France Unemployment predictions

Importing dataset:

```
rm(list=ls())
data_france=read.csv("France_Unemployment.csv")
head(data_france)
```

```
##      TIME Unemployment_Rate Unemployment    CPI Total_new_Job_Vacancies
## 1 1989-01             0.098      2554100 64.215                  101700
## 2 1989-02             0.097      2541300 64.398                  100000
## 3 1989-03             0.097      2533900 64.582                   99500
## 4 1989-04             0.097      2547800 64.986                  106100
## 5 1989-05             0.097      2537300 65.244                  105300
## 6 1989-06             0.096      2530900 65.317                  110600
##   Domestic_Producer_Prices_Index Unemployed_Male Unemployed_Female
## 1                        82.4663         1047000           1365000
## 2                        82.4663         1037000           1360000
## 3                        82.5494         1033000           1360000
## 4                        83.0482         1030000           1359000
## 5                        83.1313         1029000           1357000
## 6                        82.6325         1028000           1355000
```

```
tail(data_france)
```

```
##        TIME Unemployment_Rate Unemployment    CPI Total_new_Job_Vacancies
## 405 2022-09             0.071      2905800 112.74                  323400
## 406 2022-10             0.072      2875100 113.90                  324100
## 407 2022-11             0.071      2810400 114.26                  341800
## 408 2022-12             0.071      2816600 114.16                  324600
## 409 2023-01             0.071      2808500 114.60                  337100
## 410 2023-02             0.070      2780800 115.78                  342300
##     Domestic_Producer_Prices_Index Unemployed_Male Unemployed_Female
## 405                          144.5         1154000           1019000
## 406                          144.2         1165000           1039000
## 407                          145.2         1154000           1033000
## 408                          146.9         1157000           1033000
## 409                          150.5         1154000           1020000
## 410                          149.2         1147000           1005000
```

Feature Scaling - Creating a new variable:

```
data_france$male_to_female_unemp=round((data_france$Unemployed_Male/data_france$Unemployed_Femal
e),4)
print(data_france$male_to_female_unemp[10])
```

```
## [1] 0.7654
```

This is to incorporate the factor - whether female are getting more unemployed or not compared to males over the years - as an external variable for overall unemployment rate.

Checking Multicollinearity using VIFs:

```
suppressWarnings(library(regclass))
```

```
## Loading required package: bestglm
```

```
## Loading required package: leaps
```

```
## Loading required package: VGAM
```

```
## Loading required package: stats4
```

```
## Loading required package: splines
```

```
## Loading required package: rpart
```

```
## Loading required package: randomForest
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
## Important regclass change from 1.3:
## All functions that had a . in the name now have an _
## all.correlations -> all_correlations, cor.demo -> cor_demo, etc.
```

```
VIF(lm(formula = Unemployment_Rate ~ Unemployment+CPI+Total_new_Job_Vacancies+Domestic_Producer_Pr
ices_Index+male_to_female_unemp, data = data_france))
```

```
##                Unemployment                          CPI
##                    2.359338                    22.296290
##     Total_new_Job_Vacancies Domestic_Producer_Prices_Index
##                    3.574387                     6.225157
##        male_to_female_unemp
##                    7.026014
```

Removing CPI:

```
VIF(lm(formula = Unemployment_Rate ~ Unemployment+Total_new_Job_Vacancies+Domestic_Producer_Prices
_Index+male_to_female_unemp, data = data_france))
```

```
##                    Unemployment        Total_new_Job_Vacancies
##                        1.732834                       2.039151
## Domestic_Producer_Prices_Index           male_to_female_unemp
##                        2.880349                       3.636023
```

This is the final set of variables free from multicollinearity.

Train-Test split of the dataset - last 6 months of the data would be taken into testing part:

```
df_france_train1=data_france[1:(nrow(data_france)-6),]
df_france_test1=data_france[(nrow(data_france)-5):nrow(data_france),]
head(df_france_train1)
```

```
##      TIME Unemployment_Rate Unemployment    CPI Total_new_Job_Vacancies
## 1 1989-01             0.098      2554100 64.215                  101700
## 2 1989-02             0.097      2541300 64.398                  100000
## 3 1989-03             0.097      2533900 64.582                   99500
## 4 1989-04             0.097      2547800 64.986                  106100
## 5 1989-05             0.097      2537300 65.244                  105300
## 6 1989-06             0.096      2530900 65.317                  110600
##   Domestic_Producer_Prices_Index Unemployed_Male Unemployed_Female
## 1                        82.4663         1047000           1365000
## 2                        82.4663         1037000           1360000
## 3                        82.5494         1033000           1360000
## 4                        83.0482         1030000           1359000
## 5                        83.1313         1029000           1357000
## 6                        82.6325         1028000           1355000
##   male_to_female_unemp
## 1               0.7670
## 2               0.7625
## 3               0.7596
## 4               0.7579
## 5               0.7583
## 6               0.7587
```

- The model would be trained on the train dataset.
- And the performance of the fitted model would be checked on the test dataset.
- If this performs fairly well, this model would be considered to get the future forecasts.

Time series plot:

```
suppressWarnings(library(fpp2))
```

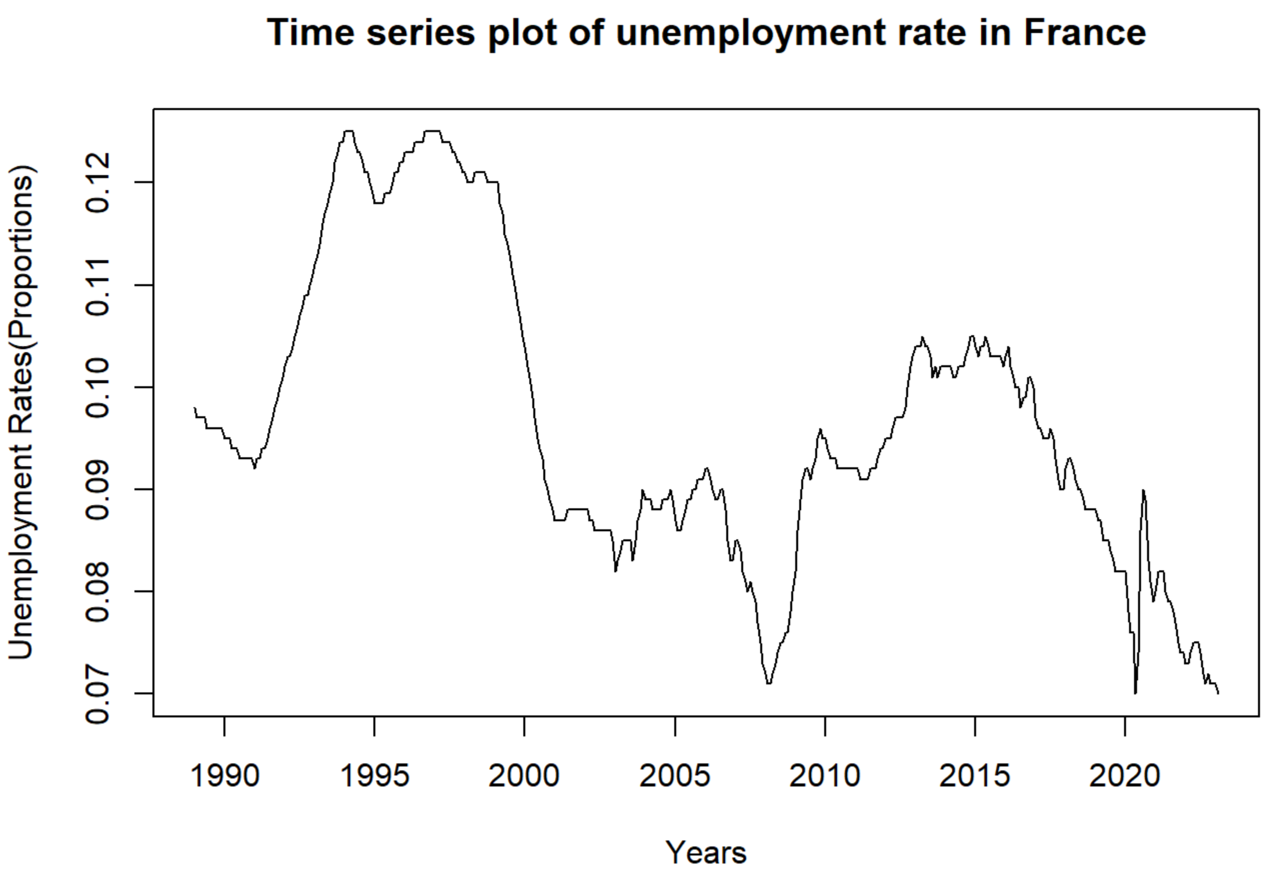
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## ── Attaching packages ────────────────────────────────────────────── fpp2 2.5 ──
```

```
## ✔ ggplot2   3.3.6     ✔ fma       2.4
## ✔ forecast  8.18      ✔ expsmooth 2.3
```

```
## ── Conflicts ────────────────────────────────────────────────── fpp2_conflicts ──
## ✖ ggplot2::margin() masks randomForest::margin()
```

```
suppressWarnings(library(urca))
df.ts=ts(data_france$Unemployment_Rate, frequency = 12, start = c(1989,1))
plot(df.ts,xlab="Years",ylab="Unemployment Rates(Proportions)")
title(main="Time series plot of unemployment rate in France")
```



Testing stationarity:

```
df_france_train1[,"Unemployment_Rate"] %>%
  ur.kpss() %>%
  summary()
```

```
##
## #######################
## # KPSS Unit Root Test #
## #######################
##
## Test is of type: mu with 5 lags.
##
## Value of test-statistic is: 2.3638
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

This series is non-stationary - 1st order differencing would be necessary.

Testing stationarity after 1st order differencing:

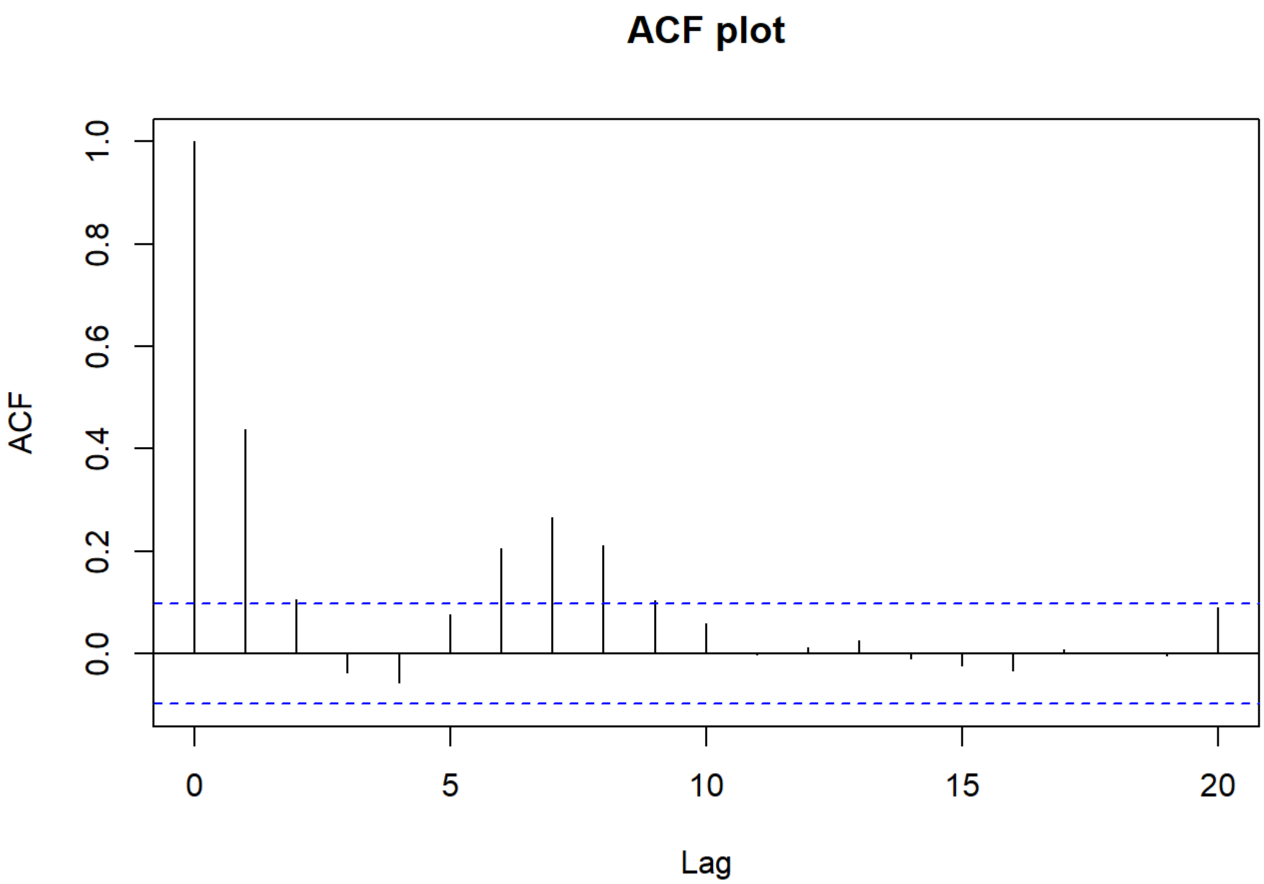
```
diff(df_france_train1[,"Unemployment_Rate"]) %>%
  ur.kpss() %>%
  summary()
```

```
##
## #######################
## # KPSS Unit Root Test #
## #######################
##
## Test is of type: mu with 5 lags.
##
## Value of test-statistic is: 0.2404
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

1st order differences are stationary.

ACF plot:

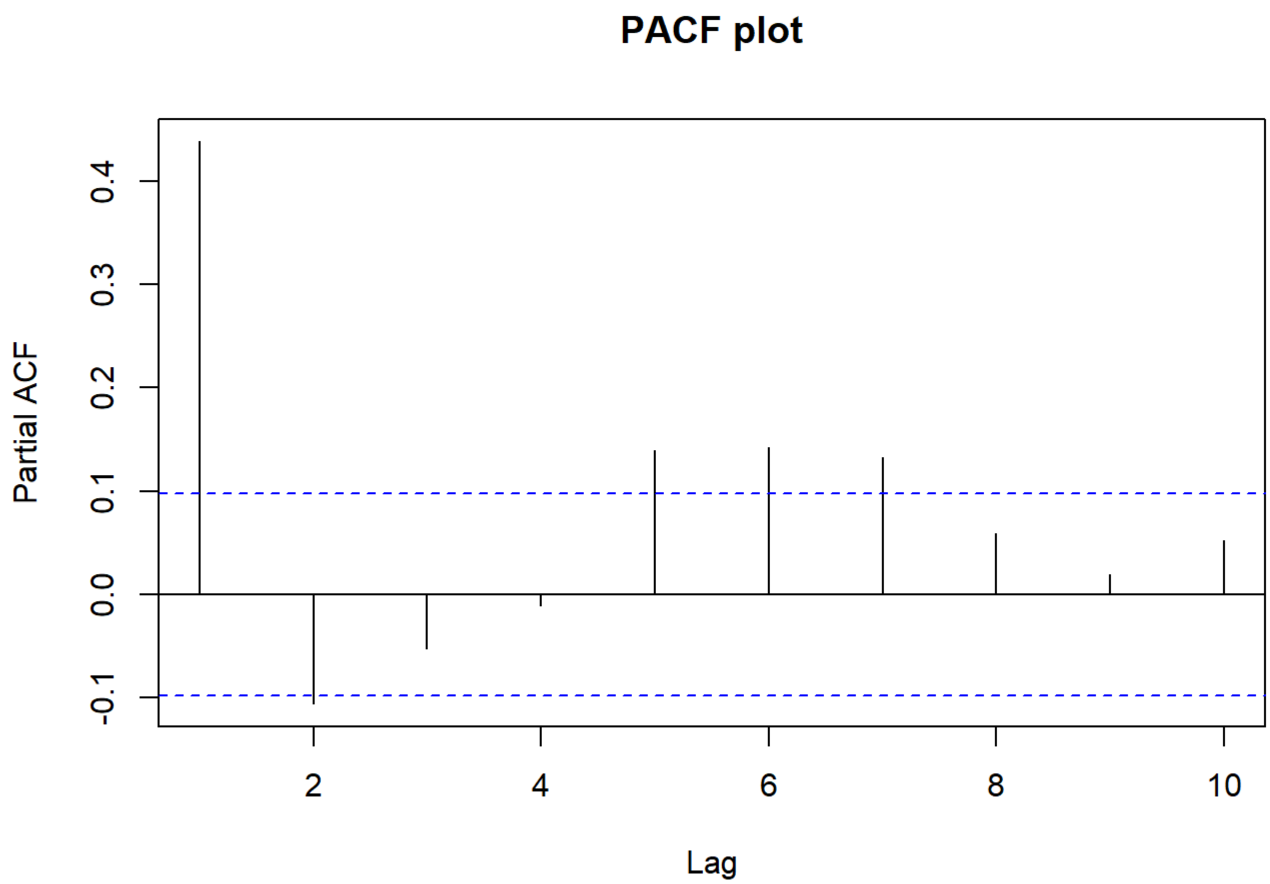
```
par(mfrow=c(1,1))
acf(diff(df_france_train1$Unemployment_Rate), lag.max = 20, main = "ACF plot")
```



p can be taken as 0/1/2 based on the no. of significant lags.

PACF plot:

```
par(mfrow=c(1,1))
pacf(diff(df_france_train1$Unemployment_Rate), lag.max = 10, main = "PACF plot")
```



q can be 0/1/2, based on the no. of significant lags.

Fitting ARIMAX model ignoring the variables that were eliminated due to high VIF:

Starting with the value of p & q as 2 and with the rest of the regressors:

```
est_train=arima(df_france_train1$Unemployment_Rate, order=c(2,1,2), xreg = as.matrix(df_france_tra
in1[,c(3,5,6,9)]), method = "ML")
summary(est_train)
```

```
##
## Call:
## arima(x = df_france_train1$Unemployment_Rate, order = c(2, 1, 2), xreg = as.matrix(df_france_tr
ain1[,
##     c(3, 5, 6, 9)]), method = "ML")
##
## Coefficients:
```

```
## Warning in sqrt(diag(x$var.coef)): NaNs produced
```

```
##          ar1      ar2      ma1     ma2  Unemployment  Total_new_Job_Vacancies
##       0.8795  -0.6445  -0.2222  0.3565             0                        0
## s.e.  0.0959   0.0559   0.1065  0.0705           NaN                      NaN
##       Domestic_Producer_Prices_Index  male_to_female_unemp
##                               0e+00               -0.0093
## s.e.                          1e-04                0.0039
##
## sigma^2 estimated as 1.049e-06:  log likelihood = 2201.9,  aic = -4385.81
##
## Training set error measures:
##                         ME        RMSE          MAE         MPE      MAPE
## Training set -4.424217e-05 0.001022971 0.0006904664 -0.05289185 0.7428873
##                  MASE         ACF1
## Training set 0.875025 -0.008829127
```

Test of significance of individual coefficients:

```
suppressWarnings(library(lmtest))
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

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

```
##
## Attaching package: 'lmtest'
```

```
## The following object is masked from 'package:VGAM':
##
##     lrtest
```

```
coeftest(est_train)
```

```
## Warning in sqrt(diag(se)): NaNs produced
```

```
##
## z test of coefficients:
##
##                                  Estimate  Std. Error  z value  Pr(>|z|)
## ar1                            8.7946e-01  9.5888e-02   9.1717 < 2.2e-16 ***
## ar2                           -6.4447e-01  5.5925e-02 -11.5239 < 2.2e-16 ***
## ma1                           -2.2216e-01  1.0651e-01  -2.0858   0.03700 *
## ma2                            3.5654e-01  7.0496e-02   5.0576 4.245e-07 ***
## Unemployment                   1.1161e-08         NaN      NaN       NaN
## Total_new_Job_Vacancies       -3.3634e-09         NaN      NaN       NaN
## Domestic_Producer_Prices_Index -2.8934e-06  9.9732e-05  -0.0290   0.97686
## male_to_female_unemp          -9.3459e-03  3.9176e-03  -2.3856   0.01705 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We need to remove the variables producing NaNs & the insignificant variables.

After doing that, the summary & test of significances of the final model would look like:

```
est_1=arima(df_france_train1$Unemployment_Rate, order=c(1,1,1), method = "ML")
summary(est_1)
```

```
##
## Call:
## arima(x = df_france_train1$Unemployment_Rate, order = c(1, 1, 1), method = "ML")
##
## Coefficients:
##          ar1     ma1
##       0.2906  0.1905
## s.e.  0.0986  0.0994
##
## sigma^2 estimated as 1.335e-06:  log likelihood = 2153.59,  aic = -4301.19
##
## Training set error measures:
##                         ME        RMSE          MAE         MPE      MAPE
## Training set -3.973705e-05 0.001154181 0.0007430032 -0.05063419 0.8069906
##                   MASE        ACF1
## Training set 0.9416046 0.002304672
```

Test of significance of coefficients:

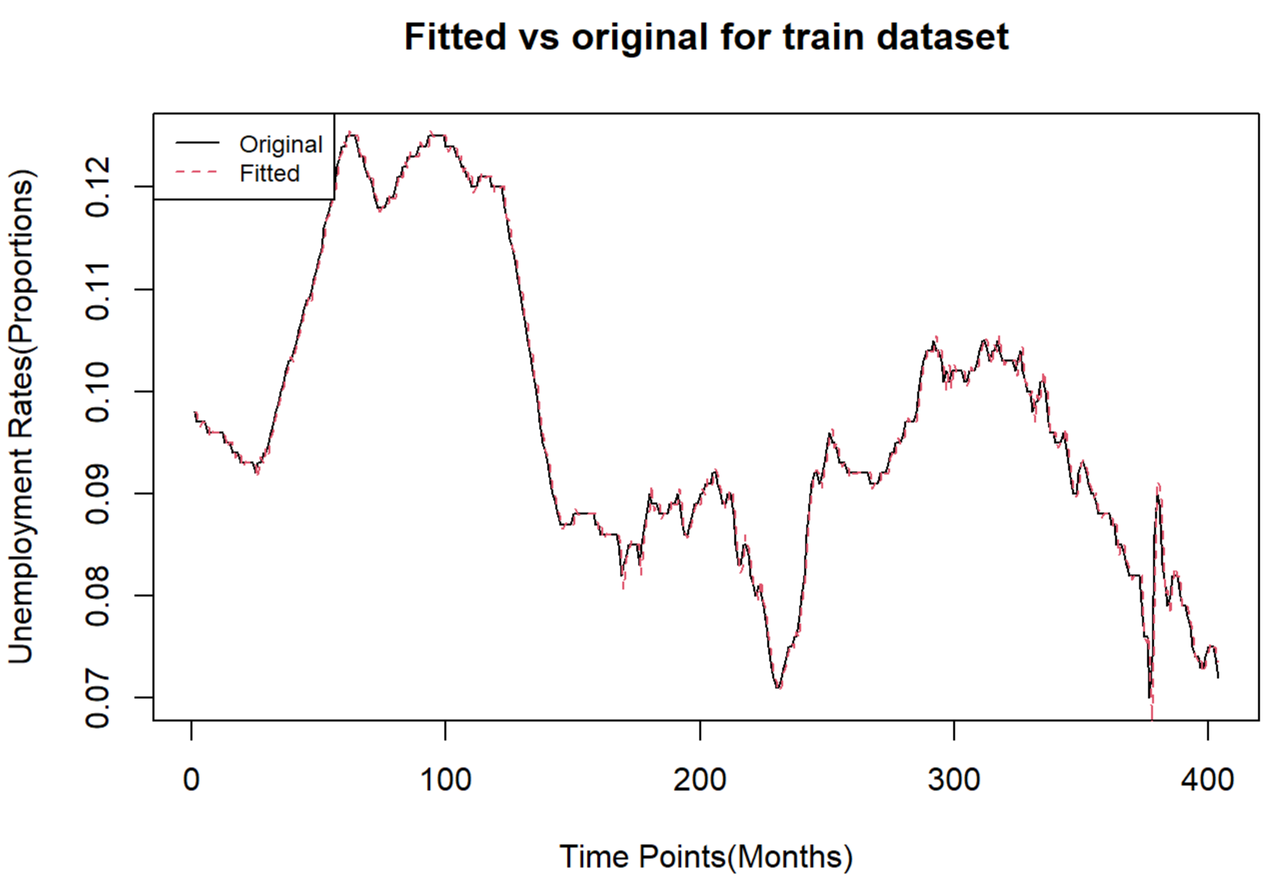
```
suppressWarnings(library(lmtest))
coeftest(est_1)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1 0.290572   0.098581  2.9475 0.003203 **
## ma1 0.190456   0.099390  1.9163 0.055332 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Thus only those parameters are kept as final, which are significant in prediction of the target variable.

Plot of Fitted vs Original values for train dataset:

```
res=residuals(est_1)
data_fit=df_france_train1$Unemployment_Rate-res
ts.plot(df_france_train1$Unemployment_Rate, type="l", xlab="Time Points(Months)", ylab="Unemployme
nt Rates(Proportions)", main="Fitted vs original for train dataset")
points(data_fit, type="l", col=2, lty=2)
legend("topleft",c("Original","Fitted"), col=c(1,2), lty=c(1,2), cex=0.75)
```



Predictions of unemployment rates for the test dataset using above fitted model:

```
test_pred=predict(est_1, n.ahead=6, se.fit=FALSE, method="ML")
```

Predicted values:

```
print(as.vector(test_pred))
```

```
## [1] 0.07113007 0.07087730 0.07080385 0.07078251 0.07077630 0.07077450
```

Original values:

```
print(df_france_test1$Unemployment_Rate)
```

```
## [1] 0.071 0.072 0.071 0.071 0.071 0.070
```

Performance on test dataset:

MAPE (in %):

```
(1/length(df_france_test1$Unemployment_Rate))*(sum(abs(df_france_test1$Unemployment_Rate-as.vector
(test_pred))/abs(df_france_test1$Unemployment_Rate)))*100
```

```
## [1] 0.6244345
```

RMSE:

```
sqrt(mean((df_france_test1$Unemployment_Rate-as.vector(test_pred))^2))
```

```
## [1] 0.000579231
```

Thus, the fitted model is working well, more or less, for future dataset.

Now going with the same approach with the actual dataset for getting the future forecast of March,23:

Checking stationarity:

```
data_france[,"Unemployment_Rate"] %>%
  ur.kpss() %>%
  summary()
```

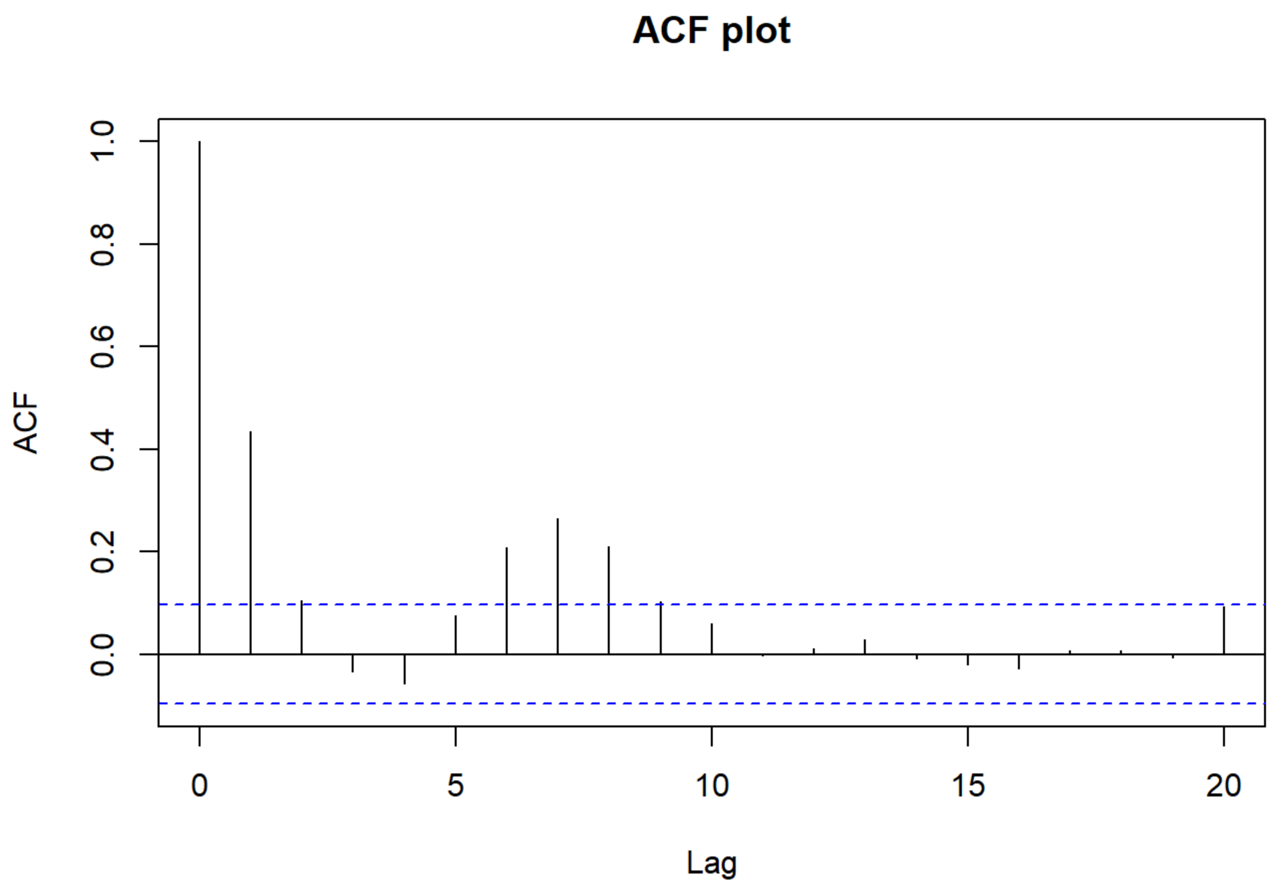
```
##
## #######################
## # KPSS Unit Root Test #
## #######################
##
## Test is of type: mu with 5 lags.
##
## Value of test-statistic is: 2.5039
##
## Critical value for a significance level of:
##                  10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

```
diff(data_france[,"Unemployment_Rate"]) %>%
  ur.kpss() %>%
  summary()
```

```
##
## #######################
## # KPSS Unit Root Test #
## #######################
##
## Test is of type: mu with 5 lags.
##
## Value of test-statistic is: 0.2521
##
## Critical value for a significance level of:
##                  10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

ACF plot:

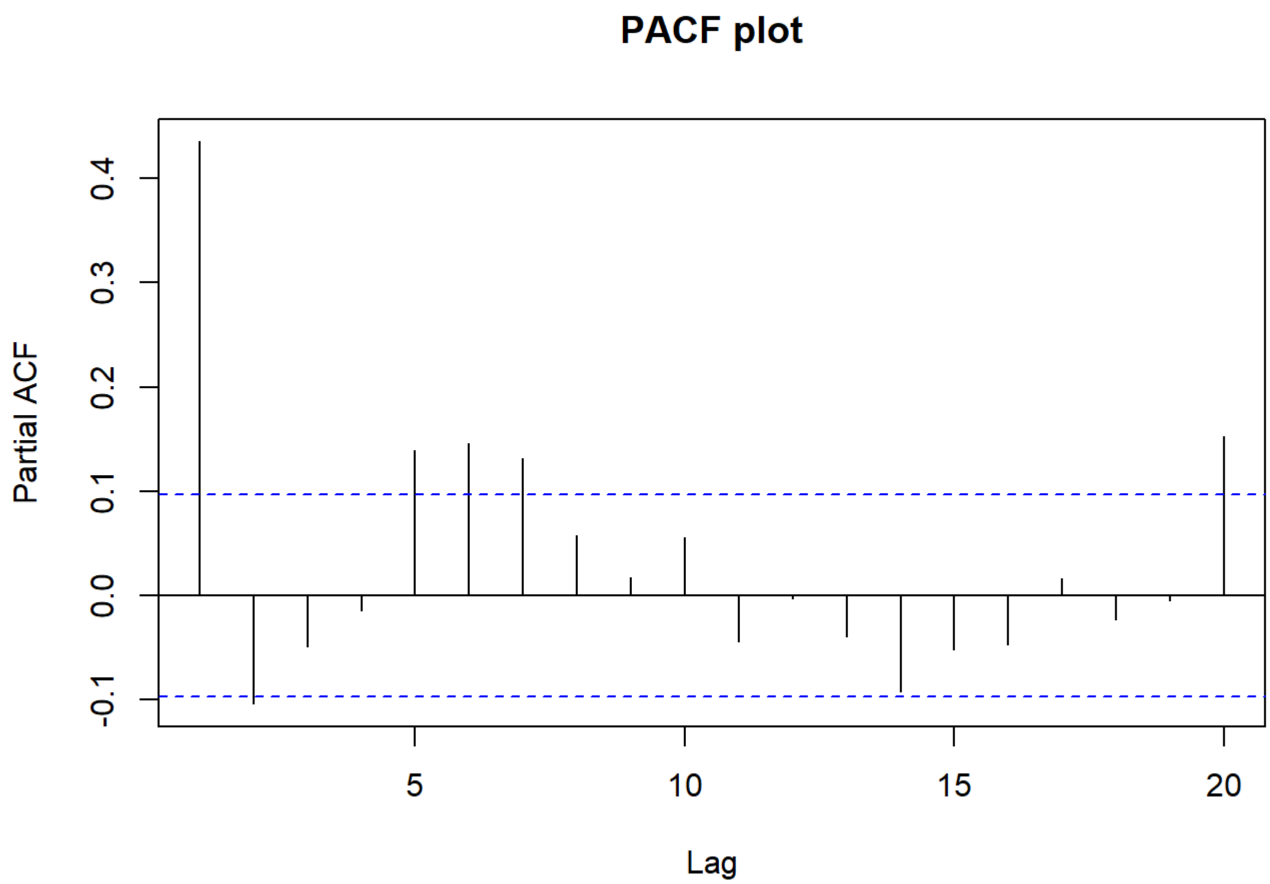
```
par(mfrow=c(1,1))
acf(diff(data_france$Unemployment_Rate), lag.max = 20, main = "ACF plot")
```



p can be taken as 0/1/2, based on the no. of significant lags.

PACF plot:

```
par(mfrow=c(1,1))
pacf(diff(data_france$Unemployment_Rate), lag.max = 20, main = "PACF plot")
```




```
#q=0/1
```

q can be taken as 0/1, based on the no. of significant lags.

Fitting the model that we tested before - on the actual data:

```
est_actual=arima(data_france$Unemployment_Rate, order=c(1,1,1), method = "ML")
summary(est_actual)
```

```
##
## Call:
## arima(x = data_france$Unemployment_Rate, order = c(1, 1, 1), method = "ML")
##
## Coefficients:
##          ar1     ma1
##       0.2849  0.1913
## s.e.  0.0994  0.1004
##
## sigma^2 estimated as 1.329e-06:  log likelihood = 2186.63,  aic = -4367.27
##
## Training set error measures:
##                         ME        RMSE          MAE         MPE      MAPE
## Training set -4.15456e-05 0.001151466 0.000743623 -0.05339375 0.8113157
##                   MASE        ACF1
## Training set 0.9445397 0.001563919
```

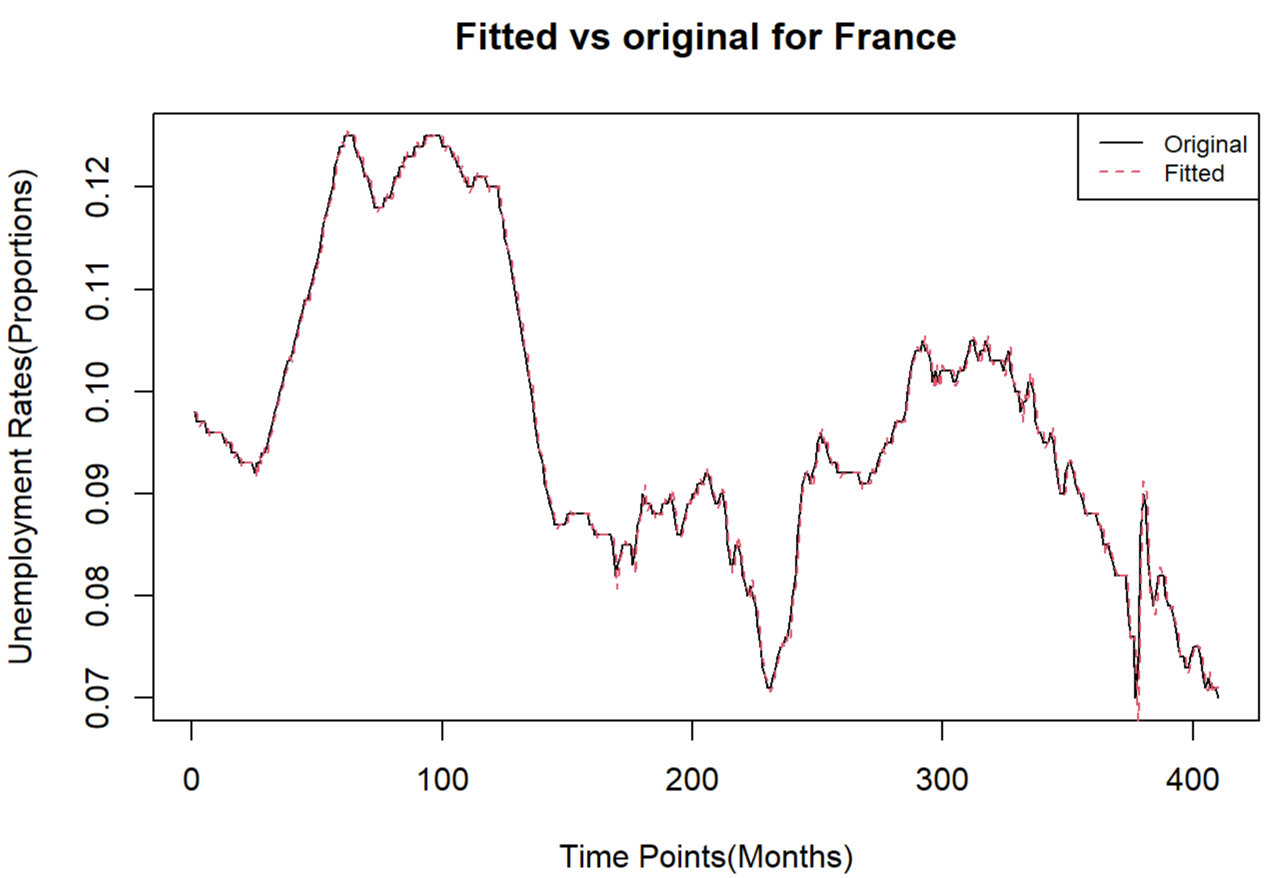
Test of significance of individual coefficients:

```
suppressWarnings(library(lmtest))
coeftest(est_actual)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1 0.284904   0.099398  2.8663 0.004153 **
## ma1 0.191338   0.100386  1.9060 0.056646 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Plot of Fitted vs Original on the actual data:

```
res=residuals(est_actual)
data_fit=data_france$Unemployment_Rate-res
ts.plot(data_france$Unemployment_Rate, type="l", xlab="Time Points(Months)", ylab="Unemployment Ra
tes(Proportions)", main="Fitted vs original for France")
points(data_fit, type="l", col=2, lty=2)
legend("topright",c("Original","Fitted"), col=c(1,2), lty=c(1,2), cex=0.75)
```



Obtaining prediction of Unemployment rate for April 2023:

```
#Adding actual unemployment rate of March 23:
unemp_fr=c(data_france$Unemployment_Rate,0.069)
est_actual=arima(unemp_fr, order=c(1,1,1), method = "ML")

future_unemp_pred=predict(est_actual, n.ahead=1, se.fit=FALSE, method="ML")
print(as.vector(future_unemp_pred))
```

```
## [1] 0.06861347
```

Upper & Lower limits (95% C.I.s):

```
upper=as.vector(future_unemp_pred)+(1.96*(sqrt(est_actual$sigma2)))
lower=as.vector(future_unemp_pred)-(1.96*(sqrt(est_actual$sigma2)))
```

Upper limit for April 2023 forecast:

```
print(as.vector(upper))
```

```
## [1] 0.0708709
```

Lower limit for April 2023 forecast:

```
print(as.vector(lower))
```

```
## [1] 0.06635605
```

April,23 forecast - 6.8613 %

Upper & Lower limits - (6.6356 %, 7.087 %)