ECON90033 – QUANTITATIVE ANALYSIS OF FINANCE I

TUTORIAL 7

Download the t7e3 Excel data file from the subject website and save them to your computer or USB flash drive. Read this handout and complete the tutorial exercises before your tutorial class so that you can ask for help during the tutorial if necessary.

Introduction to Forecasting

Economic forecasting is the process of attempting to predict the future value(s) of some element of economic activity. On the basis of the information used in forecasting, forecasts can be qualitative and quantitative. Qualitative forecasts are based on subjective assessments and thus they are neither reproducible nor testable, while quantitative forecasts are developed from some historical data and econometric model in a reproducible and testable way. In this course we focus on quantitative forecasting.

Forecasts are classified in various ways. Suppose we have observations on variable $\{y_t\}$ for times $t = 1, 2,\ldots, T$ and based on them develop forecasts for the sample period, and also for 1, 2, .., h periods ahead, i.e., for times $t = T + 1, T + 2,\ldots, T + h$. The forecasts prepared in time T for the sample period, i.e., $f_{T,1}, f_{T,2},\ldots, f_{T,T}$, are called ex post (after the event) forecasts, and the ones prepared for some periods ahead, i.e., $f_{T,T+1}, f_{T,\,T+2},\ldots, f_{T,T+h}$, are called ex ante (before the event) forecasts. In practice, the ultimate aim of forecasting is to develop ex ante forecasts, the ex post forecasts are used for model evaluation.

The evaluation of forecasting models is based on the comparison of forecast values to the corresponding true or observed values. Clearly, ex ante forecasts cannot be used for model evaluation since the corresponding true values are not available, we have to rely on ex post forecasts. However, the comparison of ex post forecasts to observations that were actually used in their development might be misleading because models that produce accurate ex post forecasts do not necessarily produce accurate ex ante forecasts as well.

For this reason, it is recommended to withhold the most recent observations for evaluation, develop ex post forecasts from the rest of the sample, and compare the unused observations to the corresponding forecasts. For example, one can forecast $\{y_t\}$ for times $t = 1, 2,\ldots, T$ from the first $t^* < T$ observations only and to check how accurate the $\{f_{T,t^*+1}, f_{T,t^*+2},\ldots, f_{T,T}\}$ 'out-of-sample' forecasts are by comparing them to the unused $\{y_{t^*+1}, y_{t^*+2},\ldots, y_T\}$ observations.

There are various measures of forecast accuracy and on the week 5 lectures we discussed six of them: mean error (*ME*), mean absolute error (*MAE*), root mean squared error (*RMSE*), mean percent error (*MPE*), mean absolute percent error (*MAPE*) and mean absolute scaled error (*MASE*).

They are all based on forecast errors, e_t, or on percent errors, p_t,

$$e_t = y_t - f_t \quad , \quad p_t = \frac{e_t}{y_t} = \frac{y_t - f_t}{y_t} \quad (\times 100\%)$$

and are defined as

$$ME = \frac{1}{T - t^*} \sum_{t=t^*+1}^{T} e_t$$

$$MAE = \frac{1}{T - t^*} \sum_{t=t^*+1}^{T} | e_t |$$

$$RMSE = \sqrt{\frac{1}{T - t^*} \sum_{t=t^*+1}^{T} e_t^2}$$

$$MPE = \frac{1}{T - t^*} \sum_{t=t^*+1}^{T} p_t$$

$$MAPE = \frac{1}{T - t^*} \sum_{t=t^*+1}^{T} | p_t |$$

$$MASE = \frac{1}{T - t^*} \sum_{t=t^*+1}^{T} \frac{| e_t |}{MAE_{NF}}$$

In the last formula

$$MAE_{NF} = \frac{1}{t^*} \sum_{t=1}^{t^*} | y_t - y_{t-1} |$$

is the mean absolute error of the naive forecast, $f_{t+1} = y_t$.

This process will be illustrated by Exercise 2. Before that, let's discuss how to forecast with *ARMA* models.

Forecasting with Stationary *ARMA* Models

Forecasting with stationary *ARMA* models is based on a two-step recursive substitution. Namely, if we want to prepare a forecast in time T for $h > 0$ periods ahead, in the first step we have to write out or project the alleged process for time $T + h$, and in the second step we have to replace the unknown population parameters by their estimates, the historical expectations by their realisations, and all expectations of future innovations by their unconditional expected value, which is zero. This process is illustrated in the first exercise.

Exercise 1

Suppose that you have data on variable y_t for t = 1, 2, …, 50, estimate an AR(2) model and get the following sample regression equation[1]:

$$\hat{y}_t = \hat{\varphi}_0 + \hat{\varphi}_1 y_{t-1} + \hat{\varphi}_2 y_{t-2} = 0.045 + 1.182 y_{t-1} - 0.219 y_{t-2}$$

Obtain forecasts for one, two and three periods ahead (i.e., for t = 51, 52, 53), given that the last two available observations are y_{49} = 1.63 and y_{50} = 1.72.

To forecast y in t = 50 for one period ahead, we project the population regression equation,

$$y_t = \varphi_0 + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \varepsilon_t$$

one period ahead,

$$y_{t+1} = \varphi_0 + \varphi_1 y_t + \varphi_2 y_{t-1} + \varepsilon_{t+1} \quad \rightarrow \quad y_{51} = \varphi_0 + \varphi_1 y_{50} + \varphi_2 y_{49} + \varepsilon_{51}$$

The corresponding optimal forecast is the conditional expected value of y_{51},

$$E(y_{t+1} | \Omega_t) = \varphi_0 + \varphi_1 E(y_t | \Omega_t) + \varphi_2 E(y_{t-1} | \Omega_t) + E(\varepsilon_{t+1} | \Omega_t)$$
$$\rightarrow \quad E(y_{51} | \Omega_{50}) = \varphi_0 + \varphi_1 E(y_{50} | \Omega_{50}) + \varphi_2 E(y_{49} | \Omega_{50}) + E(\varepsilon_{51} | \Omega_{50})$$

where the information set, Ω_t, is supposed to contain the realizations of y_t and ε_t up to and including time t,

$$\Omega_t = \{y_1, y_2, ..., y_t; \varepsilon_1, \varepsilon_2, ..., \varepsilon_t\} \quad \rightarrow \quad \Omega_{50} = \{y_1, y_2, ..., y_{50}; \varepsilon_1, \varepsilon_2, ..., \varepsilon_{50}\}$$

Since y_{49} and y_{50} are in the information set, i.e., they are known, their conditional expected values are themselves,

$$E(y_{49} | \Omega_{50}) = y_{49} \quad , \quad E(y_{50} | \Omega_{50}) = y_{50}$$

ε_{51}, however, is not in the information set, so its conditional expected value is the same as its unconditional expected value, i.e.,

$$E(\varepsilon_{51} | \Omega_{50}) = E(\varepsilon_{51}) = 0$$

Accordingly, our forecast for one period ahead is

$$E(y_{51} | \Omega_{50}) = \varphi_0 + \varphi_1 y_{50} + \varphi_2 y_{49}$$

So far, we have tacitly assumed that the φ_0, φ_1, φ_2 population parameters are known. In reality, however, they are unknown, so to make this forecasting procedure operational, we replace φ_0, φ_1, φ_2 with their estimates. Hence,

[1] It can be checked by calculating the characteristic roots that the process represented by this AR(2) model is stationary.

$$\begin{aligned}E(y_{51}\,|\,\Omega_{50}) &= \varphi_0+\varphi_1 y_{50}+\varphi_2 y_{49}\\ &\approx \hat{\varphi}_0+\hat{\varphi}_1 y_{50}+\hat{\varphi}_2 y_{49} = 0.045+1.182y_{50}-0.219y_{49}\\ &= 0.045+1.182\times 1.72-0.219\times 1.63 = 1.721\end{aligned}$$

Likewise, to forecast *y* in *t* = 50 for two periods ahead, we project the population regression equation to *t* = 52,

$$y_{52}=\varphi_0+\varphi_1 y_{51}+\varphi_2 y_{50}+\varepsilon_{52} \quad\rightarrow\quad E(y_{52}\,|\,\Omega_{50})=\varphi_0+\varphi_1 E(y_{51}\,|\,\Omega_{50})+\varphi_2 E(y_{50}\,|\,\Omega_{50})+E(\varepsilon_{52}\,|\,\Omega_{50})$$

y_{50} is in the information set but y_{51} is not. Hence, we use the observed value for the former and the previous forecast for the latter, i.e.,

$$E(y_{50}\,|\,\Omega_{50})=y_{50} \quad,\quad E(y_{51}\,|\,\Omega_{50})=1.721$$

Replacing the unknown population parameters with their estimates again,

$$\begin{aligned}E(y_{52}\,|\,\Omega_{50}) &= \varphi_0+\varphi_1 E(y_{51}\,|\,\Omega_{50})+\varphi_2 y_{50}\\ &\approx \hat{\varphi}_0+\hat{\varphi}_1 E(y_{51}\,|\,\Omega_{50})+\hat{\varphi}_2 y_{50}\\ &= 0.045+1.182\times 1.721-0.219\times 1.72 = 1.703\end{aligned}$$

Following the same logic, the forecast of *y* prepared in *t* = 50 for three periods ahead is:

$$\begin{aligned}&y_{53}=\varphi_0+\varphi_1 y_{52}+\varphi_2 y_{51}+\varepsilon_{53}\\ &\quad\rightarrow\quad E(y_{53}\,|\,\Omega_{50}) = \varphi_0+\varphi_1 E(y_{52}\,|\,\Omega_{50})+\varphi_2 E(y_{51}\,|\,\Omega_{50})+E(\varepsilon_{53}\,|\,\Omega_{50})\\ &\qquad\qquad = \varphi_0+\varphi_1 E(y_{52}\,|\,\Omega_{50})+\varphi_2 E(y_{51}\,|\,\Omega_{50})\\ &\qquad\qquad \approx \hat{\varphi}_0+\hat{\varphi}_1 E(y_{52}\,|\,\Omega_{50})+\hat{\varphi}_2 E(y_{51}\,|\,\Omega_{50})\\ &\qquad\qquad = 0.045+1.182\times 1.703-0.219\times 1.721 = 1.681\end{aligned}$$

Exercise 2

Continuing the previous exercise, suppose that the test period is *t* = 51, 52, 53 and the true values of y_t in the test period are y_{51} = 1.70, y_{52} = 1.71 and y_{53} = 1.69. Given these true values and the forecasts in Exercise 1, calculate the *ME*, *MAE*, *RMSE*, *MPE* and *MAPE* measures of forecast accuracy for the test period with your calculator.

For the sake of simplicity, in the following solution some of the details were computed in Excel. Note, however, that on the exam you will have no access to a computer, so if you really want to benefit from this exercise, try to reproduce the results with your calculator only.

The observed values, forecasts, forecast errors, absolute errors, squared errors, percent errors and absolute percent errors over the test period, and their sums, are in the following table:

t	y_t	f_t	e_t	$\lvert e_t \rvert$	e_t^2	p_t	$\lvert p_t \rvert$
51	1.700	1.721	-0.021	0.021	0.000441	-1.235%	1.235%
52	1.710	1.703	0.007	0.007	0.000049	0.409%	0.409%
53	1.690	1.681	0.009	0.009	0.000081	0.533%	0.533%
Sum			-0.005	0.037	0.000571	-0.293%	2.177%

In this hypothetical example *T* – *t** = 3, so from the sums the forecast accuracy measures are

$$ME = \frac{1}{T-t^*}\sum_{t=t^*+1}^{T} e_t = \frac{-0.005}{3} = -0.0017$$

$$MAE = \frac{1}{T-t^*}\sum_{t=t^*+1}^{T} |e_t| = \frac{0.037}{3} = 0.0123$$

$$RMSE = \sqrt{\frac{1}{T-t^*}\sum_{t=t^*+1}^{T} e_t^2} = \sqrt{\frac{0.000571}{3}} = 0.0138$$

$$MPE = \frac{1}{T-t^*}\sum_{t=t^*+1}^{T} p_t = \frac{-0.293\%}{3} = -0.0977\%$$

$$MAPE = \frac{1}{T-t^*}\sum_{t=t^*+1}^{T} |p_t| = \frac{2.177\%}{3} = 0.7257\%$$

Let's now see a 'real' forecast project.

Exercise 3

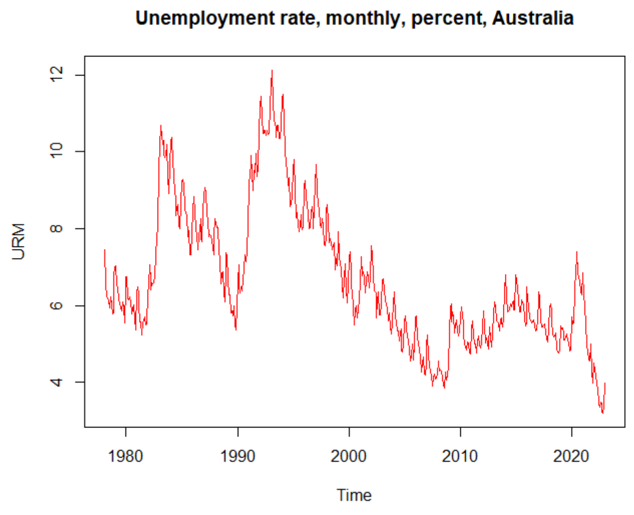
The *t7e3.xlsx* file contains observations on monthly unemployment rate (*URM*) in Australia from Feb 1978 to Jan 2023 (downloaded from ABS, 6/03/2023).

a) Plot *URM* and briefly describe the pattern of the data series.

Launch *RStudio*, create a new project and script, and name both *t7e3*. Import the data from the *URM* sheet and prepare a time series plot of *URM* by executing the following commands:

```
attach(t7e3)
URM = ts(URM, frequency = 12, start = c(1978,2), end = c(2023,1))
plot(URM, main = "Unemployment rate, monthly, percent, Australia", col = "red")
```

You should get,



This time-series plot shows that *URM* has an additive seasonal component.

Suppose we are not interested in the seasonal variations. Then it is better to eliminate them by changing the frequency of the data from monthly to annual. This can be achieved by the following function of the *tsbox R* library:

ts_frequency(x, to = , aggregate =)

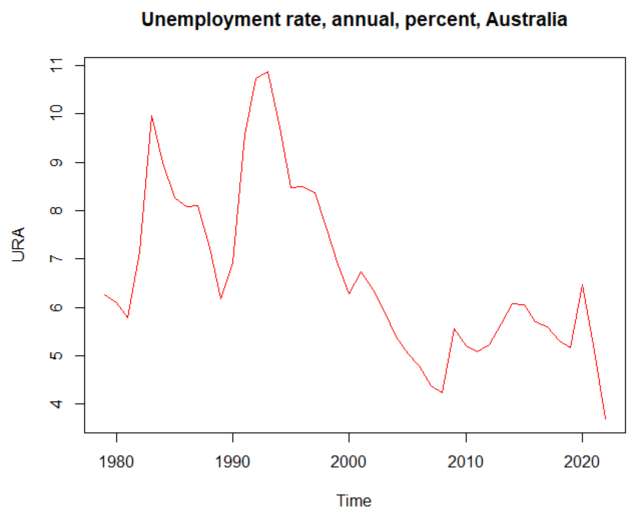
where x is a time series, to is the desired frequency given as a character string ("year", "quarter", "month") or as an integer (1, 4, 12), and aggregate is the function ("mean", "sum", "first", "last") used to generate the new values of x that conform to the desired frequency.

In this case *x* is *URM*, *to* is *year*, and since the annual unemployment rate is the average of the monthly unemployment rates in the given year, *aggregate* is *mean*. Hence, after having installed the *tsbox* library on your computer, execute

```
library(tsbox)
URA = ts_frequency(URM, to = "year", aggregate = "mean")
plot(URA, main = "Unemployment rate, annual, percent, Australia", col = "red")
```

You should get the plot at the top of the next page. As expected, the annual unemployment series is far smoother than the original monthly series.

It is also evident that *URA* (and *URM* as well) tends to decline, at least during this sample period.



b) Perform the *ADF* and *KPSS* tests to determine the level of integration of *URA*.

You can proceed as in Exercise 3 of Tutorial 5 and perform the tests both on the level and the first difference of *URA*. Based on the previous time series plot, use Model 3 for the level series and Model 2 for the differenced series. In both cases, let the lag length be determined by the automatic selection procedure based on *BIC*.

Execute

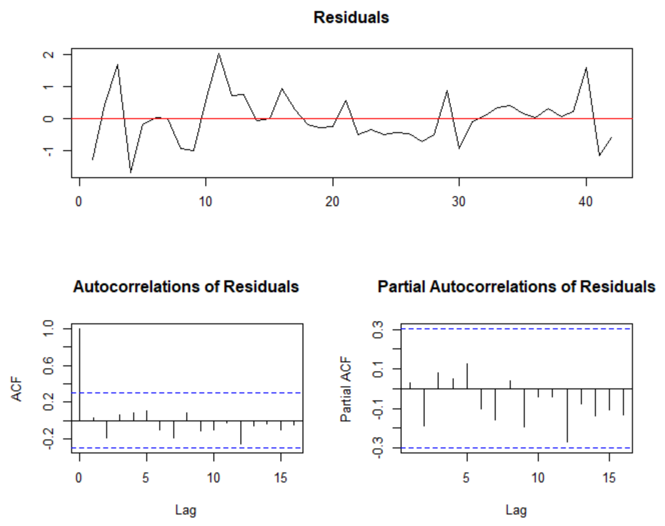
```
library(urca)
DF.URA = ur.df(URA, type = "trend", selectlags = "BIC")
summary(DF.URA)
plot(DF.URA
```

to get the printouts on the next page.

Recall that there are the results of three tests on this printout, but we are concerned only with the first one. The observed τ_3 test statistic value is -3.575, and it is between the 1% and 5% critical values, -4.15 and -3.50. Hence, at the 5% significance level we reject the unit root null hypothesis.

As you can also see on the printout, *R* used only one lag of the differenced *URA* (*z.diff.lag*) in the *ADF* test regression. Since the residual plot appears random and the sample correlograms do not have any significant spike (apart from the trivial r_0), one lag is sufficient.

```
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression trend


Call:
lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.67075 -0.49025  0.00632  0.38888  2.02883 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  3.16336    0.87174   3.629 0.000835 ***
z.lag.1     -0.34083    0.09534  -3.575 0.000974 ***
tt          -0.03987    0.01309  -3.046 0.004197 ** 
z.diff.lag   0.40280    0.14384   2.800 0.007982 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7987 on 38 degrees of freedom
Multiple R-squared:  0.3196,	Adjusted R-squared:  0.2659 
F-statistic: 5.951 on 3 and 38 DF,  p-value: 0.001977


Value of test-statistic is: -3.575 4.676 6.9327 

Critical values for test statistics: 
      1pct  5pct 10pct
tau3 -4.15 -3.50 -3.18
phi2  7.02  5.13  4.31
phi3  9.31  6.73  5.61
```

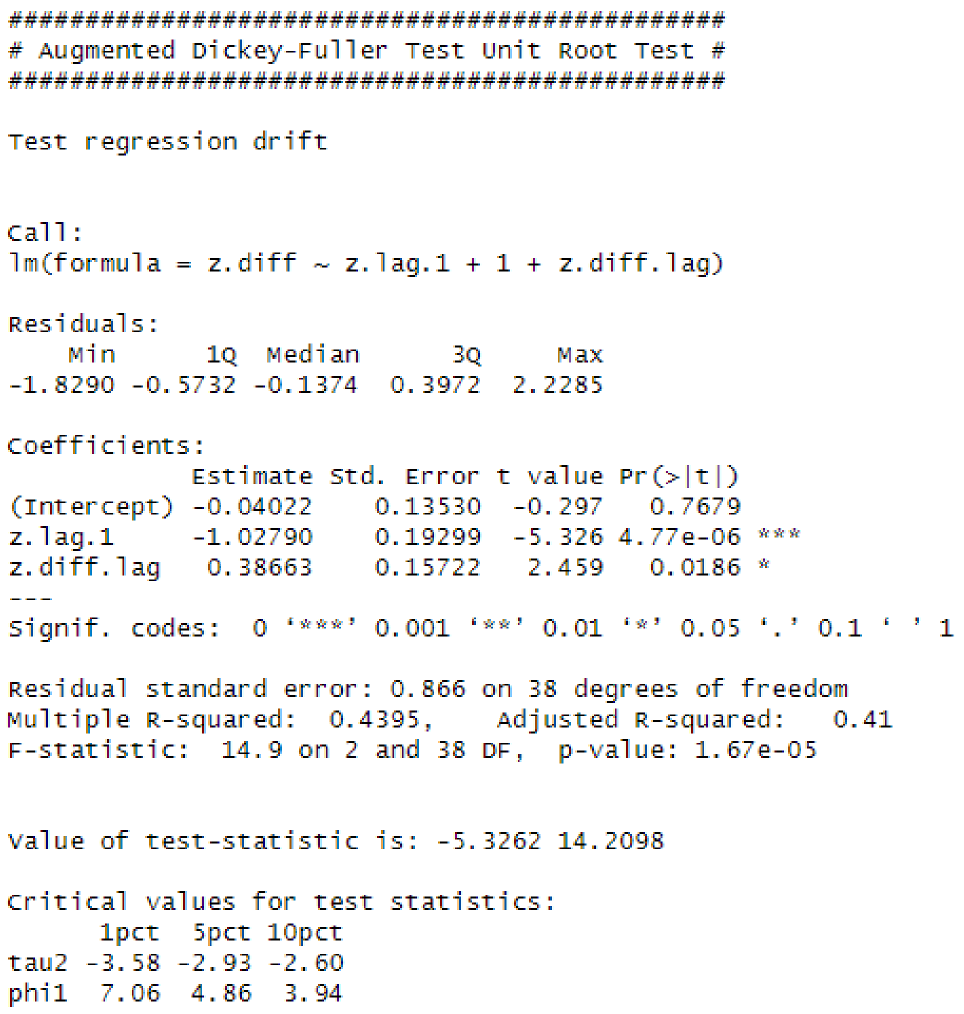


To repeat the *ADF* test on the first difference of *URA*, execute

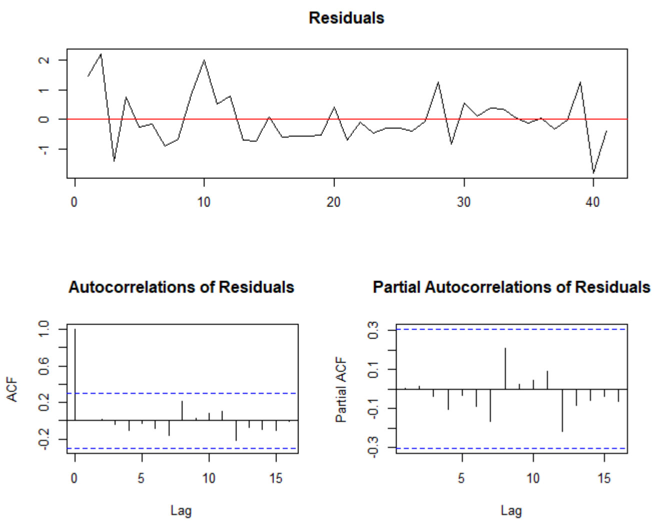
DF.DURA = ur.df(diff(URA), type = "drift", selectlags = "BIC")
summary(DF.DURA)
plot(DF.DURA)

The new printout is below and on the next page. Again, we are concerned about the first test only. The observed τ_2 test statistic value is -5.3262, and it is below the 1% critical value, -3.58. Hence, at the 1% significance level we reject the unit root null hypothesis.

There is again only one lag of the differenced *DURA* (*z.diff.lag*) in the *ADF* test regression. It is sufficient since the residual plot appears random and the sample correlograms do not have any significant spike (apart from the trivial r_0).



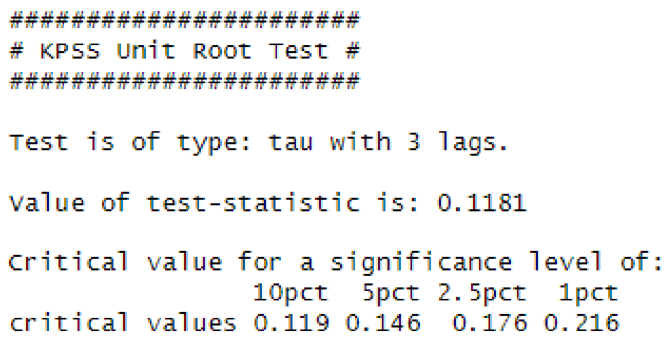
```
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression drift


Call:
lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.8290 -0.5732 -0.1374  0.3972  2.2285 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -0.04022    0.13530  -0.297   0.7679    
z.lag.1     -1.02790    0.19299  -5.326 4.77e-06 ***
z.diff.lag   0.38663    0.15722   2.459   0.0186 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.866 on 38 degrees of freedom
Multiple R-squared:  0.4395,	Adjusted R-squared:   0.41 
F-statistic:  14.9 on 2 and 38 DF,  p-value: 1.67e-05


Value of test-statistic is: -5.3262 14.2098 

Critical values for test statistics: 
      1pct  5pct 10pct
tau2 -3.58 -2.93 -2.60
phi1  7.06  4.86  3.94
```



To perform the *KPSS* test on the level of *URA*, execute the following commands:

KPSS.URA = ur.kpss(URA, type = "tau", lags = "short")
summary(KPSS.URA)

They return



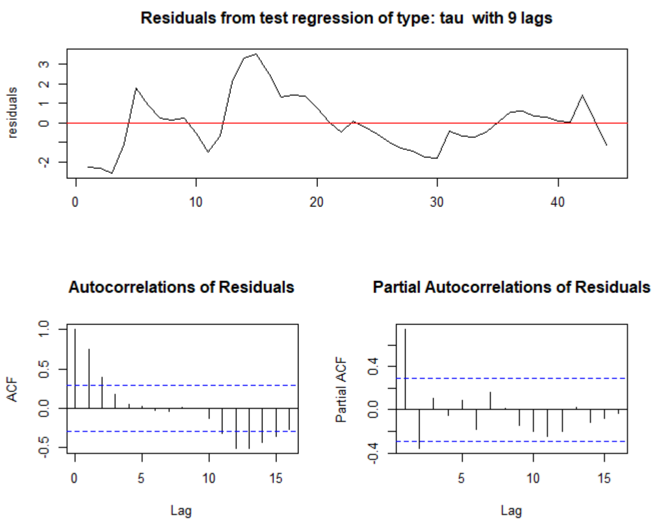
```
#######################
# KPSS Unit Root Test #
#######################

Test is of type: tau with 3 lags.

Value of test-statistic is: 0.1181

Critical value for a significance level of:
                10pct  5pct 2.5pct  1pct
critical values 0.119 0.146  0.176 0.216
```

The observed test statistic value is 0.1181, and it is below the 10% critical value, 0.119. Hence, the null hypothesis of stationarity is maintained even at the 10% significance level.[2]

To check the lag length, execute

plot(KPSS. URA)

[2] Recall that the *ADF* and *KPSS* tests have different decision rules. The rejection region is left to the critical value in the *ADF* test, but it is right to the critical value in the *KPSS* test.

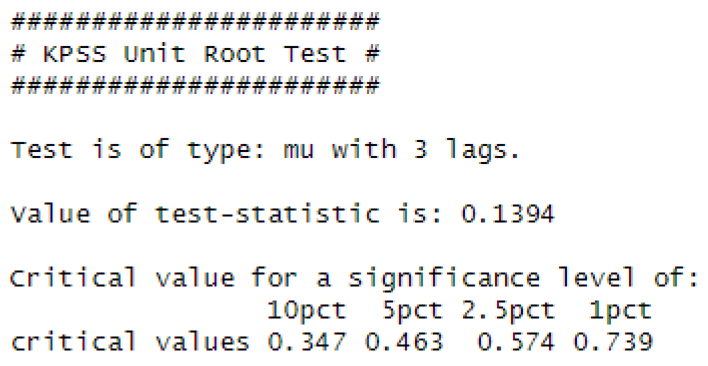


This residual plot does not seem random. Since the residuals change sign relatively infrequently, they appear to have positive (first order) autocorrelation. Also notice that *SACF* and *SPACF* alike have several significant spikes. For this reason, the *KPSS* test results on the level of *URA* must be taken with some reservation.[3]

To perform the *KPSS* test on the first difference of *URA*, execute the following commands:

KPSS.DURA = ur.kpss(diff(URA), type = "mu", lags = "short")
summary(KPSS.DURA)

They return



```
#######################
# KPSS Unit Root Test #
#######################

Test is of type: mu with 3 lags.

Value of test-statistic is: 0.1394

Critical value for a significance level of:
                10pct  5pct 2.5pct  1pct
critical values 0.347 0.463  0.574 0.739
```

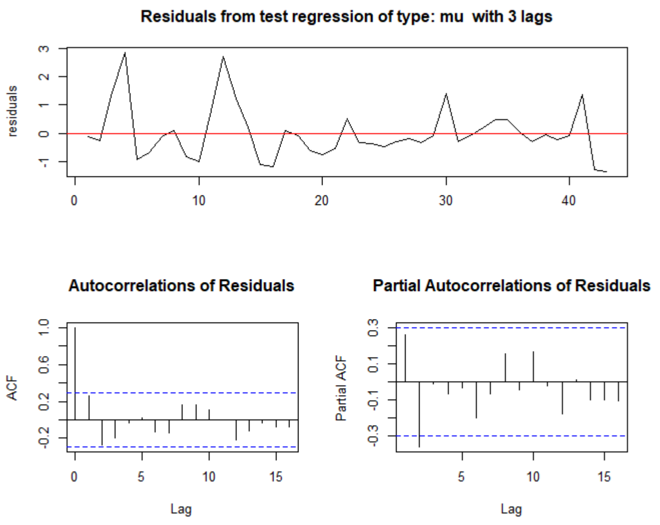
The observed test statistic value is 0.1394, and it is below the 10% critical value, 0.347. Hence, again, the null hypothesis of stationarity is maintained even at the 10% level.

[3] LK: Replace the *lags = "short"* argument with *lags = "long"*. As you will see, there are now 9 lags in the *KPSS* test regression, but unfortunately, the residuals are still autocorrelated this time.

Finally,

plot(KPSS.DURA)

returns



There are 3 lags in the *KPSS* test regression this time and they are probably sufficient, though *SPACF* has a significant spike at lag 2.

All things considered, the *ADF* tests found sufficient evidence in the data against the unit root null hypotheses for the level and the first difference of *URA* alike, while the *KPSS* tests failed to find sufficient evidence against the stationarity null hypotheses. These outcomes are best summarised in a table:

	ADF	*KPSS*
URA	stationary	stationary
DURA	stationary	stationary

Consequently, we conclude at the 5% significance level, at least, that *URA* is (trend) stationary and thus we can fit an *ARMA* model to it.

c) Split the sample data into two parts, *training* data from 1979 to 2018 and *test* data from 2019 to 2022 and use the *auto.arima()* function of *R* and the *AICc* specification criterion to find the best fitting *ARMA* model for *URA* over the training period.

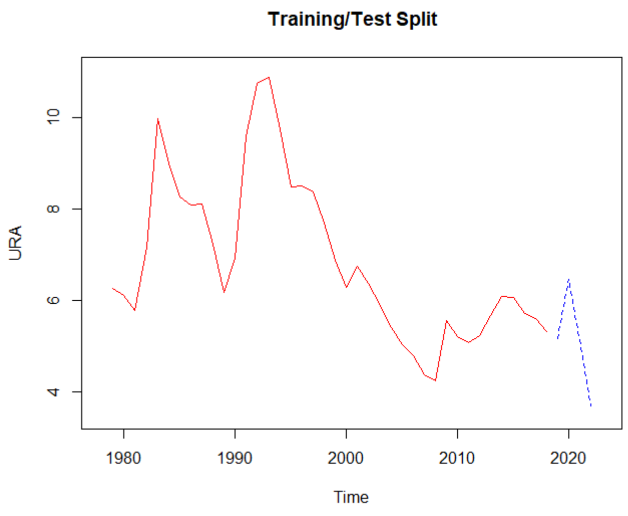
Execute

```
URA_train = window(URA, end = 2018)
URA_test = window(URA, start = 2019)
```

to split the sample data. The training data and test data can be illustrated by executing the following commands:

```
plot(URA_train, ylab = "URA", xlim = c(1978, 2023), ylim = c(3.5, 11),
     col = "red", type = "l", main = "Training/Test Split")
lines(URA_test, col = "blue", type = "l", lty = 2)
legend("topright", legend = c("Training", "Test"), col = c("red", "blue"), lty=1:2)
```

They return this plot:



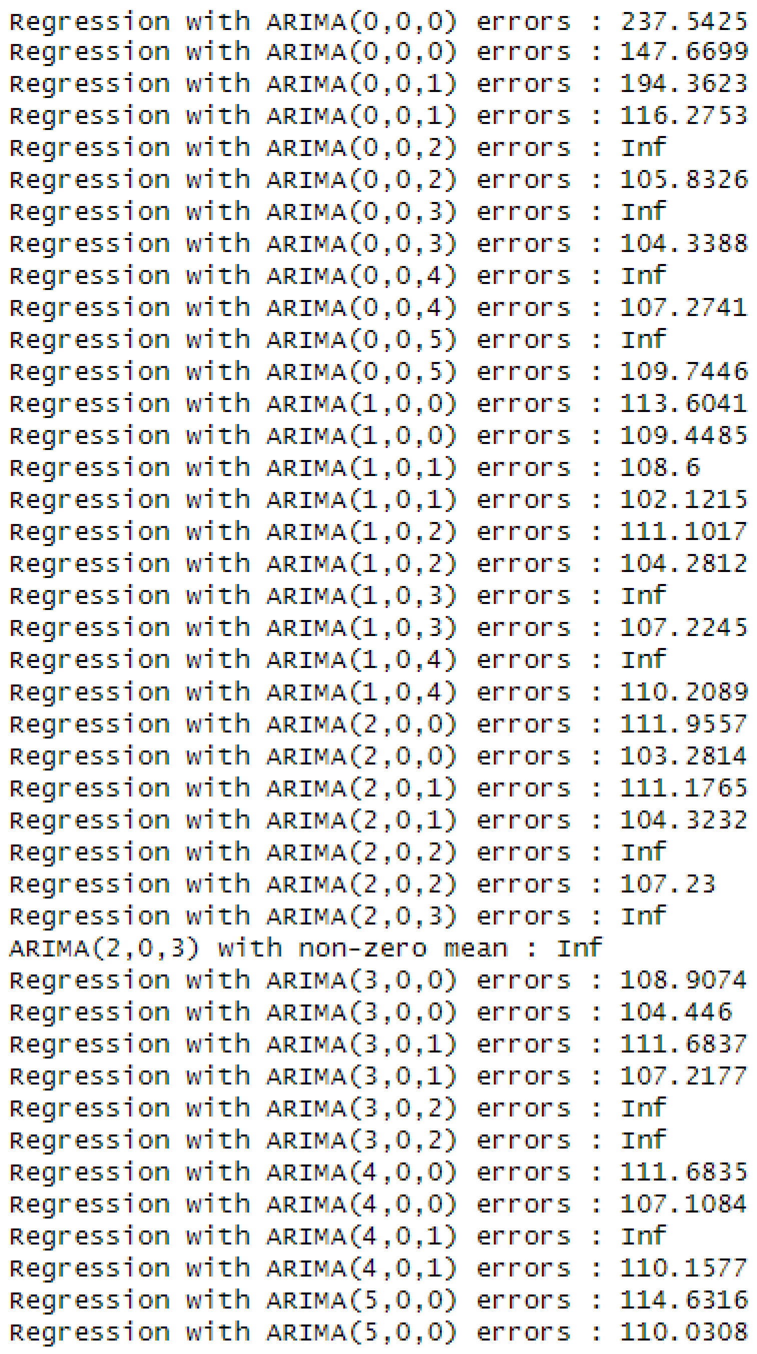
Next, create a time variable for the training period,

```
t = ts(1:length(URA_train), start = 1979, frequency = 1)
```

and use it as an extra independent variable in your search for the 'best' *ARMA* model,

```
library(forecast)
best_train = auto.arima(URA_train, seasonal = FALSE, xreg = t, ic = "aicc",
                        approximation = FALSE, stepwise= FALSE, trace = TRUE)
```

The specifications experimented with and their *AICc* values are shown on the next page. As you can see, the *ARMA*(1,1) and *ARMA*(2,0) models, both with intercept, have the smallest *AICc* values (102.1215 and 103.2814).



```
Regression with ARIMA(0,0,0) errors : 237.5425
Regression with ARIMA(0,0,0) errors : 147.6699
Regression with ARIMA(0,0,1) errors : 194.3623
Regression with ARIMA(0,0,1) errors : 116.2753
Regression with ARIMA(0,0,2) errors : Inf
Regression with ARIMA(0,0,2) errors : 105.8326
Regression with ARIMA(0,0,3) errors : Inf
Regression with ARIMA(0,0,3) errors : 104.3388
Regression with ARIMA(0,0,4) errors : Inf
Regression with ARIMA(0,0,4) errors : 107.2741
Regression with ARIMA(0,0,5) errors : Inf
Regression with ARIMA(0,0,5) errors : 109.7446
Regression with ARIMA(1,0,0) errors : 113.6041
Regression with ARIMA(1,0,0) errors : 109.4485
Regression with ARIMA(1,0,1) errors : 108.6
Regression with ARIMA(1,0,1) errors : 102.1215
Regression with ARIMA(1,0,2) errors : 111.1017
Regression with ARIMA(1,0,2) errors : 104.2812
Regression with ARIMA(1,0,3) errors : Inf
Regression with ARIMA(1,0,3) errors : 107.2245
Regression with ARIMA(1,0,4) errors : Inf
Regression with ARIMA(1,0,4) errors : 110.2089
Regression with ARIMA(2,0,0) errors : 111.9557
Regression with ARIMA(2,0,0) errors : 103.2814
Regression with ARIMA(2,0,1) errors : 111.1765
Regression with ARIMA(2,0,1) errors : 104.3232
Regression with ARIMA(2,0,2) errors : Inf
Regression with ARIMA(2,0,2) errors : 107.23
Regression with ARIMA(2,0,3) errors : Inf
ARIMA(2,0,3) with non-zero mean : Inf
Regression with ARIMA(3,0,0) errors : 108.9074
Regression with ARIMA(3,0,0) errors : 104.446
Regression with ARIMA(3,0,1) errors : 111.6837
Regression with ARIMA(3,0,1) errors : 107.2177
Regression with ARIMA(3,0,2) errors : Inf
Regression with ARIMA(3,0,2) errors : Inf
Regression with ARIMA(4,0,0) errors : 111.6835
Regression with ARIMA(4,0,0) errors : 107.1084
Regression with ARIMA(4,0,1) errors : Inf
Regression with ARIMA(4,0,1) errors : 110.1577
Regression with ARIMA(5,0,0) errors : 114.6316
Regression with ARIMA(5,0,0) errors : 110.0308
```

The *ARMA*(1,1) printout is displayed by the

summary(best_train)

command,

```
Series: URA_train
Regression with ARIMA(1,0,1) errors

Coefficients:
         ar1     ma1  intercept     xreg
      0.6696  0.5536     8.1824  -0.0703
s.e.  0.1275  0.1267     0.9859   0.0405

sigma^2 = 0.5985:  log likelihood = -45.18
AIC=100.36   AICc=102.12   BIC=108.8

Training set error measures:
                     ME      RMSE       MAE        MPE     MAPE     MASE
Training set 0.03357786 0.7339103 0.5627155 -0.6280353 8.051542 0.900061
```

while

second = Arima(URA_train, order = c(2,0,0), xreg = t)
summary(second)

display the *ARMA*(2,0) printout:

```
Series: URA_train
Regression with ARIMA(2,0,0) errors

Coefficients:
         ar1      ar2  intercept     xreg
      1.1640  -0.4534     8.4729  -0.0810
s.e.  0.1378   0.1430     0.8095   0.0338

sigma^2 = 0.6177:  log likelihood = -45.76
AIC=101.52   AICc=103.28   BIC=109.96

Training set error measures:
                     ME      RMSE       MAE       MPE     MAPE     MASE
Training set 0.01963738 0.7456095 0.5515288 -0.883973 7.957126 0.882168
```

These printouts are like the ones in the Tutorial 6 handout, except that previously we ignored the forecast accuracy measures at the bottom of the printouts. If you compare the corresponding measures to each other, you can see that according to *RMSE* and *MPE*, the *ARMA*(1,1) model produced the more accurate 'in-sample' forecasts for *URA*, but *ME*, *MAE*, *MAPE* and *MASE* favour the *ARMA*(2,0) model.[4]

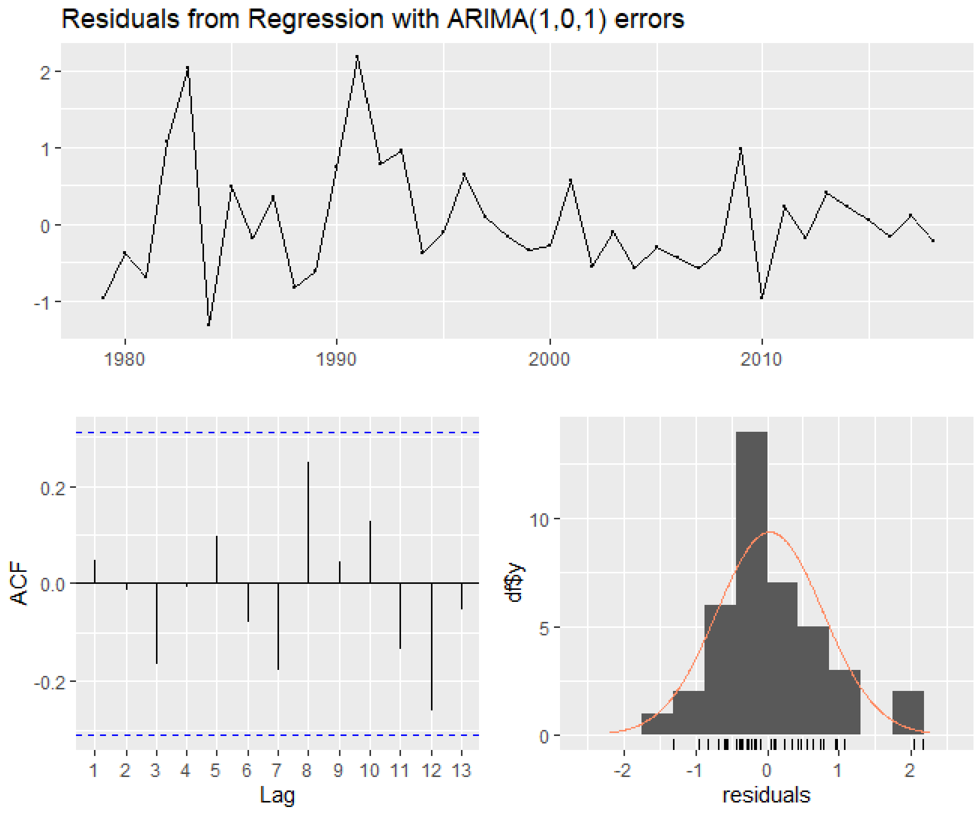
You might find this somewhat unexpected since over the training period the *ARMA*(1,1) model performed better, at least it returned the smaller *AICc* value. Note, however, that the model specification criteria, like *AICc*, serve different purposes than the forecast accuracy measures and thus they might rank the specifications differently.

[4] LK: There is one more error measure on the printouts, the estimate of the first-order autocorrelation coefficient of the forecast errors, *ACF1*. Do not worry about it, we do not use it as a forecast accuracy measure.

To check the adequacy of the *ARMA*(1,1) model, execute

checkresiduals(best_train)
shapiro.test(best_train$residuals)

They return



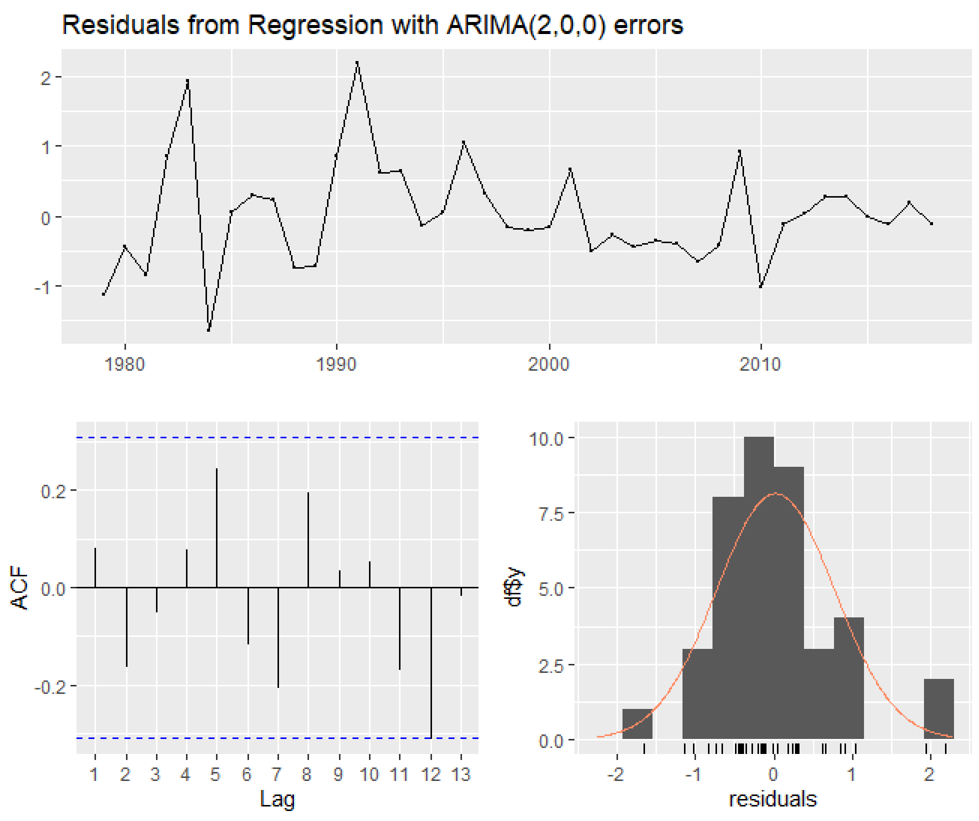
```
        Ljung-Box test

data:  Residuals from Regression with ARIMA(1,0,1) errors
Q* = 6.9536, df = 6, p-value = 0.3252

Model df: 2.   Total lags used: 8

        Shapiro-Wilk normality test

data:  best_train$residuals
W = 0.93689, p-value = 0.02719
```

The sample correlogram and the *LB* test suggest that the residuals are serially uncorrelated, while based on the histogram and the *SW* test the residuals are not normally distributed.

To check the adequacy of the *ARMA*(2,0) model, execute

> *checkresiduals(second)*
> *shapiro.test(second$residuals)*

They return



```
        Ljung-Box test

data:  Residuals from Regression with ARIMA(2,0,0) errors
Q* = 9.6672, df = 6, p-value = 0.1394

Model df: 2.   Total lags used: 8


        Shapiro-Wilk normality test

data:  second$residuals
W = 0.95739, p-value = 0.1363
```

The sample correlogram and the *LB* test again suggest that the residuals are serially uncorrelated, and based on the histogram and the *SW* test the residuals appear to be normally distributed this time. Hence, over the training period the *ARMA*(2,0) model proves to be more adequate.

d) Forecast *URA* for the 2019 – 2023 test period with both models. Which model produces the more accurate 'out-of-sample' ex post forecasts?

To generate these forecasts, first we need to extend the *t* time variable,

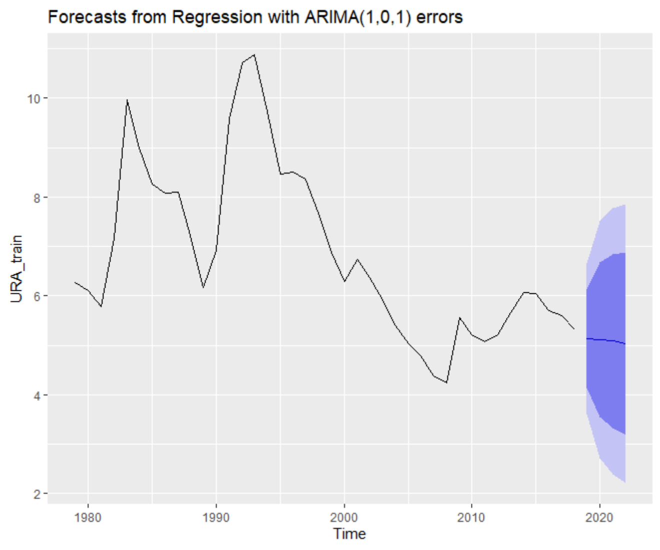
```
t = ts(1:length(URA), start = 1979, frequency = 1)
```

Then, to get the *ARMA*(1,1) multi-step forecasts, execute

```
ar1ma1_ftest = forecast(best_train, xreg = window(t, start = 2019))
print(ar1ma1_ftest)
autoplot(ar1ma1_ftest)
```

They return the following printout

```
     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2019       5.133553 4.142133 6.124974 3.617307 6.649800
2020       5.118269 3.551952 6.684586 2.722793 7.513744
2021       5.084804 3.320561 6.849047 2.386628 7.782981
2022       5.039167 3.193069 6.885265 2.215804 7.862530
```



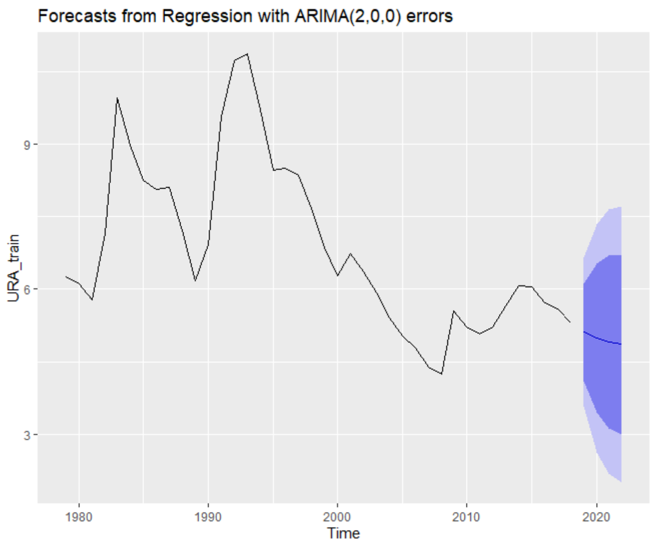
which shows the point forecasts and the lower (*Lo*) and upper (*Hi*) limits of the 80% and 95% prediction intervals.

To get the corresponding *ARMA*(2,0) multi-step forecasts, execute

```
ar2ma0_ftest = forecast(second, xreg = window(t, start = 2019))
print(ar2ma0_ftest)
autoplot(ar2ma0_ftest)
```

They return

```
     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2019       5.110516 4.103292 6.117740 3.570099 6.650933
2020       4.988241 3.442590 6.533891 2.624372 7.352110
2021       4.911841 3.119194 6.704488 2.170224 7.653458
2022       4.854894 2.986827 6.722962 1.997932 7.711857
```



The accuracy of these 'out-of-sample' ex post forecasts can be evaluated by the following function of the *forecast* package:

accuracy(f, x)

returns a range of summary measures of the forecast accuracy. f is a numerical vector of forecasts or the Arima or lm object used to generate the forecasts and x is the observed series.

In this case, *f* is *ar1ma1_ftest* or *ar2ma0_ftest* and *x* is *URA*. Hence, execute

```
accuracy(ar1ma1_ftest, URA_test)
```

to get

```
                     ME      RMSE       MAE        MPE      MAPE     MASE
Training set 0.03357786 0.7339103 0.5627155 -0.6280353  8.051542 0.900061
Test set     0.01746977 0.9485569 0.6860592 -3.5259676 14.534312 1.097349
```

and

accuracy(ar2ma0_ftest, URA_test)

to get

```
                     ME      RMSE       MAE        MPE      MAPE     MASE
Training set 0.01963738 0.7456095 0.5515288 -0.8839730  7.957126 0.882168
Test set     0.14504514 0.9419117 0.7214982 -0.8215184 14.749928 1.154033
```

The forecast accuracy measures are automatically reported for both the training set and the test set.[5] The training set measures are the same as on the printouts on page 15 and we already saw that according to most measures the *ARMA*(2,0) model produced the more accurate ex post 'in-sample' forecasts.

Let's now focus on the accuracy measures over the test set. If you compare the corresponding measures on the two printouts to each other, you can see that according to *ME*, *MAE*, *MAPE* and *MASE* the *ARMA*(1,1) model produced the more accurate 'out-of-sample' ex post forecasts, while *RMSE* and *MPE* are in favour of the *ARMA*(2,0) model. Going by the majority view, we can expect *ARMA*(1,1) to produce the more accurate ex ante forecasts as well.

e) Forecast *URA* for 2023-2026 with the *ARMA*(1,1) model.

In part (c) you used the *auto.arima* function to compare various specifications to each other over the training period, withholding the test set observations for evaluation. Now, after having concluded that *ARMA*(1,1) is the best performing model, it is time to re-estimate it from all available observations before generating the ex ante forecasts.

Execute the following commands:

ar1ma1 = Arima(URA, c(1,0,1), xreg = t)
summary(ar1ma1)

They return the printout shown on the next page. The new *ARMA*(1,1) sample regression equation is just a bit different from the previous one on page 15. This is good news since a model that is very sensitive to a few additional observations cannot be expected to produce reasonably accurate ex ante forecasts.

[5] LK: You can see two more measures on the printouts, *ACF1* and *Theil's U*. They are not shown in this handout because we do not use them.

```
Series: URA
Regression with ARIMA(1,0,1) errors

Coefficients:
         ar1     ma1  intercept     xreg
      0.6202  0.5613     8.3593  -0.0801
s.e.  0.1363  0.1341     0.8762   0.0330

sigma^2 = 0.6488:  log likelihood = -51.55
AIC=113.1   AICc=114.68   BIC=122.02

Training set error measures:
                     ME      RMSE       MAE        MPE     MAPE      MASE
Training set 0.03125849 0.7680081 0.5907548 -0.8533118 8.815547 0.8887828
```

To obtain the ex ante forecasts of *URA*, extend the *t* series by 4 years for 2023-206,

t = ts(1:(length(URA) + 4), start = 1979, frequency = 1)

and run

ar1ma1_eaf = forecast(ar1ma1, xreg = window(t, start = 2023))
print(ar1ma1_eaf)

The ex ante point forecasts and the 80% and 95% prediction intervals are

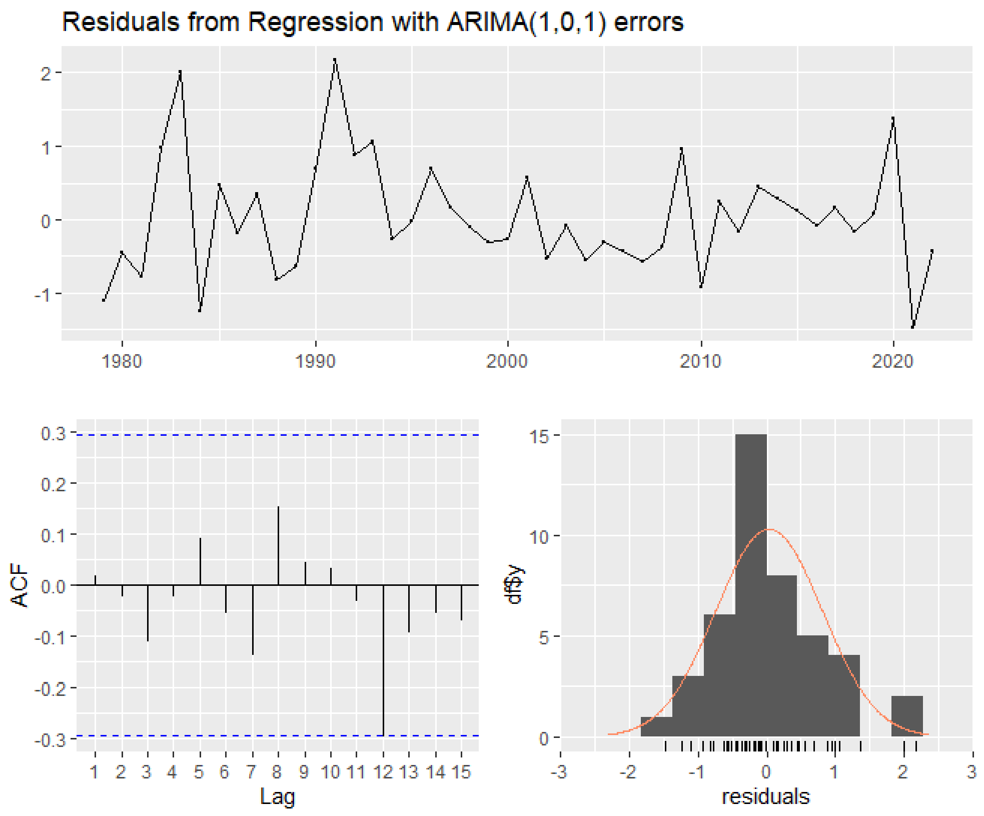
```
     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2023       3.812158 2.779877 4.844440 2.233420 5.390897
2024       4.089923 2.492106 5.687740 1.646272 6.533573
2025       4.231753 2.463963 5.999544 1.528151 6.935355
2026       4.289283 2.460322 6.118244 1.492128 7.086438
```

These prediction intervals, however, were generated by the *forecast* function assuming that the stochastic errors are serially uncorrelated and normally distributed. To check these assumptions, execute

checkresiduals(ar1ma1)
shapiro.test(ar1ma1$residuals)

It returns the printout shown on the next page.

The sample correlogram and the *LB* test indicate that the residuals are not autocorrelated and the histogram and the *SW* test do not provide evidence against normality. Hence, the stochastic errors might be serially uncorrelated and normally distributed.



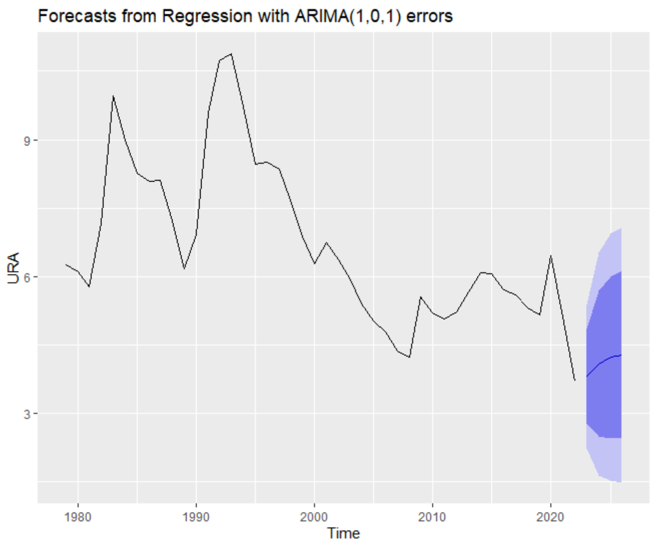
```
         Ljung-Box test

data:  Residuals from Regression with ARIMA(1,0,1) errors
Q* = 3.7692, df = 7, p-value = 0.8059

Model df: 2.   Total lags used: 9

         Shapiro-Wilk normality test

data:  ar1ma1$residuals
W = 0.96546, p-value = 0.2075
```

Finally, plot the observed values of *URA* along with these ex ante forecasts and prediction intervals by executing

autoplot(ar1ma1_eaf)

It returns the plot shown on the next page.

Save your *R* code and quit *RStudio*.


Forecasts from Regression with ARIMA(1,0,1) errors
URA
9
6
3
1980
1990
2000
2010
2020
Time