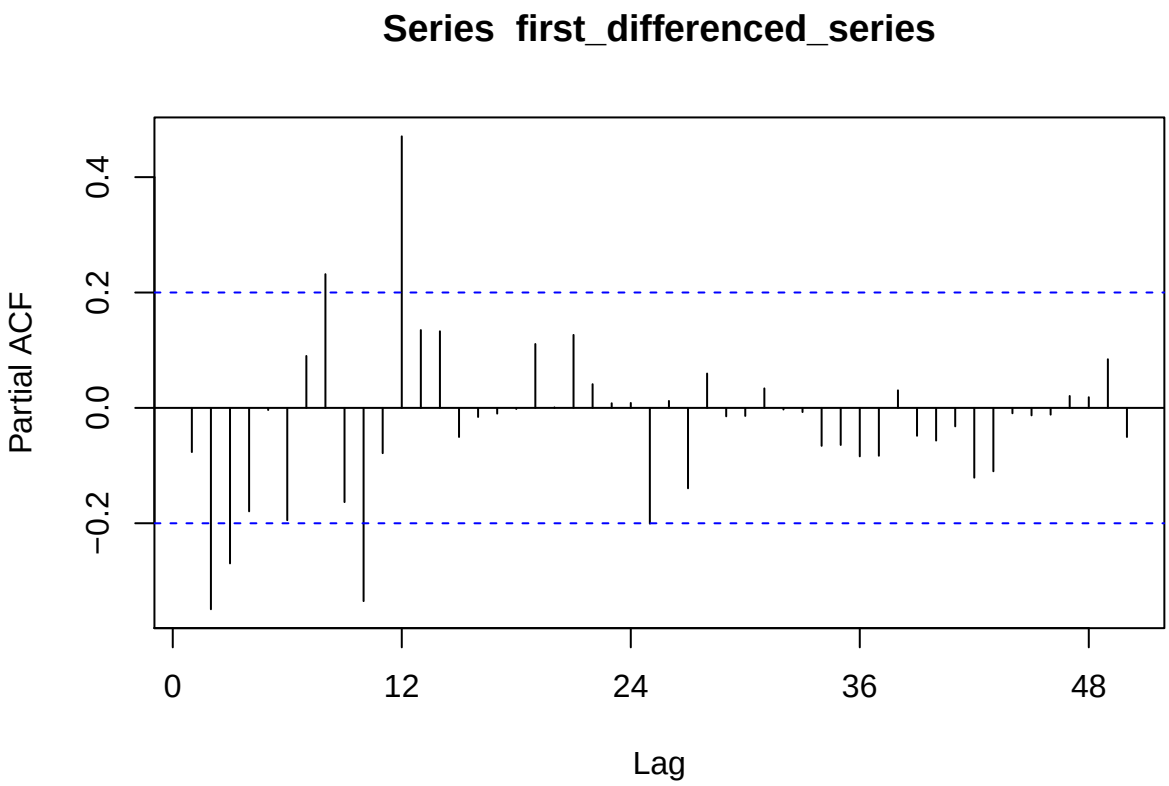
```
## [4,]   3  -8.43    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

The p-value is significantly low indicating the differenced series is stationary. Now, let us check the acf and pacf plot of differenced series.

```
Acf(as.vector(first_differenced_series), lag.max= 50)
```



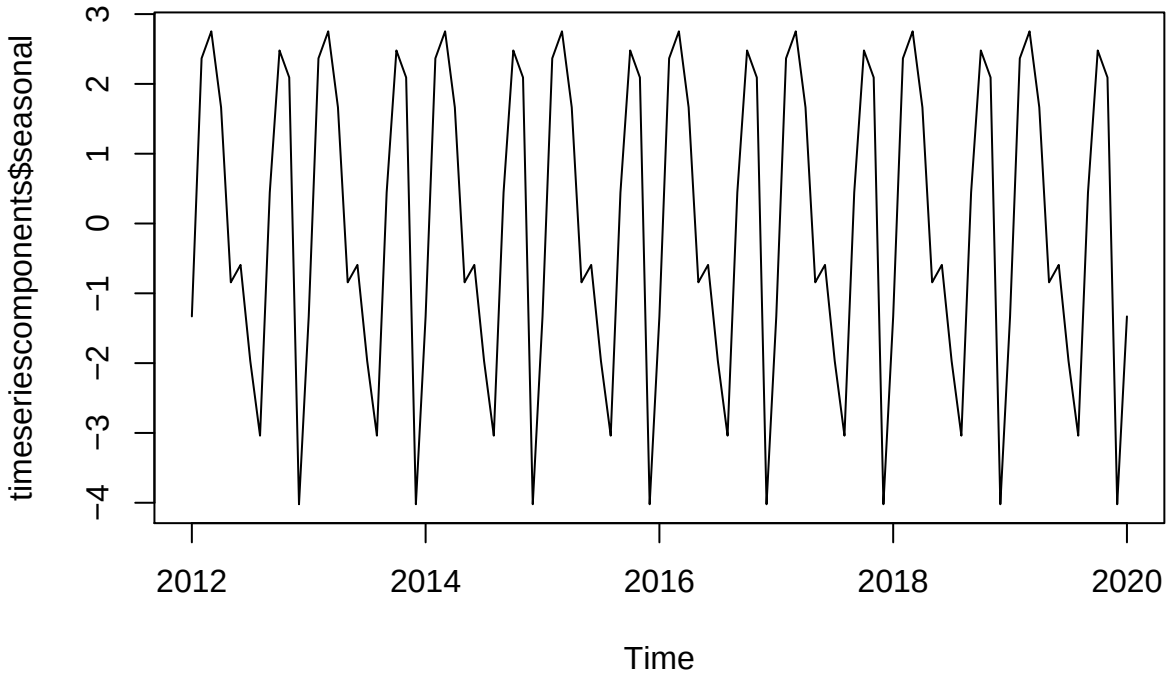

```
Pacf(first_differenced_series, lag.max = 50)
```



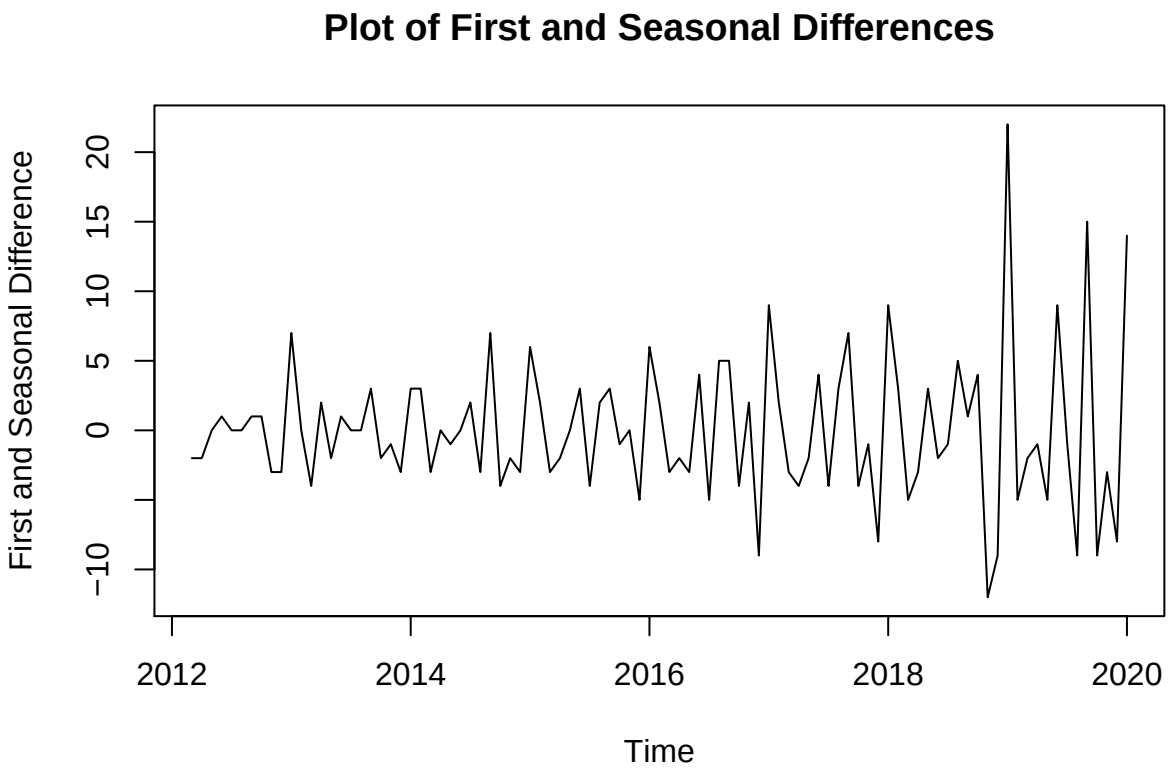
We can see from the ACF plot that there is a big negative spike at lag 2 followed by negative and positive spikes which suggests that those lags are either not statistically significant or missed the bound of significance by few extra points. Similarly, the PACF plot shows a spike at lag 2,3,8 and 12. We choose the order 2 for the purpose of this analysis.Also, there is a steady decline of values towards zero. Even after applying the first difference, we can still observe slow residual decay in ACF plots hence suggesting seasonal difference. This also suggests that the series have the characteristics of AR(2) and MA(2) process.

We can also see the seasonal component of the time series. To seasonally adjust the number of search terms for Python by month, we can estimate the seasonal component using decompose function and then subtract the seasonal component from the original time series

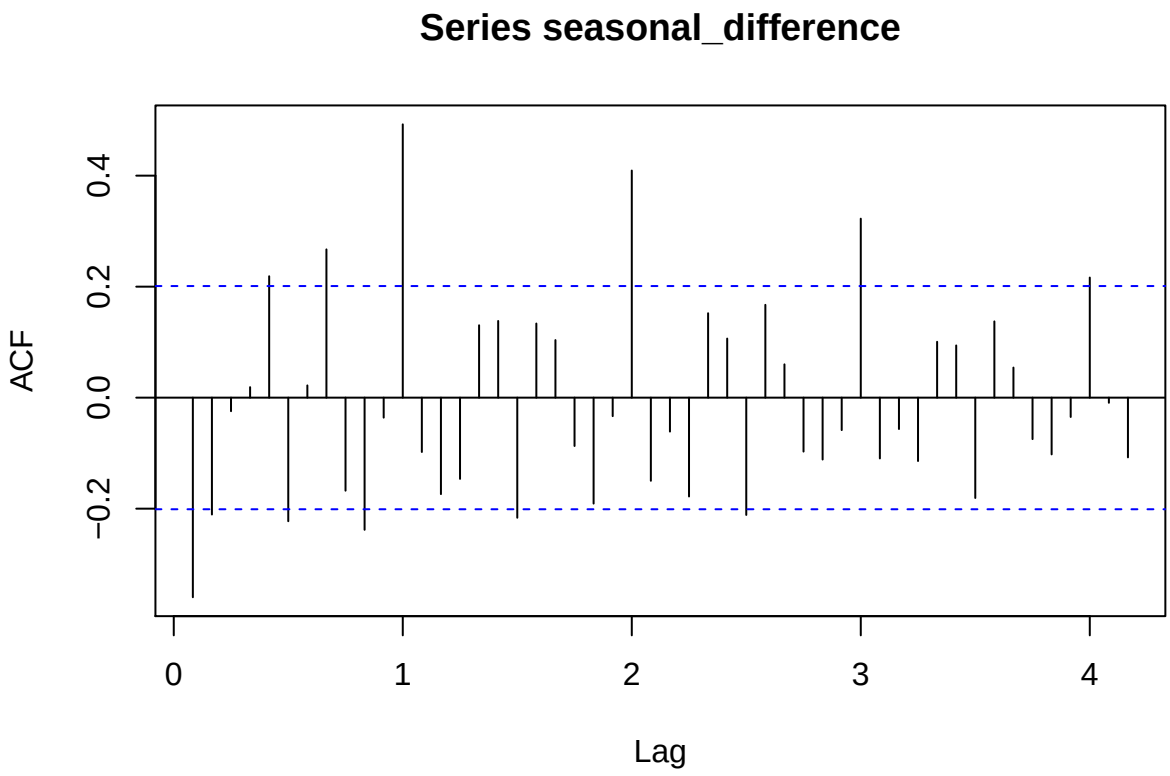
```
seasonal_difference = diff(diff(Python_Series))

# Plot of only seasonal component
plot(timeseriescomponents$seasonal)
```



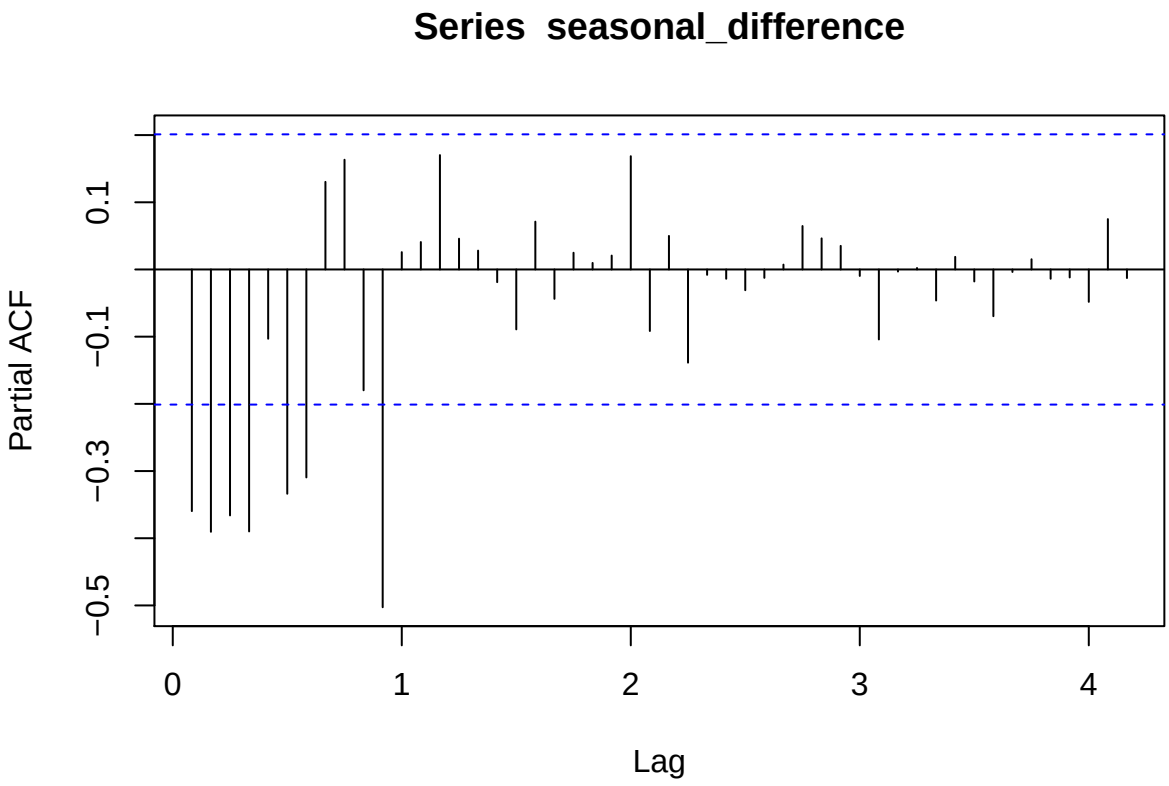

```
# Plot of first and seasonal difference
plot(seasonal_difference,xlab='Time',ylab='First and Seasonal Difference', main = 'Plot of First and Sea
```




```
acf(seasonal_difference, lag.max = 50)
```




```
pacf(seasonal_difference, lag.max = 50)
```



The plot above shows the yearly cyclical rise and fall of people searching for keyword 'Python' in internet. Another plot shows the time series data after accounting for seasonal difference and trend difference. If we look at the ACF plot, we can see that all the values are bouncing around between negative and positive lags suggesting no lags from the seasonal part is significant for our analysis. Similarly, if we inspect PACF plot, none of the lags are significant but it does indeed decays to zero.

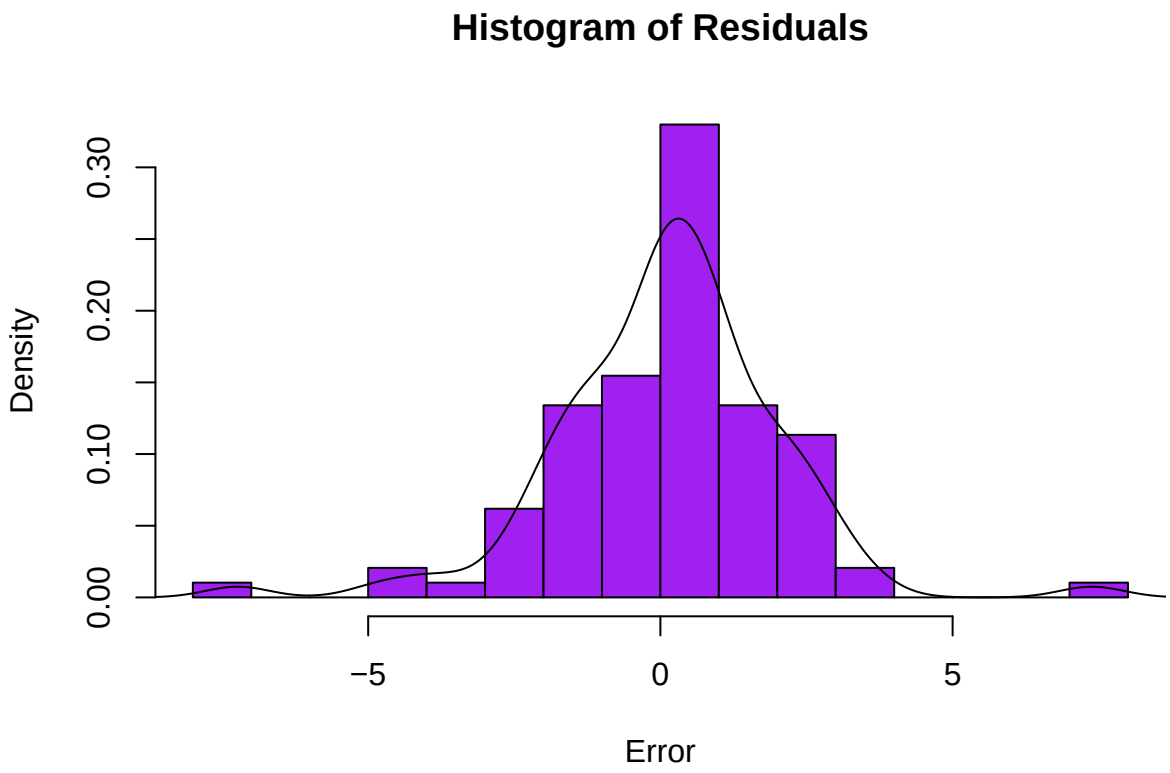
Hence, we can conclude that the best model to fit for this series is $SARIMA(2,1,2)(0,1,0)[12]$ model. Let us verify the model and look at the histogram of the residuals, normal Q-Q plot and p-values to diagnose and inspect the fit of the model.

```
model = auto.arima(Python_Series)
model
```

```
## Series: Python_Series
## ARIMA(2,1,2)(0,1,0)[12]
##
## Coefficients:
##           ar1      ar2     ma1     ma2
##       -1.3734  -0.9424  1.0800  0.6157
## s.e.   0.0705   0.0510  0.1367  0.1076
##
## sigma^2 estimated as 4.376:  log likelihood=-180.07
## AIC=370.15   AICc=370.92   BIC=382.3
```

```
par(mfrow=c(1, 1))
hist(model$residuals,
     col = 'purple',
```

```
     xlab = 'Error',
     main = 'Histogram of Residuals',
     freq = FALSE,
     breaks = 20
     )
lines(density(model$residuals))
```

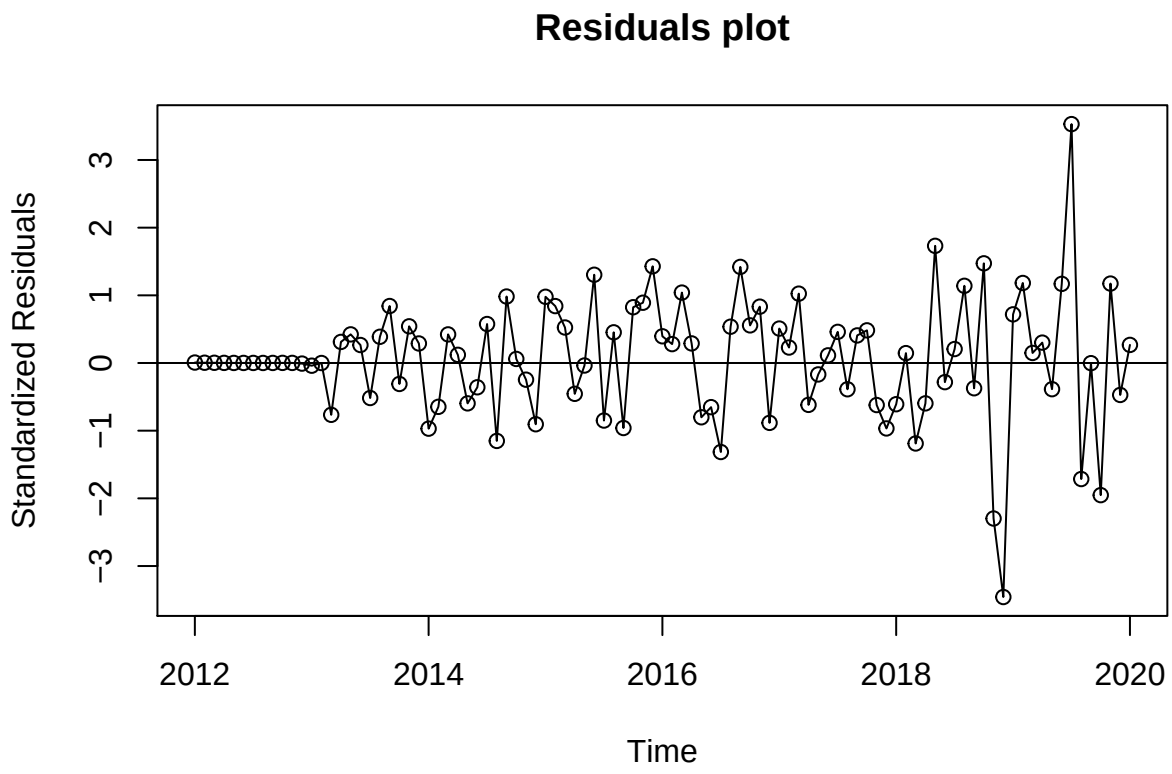


By using auto.arima function in R, we were able to verify our model. Looking at the histogram, our residuals are approximately distributed normally.

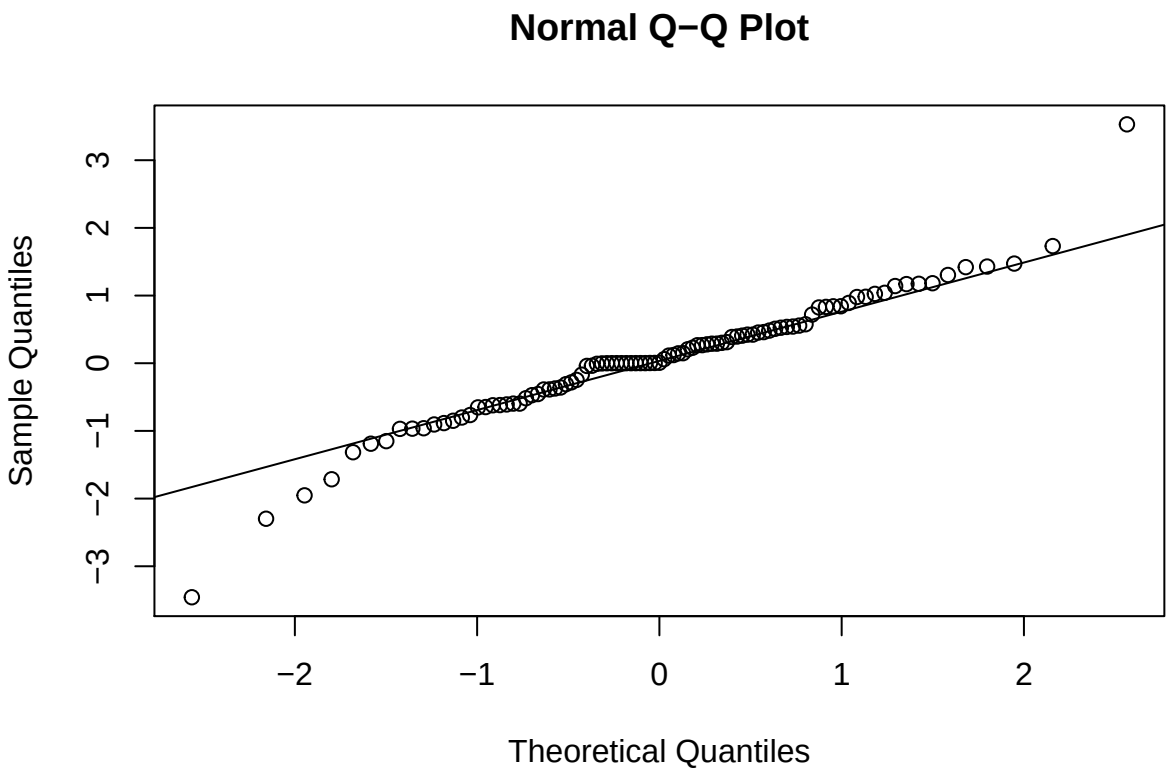
```
Box.test(model$residuals, lag = 50, type = 'Ljung-Box')
```

```
##
##  Box-Ljung test
##
## data:  model$residuals
## X-squared = 72.858, df = 50, p-value = 0.01909
```

```
plot(window(rstandard(model),start=c(2012,1)),ylab="Standardized Residuals", type = 'o', main = "Residua
```



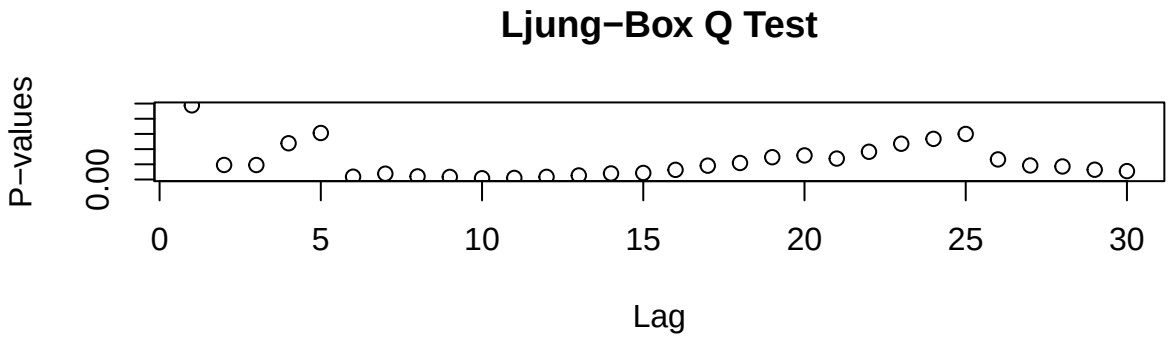

```
qqnorm(window(rstandard(model),start=c(2012,1)));qqline(window(rstandard(model),start=c(2012,1)))
```

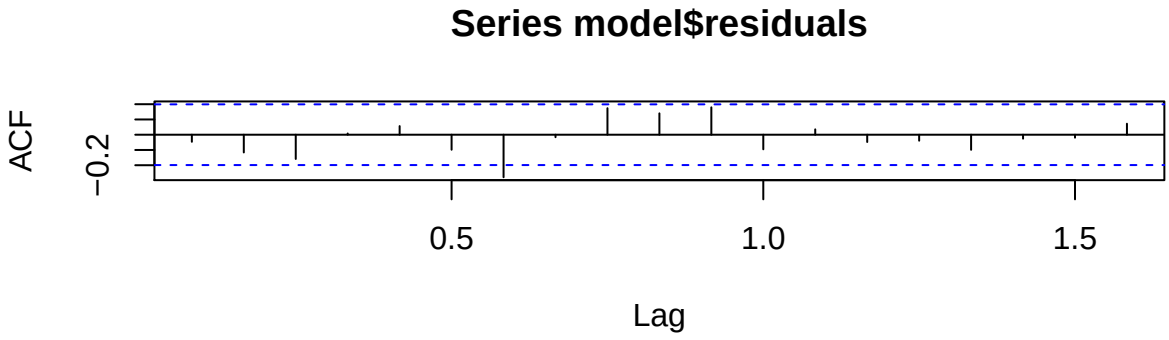



```
boxresult<-LjungBoxTest (model$residuals,k=2,StartLag=1) # residual?? or the original series?
```

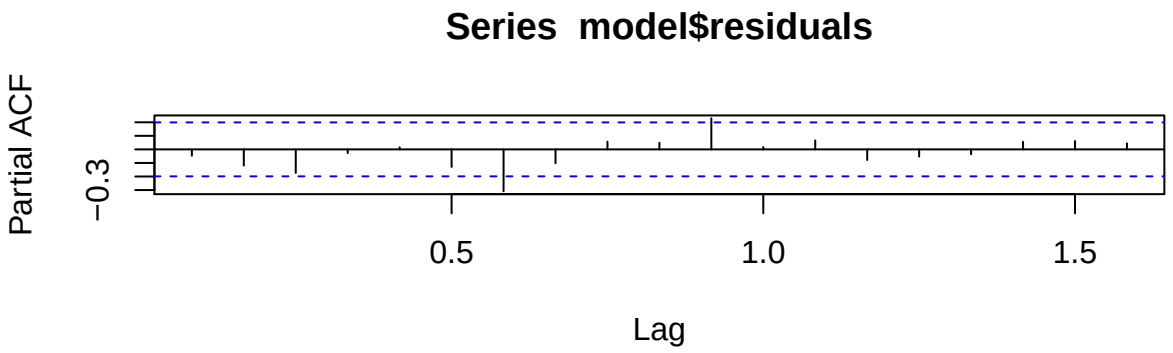
```
## Warning in (ra^2)/(n - (1:lag.max)): longer object length is not a multiple of
## shorter object length
```

```
par(mfrow=c(2,1))
plot(boxresult[,3],main="Ljung-Box Q Test", ylab="P-values", xlab="Lag")
acf(model$residuals)
```





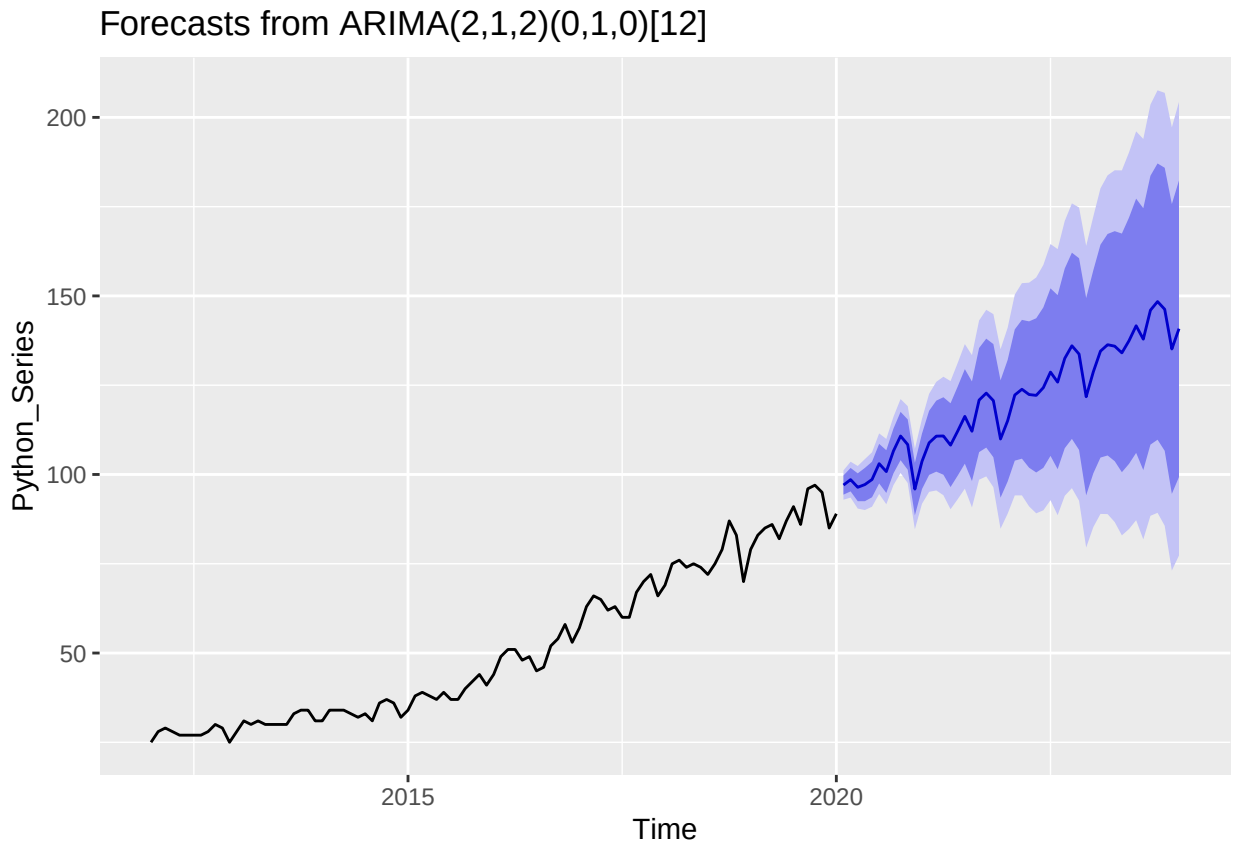

```
pacf(model$residuals)
```



From the residual plot above, we can see that there are some strange behavior near the end but it does not suggest any irregularities in the model. From the diagnostics above, the model does fit fine but there are still some outliers present in the data with unexplained variance which is shown in the Q-Q plot and from the standardized residuals plots

Using the forecast function, we can forecast 48 months ahead for how the search in the internet worldwide for word 'Python' changes with respect to time.

```
f = forecast::forecast(model, 48)
autoplot(f)
```



The darker blue shaded region represents the bound for plus and minus 1 standard deviation away and lighter blue representing bound of plus and minus 2. As time progresses, the error is being carried forward from previous time step which increases the confidence bound of the prediction. As time goes on, uncertainty increases.