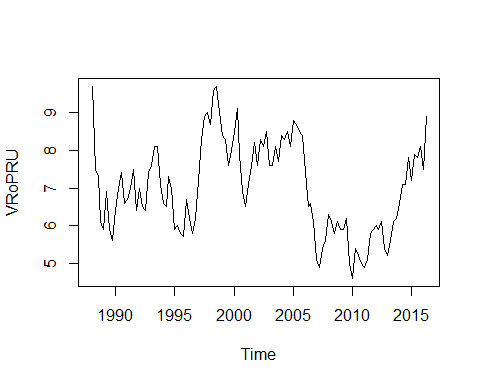
Optional Assignment

Leanne Tan Jing Yi A0130872J D01

8 September 2016

This project was done using R because I feel that it is the easiest to display the method to replicate my results.This document was generated using R Markdown.

#CSV file was downloaded from REALIS.   
  
 df <- read.csv("REALIS.csv")  
 #transformation of dataframe  
 df <- df[ 1:114,]  
 df$Vacancy.Rate.of.Private.Residential.Units <-  
 as.numeric(as.character(df$Vacancy.Rate.of.Private.Residential.Units))  
   
 VRoPRU <- ts(df$Vacancy.Rate.of.Private.Residential.Units,   
 start=c(1988, 1), end=c(2016, 2), frequency=4)   
 plot(VRoPRU)



As the data appears to be not stationary, a unit root test is required. Before doing the Augmented-Dickey-Fuller Unit Root Test, I selected the best number of lag based on AIC value.

#number of lags  
 library(vars)

## Loading required package: MASS

## Loading required package: strucchange

## Loading required package: zoo

##   
## Attaching package: 'zoo'

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

## Loading required package: sandwich

## Loading required package: urca

## Loading required package: lmtest

var <- VARselect(df$Vacancy.Rate.of.Private.Residential.Units,  
 type = "none",  
 lag.max = 50)  
 var

## $selection  
## AIC(n) HQ(n) SC(n) FPE(n)   
## 1 1 1 1   
##   
## $criteria  
## 1 2 3 4 5 6  
## AIC(n) -1.1656260 -1.1392028 -1.1081439 -1.0967759 -1.0750687 -1.0451727  
## HQ(n) -1.1523370 -1.1126249 -1.0682770 -1.0436201 -1.0086240 -0.9654390  
## SC(n) -1.1318934 -1.0717377 -1.0069462 -0.9618458 -0.9064060 -0.8427774  
## FPE(n) 0.3117283 0.3200806 0.3301939 0.3340005 0.3413832 0.3518253  
## 7 8 9 10 11 12  
## AIC(n) -1.0258872 -1.0021606 -0.9804723 -0.9601193 -0.9397802 -0.9093272  
## HQ(n) -0.9328645 -0.8958490 -0.8608717 -0.8272297 -0.7936017 -0.7498597  
## SC(n) -0.7897594 -0.7323002 -0.6768793 -0.6227938 -0.5687222 -0.5045366  
## FPE(n) 0.3587934 0.3675683 0.3758384 0.3838366 0.3920625 0.4046076  
## 13 14 15 16 17 18  
## AIC(n) -0.8865833 -0.8772592 -0.8766710 -0.8534916 -0.8294472 -0.7997569  
## HQ(n) -0.7138269 -0.6912138 -0.6773367 -0.6408683 -0.6035350 -0.5605557  
## SC(n) -0.4480602 -0.4050035 -0.3706828 -0.3137708 -0.2559939 -0.1925710  
## FPE(n) 0.4144290 0.4189208 0.4198770 0.4305611 0.4420220 0.4564943  
## 19 20 21 22 23  
## AIC(n) -0.7867511 -0.77389505 -0.7426546 -0.71204485 -0.68874869  
## HQ(n) -0.5342610 -0.50811595 -0.4635866 -0.41968785 -0.38310274  
## SC(n) -0.1458327 -0.09924408 -0.0342711 0.03007121 0.08709991  
## FPE(n) 0.4637902 0.47129650 0.4879934 0.50516651 0.51935759  
## 24 25 26 27 28 29  
## AIC(n) -0.6722454 -0.6426879 -0.6128175 -0.5815721 -0.5865167 -0.5608926  
## HQ(n) -0.3533105 -0.3104641 -0.2673046 -0.2227703 -0.2144259 -0.1755129  
## SC(n) 0.1373357 0.2006258 0.2642288 0.3292067 0.3579947 0.4173513  
## FPE(n) 0.5305781 0.5494359 0.5694445 0.5913303 0.5925933 0.6126931  
## 30 31 32 33 34  
## AIC(n) -0.5377703 -0.5200053 -0.51748710 -0.49419627 -0.46299053  
## HQ(n) -0.1391017 -0.1080477 -0.09224055 -0.05566077 -0.01116607  
## SC(n) 0.4742061 0.5257037 0.56195444 0.61897782 0.68391611  
## FPE(n) 0.6323349 0.6496081 0.65778654 0.68063949 0.71054683  
## 35 36 37 38 39  
## AIC(n) -0.49815763 -0.46807282 -0.44220896 -0.43867715 -0.48708391  
## HQ(n) -0.03304422 0.01032955 0.04948236 0.06630312 0.03118533  
## SC(n) 0.68248155 0.74629892 0.80589532 0.84315968 0.82848547  
## FPE(n) 0.69483309 0.72607124 0.75640729 0.77159083 0.74827798  
## 40 41 42 43 44  
## AIC(n) -0.45620168 -0.49676471 -0.46789174 -0.48431830 -0.4724927  
## HQ(n) 0.07535651 0.04808243 0.09024436 0.08710675 0.1122213  
## SC(n) 0.89310025 0.88626977 0.94887529 0.96618127 1.0117394  
## FPE(n) 0.78673437 0.77139974 0.81221102 0.81892571 0.8512123  
## 45 46 47 48 49 50  
## AIC(n) -0.50690263 -0.5404566 -0.7297495 -0.7076964 -0.7930267 -0.8516954  
## HQ(n) 0.09110033 0.0708353 -0.1051686 -0.0698266 -0.1418679 -0.1872476  
## SC(n) 1.01106204 1.0112406 0.8556803 0.9114659 0.8598682 0.8349320  
## FPE(n) 0.84683981 0.8454823 0.7245658 0.7696755 0.7371683 0.7282913

var$selection

## AIC(n) HQ(n) SC(n) FPE(n)   
## 1 1 1 1

Selection returns best number of lag as 1.

#test for stationary  
 library(urca)  
 urdf <- ur.df( y= df$Vacancy.Rate.of.Private.Residential.Units, type = "none", lags = 1, selectlags = "AIC")  
 summary(urdf)

##   
## ###############################################   
## # Augmented Dickey-Fuller Test Unit Root Test #   
## ###############################################   
##   
## Test regression none   
##   
##   
## Call:  
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.19366 -0.38254 0.00867 0.50645 1.41538   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)  
## z.lag.1 -0.001410 0.007931 -0.178 0.859  
## z.diff.lag 0.008000 0.091969 0.087 0.931  
##   
## Residual standard error: 0.593 on 110 degrees of freedom  
## Multiple R-squared: 0.000355, Adjusted R-squared: -0.01782   
## F-statistic: 0.01953 on 2 and 110 DF, p-value: 0.9807  
##   
##   
## Value of test-statistic is: -0.1778   
##   
## Critical values for test statistics:   
## 1pct 5pct 10pct  
## tau1 -2.58 -1.95 -1.62

Since the null hypothesis is not rejected, the process has a unit root. Thus differencing is required.

library(forecast)

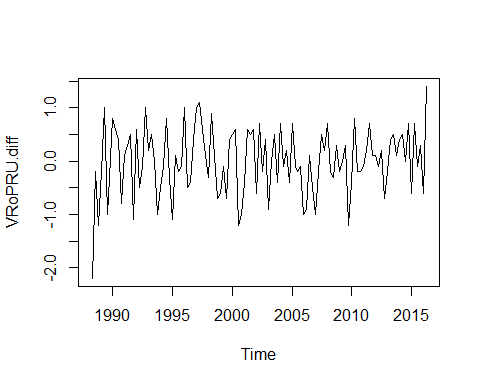
## Loading required package: timeDate

## This is forecast 7.2

#differencing  
 #ndiffs estimate the number of differences required to make a given time series stationary  
 diffs <- ndiffs(VRoPRU, test = 'adf')  
 diffs

## [1] 1

#plot to see the differencing   
 VRoPRU.diff <- diff(VRoPRU, differences = 1)  
 plot(VRoPRU.diff)



To validate my analaysis so far, I used the auto.arima function so that it can return the best ARIMA model according to the lowest AIC value (see ?auto.arima in R)

# ARIMA model

#Using Automated forecasting  
 library(forecast)  
 library(urca)  
   
 # Automated forecasting using an ARIMA model  
   
 fitted <- auto.arima(VRoPRU,  
 stepwise=FALSE,   
 approximation=FALSE,   
 seasonal = FALSE,  
 ic = c("aic"),  
 test = c("adf"),  
 trace = TRUE)

##   
## ARIMA(0,1,0) : 214.9111  
## ARIMA(0,1,0) with drift : 216.8964  
## ARIMA(0,1,1) : 216.903  
## ARIMA(0,1,1) with drift : 218.8883  
## ARIMA(0,1,2) : 218.8074  
## ARIMA(0,1,2) with drift : 220.7937  
## ARIMA(0,1,3) : 220.4358  
## ARIMA(0,1,3) with drift : 222.4255  
## ARIMA(0,1,4) : 222.207  
## ARIMA(0,1,4) with drift : 224.2015  
## ARIMA(0,1,5) : 224.0299  
## ARIMA(0,1,5) with drift : 226.0222  
## ARIMA(1,1,0) : 216.9035  
## ARIMA(1,1,0) with drift : 218.8888  
## ARIMA(1,1,1) : Inf  
## ARIMA(1,1,1) with drift : Inf  
## ARIMA(1,1,2) : Inf  
## ARIMA(1,1,2) with drift : Inf  
## ARIMA(1,1,3) : Inf  
## ARIMA(1,1,3) with drift : Inf  
## ARIMA(1,1,4) : 224.1725  
## ARIMA(1,1,4) with drift : 226.3416  
## ARIMA(2,1,0) : 218.805  
## ARIMA(2,1,0) with drift : 220.7912  
## ARIMA(2,1,1) : Inf  
## ARIMA(2,1,1) with drift : Inf  
## ARIMA(2,1,2) : Inf  
## ARIMA(2,1,2) with drift : Inf  
## ARIMA(2,1,3) : Inf  
## ARIMA(2,1,3) with drift : Inf  
## ARIMA(3,1,0) : 220.3432  
## ARIMA(3,1,0) with drift : 222.3333  
## ARIMA(3,1,1) : 222.33  
## ARIMA(3,1,1) with drift : 224.3204  
## ARIMA(3,1,2) : Inf  
## ARIMA(3,1,2) with drift : Inf  
## ARIMA(4,1,0) : 222.3023  
## ARIMA(4,1,0) with drift : 224.2936  
## ARIMA(4,1,1) : 224.2589  
## ARIMA(4,1,1) with drift : 226.2507  
## ARIMA(5,1,0) : 223.8612  
## ARIMA(5,1,0) with drift : 225.8515

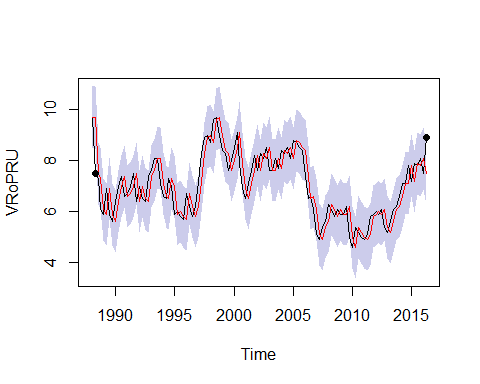
summary(fitted)

## Series: VRoPRU   
## ARIMA(0,1,0)   
##   
## sigma^2 estimated as 0.3853: log likelihood=-106.46  
## AIC=214.91 AICc=214.95 BIC=217.64  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set -0.006932456 0.6180054 0.4930675 -0.468218 7.162367 0.5632132  
## ACF1  
## Training set 0.006738558

forecast(fitted, h = 1)

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 2016 Q3 8.9 8.104498 9.695502 7.683384 10.11662

#Graph with fitted values as well as prediction intervals of 95%  
 upper <- fitted(fitted) + 1.96\*sqrt(fitted$sigma2)  
 lower <- fitted(fitted) - 1.96\*sqrt(fitted$sigma2)  
 plot(VRoPRU, type="n", ylim=range(lower,upper))  
 polygon(c(time(VRoPRU),rev(time(VRoPRU))), c(upper,rev(lower)),   
 col=rgb(0,0,0.6,0.2), border=FALSE)  
 lines(VRoPRU)  
 lines(fitted(fitted),col='red')  
 out <- (VRoPRU < lower | VRoPRU > upper)  
 points(time(VRoPRU)[out], VRoPRU[out], pch=19)



arimaorder(fitted)

## [1] 0 1 0

A nonseasonal ARIMA model is classified as an "ARIMA(p,d,q)" model, where:

p is the number of autoregressive terms, d is the number of nonseasonal differences needed for stationarity, and q is the number of lagged forecast errors in the prediction equation.

In this case, the ARIMA model returned was ARIMA(0,1,0), which is the random walk model. The value of d = 1 coincide with the previous test on the number of differencing required to make the time series stationary.

# Random walk model

The random walk model is very widely used for non-stationary data, particularly finance and economic data. Random walks also typically have long periods of apparent trends up or down and sudden, unpredictable changes in direction. ARIMA's forecasted values are shown as below.

forecast(fitted, h = 1)

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 2016 Q3 8.9 8.104498 9.695502 7.683384 10.11662

# References

## [1] OTexts. \_8.1 Stationarity and differencing | OTexts\_. 2016.  
## <URL: https://www.otexts.org/fpp/8/1>.  
##   
## [2] Rob Hyndman. \_Fitted Confidence Intervals Forecast Function R  
## - Cross Validated\_. 2015. <URL:  
## http://stats.stackexchange.com/questions/154346/fitted-confidence-intervals-forecast-function-r>.