ECON90033 – QUANTITATIVE ANALYSIS OF FINANCE I

TUTORIAL 6

Download the t6e1 and t6e2 Excel data files 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.

Box-Jenkins Methodology

Box and Jenkins (1970) proposed a three-step strategy aimed at selecting and estimating an appropriate mixed autoregressive moving average (*ARMA*) model for the purpose of forecasting univariate time series. The steps are as follows.

1) Model selection:

   Illustrate the $\{y_t\}$ sample data with a time series plot, develop the sample correlograms, and perform various tests for autocorrelation (Bartlett, Box-Pierce, Ljung-Box tests). Based on the results and assuming stationarity, try to match the patterns of *SACF* and *SPACF* with the theoretical *ACF* and *PACF* patterns of various relatively simple *ARMA* processes and select one or more tentative *ARMA*(p,q) models.

   If the data are nonseasonal, you can rely on five simple rules of thumb:

   i. If there is not any significant r_k or ϕ_{kk}-hat, $\{y_t\}$ is essentially a white-noise series which is not amenable to *ARMA* modelling.

   ii. If r_k's decrease slowly and linearly passing through zero become negative, or if they exhibit a wave-like cyclical pattern passing through zero several times, $\{y_t\}$ is not stationary and it cannot be modelled as an *ARMA* process.

   iii. If r_k's approach zero gradually following an exponential or damped sine wave pattern, while the first p ϕ_{kk}-hats are significant and *SPACF* has a sharp cut-off point at lag p, $\{y_t\}$ can be modelled as an *AR*(p) process.

   iv. If ϕ_{kk}-hats approach zero gradually following an exponential or damped sine wave pattern, while the first q r_k's are significant and *SACF* has a sharp cut-off point at lag q, $\{y_t\}$ can be modelled as an *MA*(q) process.

   v. If both r_k's and ϕ_{kk}-hats converge to zero following some exponential or damped sine wave patterns beginning at lags q and p, respectively, $\{y_t\}$ might be modelled as an *ARMA*(p,q) process.[1]

[1] In practice this last rule of thumb is the least useful because when p and q are both positive, it is usually impossible to guess their correct values from the sample correlograms.

Also keep in mind that the tentatively selected models should be as simple as possible, of course, without sacrificing a reasonable fit to the data.

2) Model estimation:

   We estimate *ARMA* models with the *Arima()* function of *R* from the *Forecast* package.[2]

3) Diagnostic checks:

   If in step 1 more than one *ARMA* model was selected and each passed the residual checks, their performances can be compared to each other on the basis of some model selection criterion, like *AIC*, *AICc* or *BIC*.[3]

   The residuals from the preferred model should behave as a white noise. This requirement can be verified with the tools already used in step 1 on the $\{y_t\}$ sample data, just this time they are used on the residuals. It might also be important to check whether the residuals are normally distributed, especially when the sample size is relatively small.

   If based on the results so far, a given *ARMA*(p_0,q_0) model appears to be adequate and the 'best', as a final check, it is useful to estimate seemingly overparameterized models, *ARMA*(p_0 +1,q_0), *ARMA*(p_0,q_0 +1) and *ARMA*(p_0 +1,q_0 +1), as well. If the *ARMA*(p_0,q_0) model is indeed the 'best', the extra term(s) should prove to be insignificant.

So far we have considered *ARMA* models whose independent variables were all lags of the dependent variable (y_{t-1}, y_{t-2}, etc.) or/and of the error term (ε_{t-1}, ε_{t-2}, etc.). In many applications, however, the dependent variable is related to additional regressors as well.

The following exercise serves two purposes. It is as an example for the application of the Box-Jenkins methodology and for the inclusion of additional regressors in *ARMA* models.

Exercise 1

The *t6e1.xlsx* file contains quarterly, seasonally adjusted index of Canadian employment rate[4] (*CER*, %) from 1976 Q1 to 2023 Q1. You are going to analyse this data set, following the three stages of the Box-Jenkins methodology.

[2] You can refresh your memory about *Arima()* by reviewing the Tutorial 4 handout. Note that *R* has an *arima()* function as well, but we use *Arima()*.
[3] These statistics were also introduced in the Tutorial 4 handout.
[4] Employment Rate: Aged 15 and over: All Persons for Canada, Percent, Quarterly, Seasonally Adjusted.

Model selection

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

Launch *RStudio*, create a new project and script, and name both *t6e1*. Import the data and then prepare a time series plot of *CER* by executing the following commands:

```
attach(t6e1)
CER = ts(CER, start = c(1976,1), end = c(2023,1), frequency = 4)
plot.ts(CER, main = "Employment rate, %, Canada", col = "red")
```

You should get the following plot.

It shows that during the sample period the Canadian employment rate probably had several cycles and a general tendency to grow. Due to the slowly evolving cyclical behaviour (high in business cycle booms and low in recessions) and the general trend, *CER* must have strong positive (first and higher order) autocorrelation. This is a quarterly series but seasonally adjusted, and indeed there is no sign of seasonality.

It is also clear that *CER* had a large dip at around 2000 coinciding with the Covid-19 outbreak. Since this was a clearly uncharacteristic period, at this stage it is better to ignore the data since the first quarter of 2020.

We can do so using the following function:

window(y, start = , end = , frequency =)

where y is a time-series object, start and end specify the start time and the end time of the period of interest, and frequency is the number of observations per unit of time.

Hence, to consider *CER* only up untill the end of 2019, execute

CER_s = window(CER, start = c(1976,1), end = c(2019,4), frequency = 4)

and then plot the shortened series as you plotted the original series, i.e., by executing

plot.ts(CER_s, main = "Employment rate, %, Canada", col = "red")

This is the new plot:

CER appears to have a long-run upward trend, so its expected value is obviously not constant, and thus *CER* is not stationary.

This trend might dominate the dynamics of the stochastic component of the data making it difficult to identify the stochastic component. Assuming that this trend is deterministic, it can be removed by regressing *CER_s* on the $t = 1, 2, \ldots, T$ time variable and taking the residuals from this regression.

Execute

```
t = ts(1:length(CER_s), start = c(1976,1), end = c(2019,4), frequency = 4)
Trend_CER_s = lm(CER_s ~ t)
summary(Trend_CER_s)
```

The trend regression printout is on the next page.

The estimated trend is

$$\widehat{CER\_s} = 57.918826 + 0.028194t$$

It is significant and the slope estimate is significantly positive.

```
Call:
lm(formula = CER_s ~ t)

Residuals:
    Min      1Q  Median      3Q     Max
-2.6083 -0.9712 -0.1440  1.0673  2.8569

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 57.918826   0.194817  297.30   <2e-16 ***
t            0.028194   0.001909   14.77   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.287 on 174 degrees of freedom
Multiple R-squared:  0.5562,    Adjusted R-squared:  0.5537
F-statistic: 218.1 on 1 and 174 DF,  p-value: < 2.2e-16
```

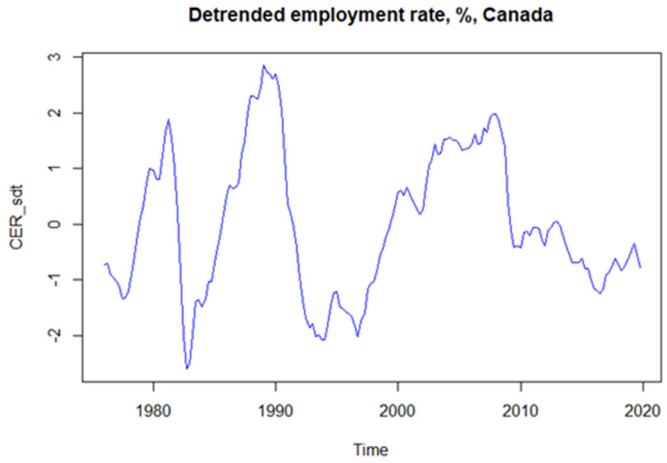
The detrended *CER_s* series (*CER_sdt*) is the residual series from this trend regression, i.e.,

```
CER_sdt = ts(Trend_CER_s$residuals,
             start = c(1976,1), end = c(2019,4), frequency = 4)
```

Illustrate it by executing

```
plot.ts(CER_sdt, main = "Detrended employment rate, %, Canada", col = "blue")
```

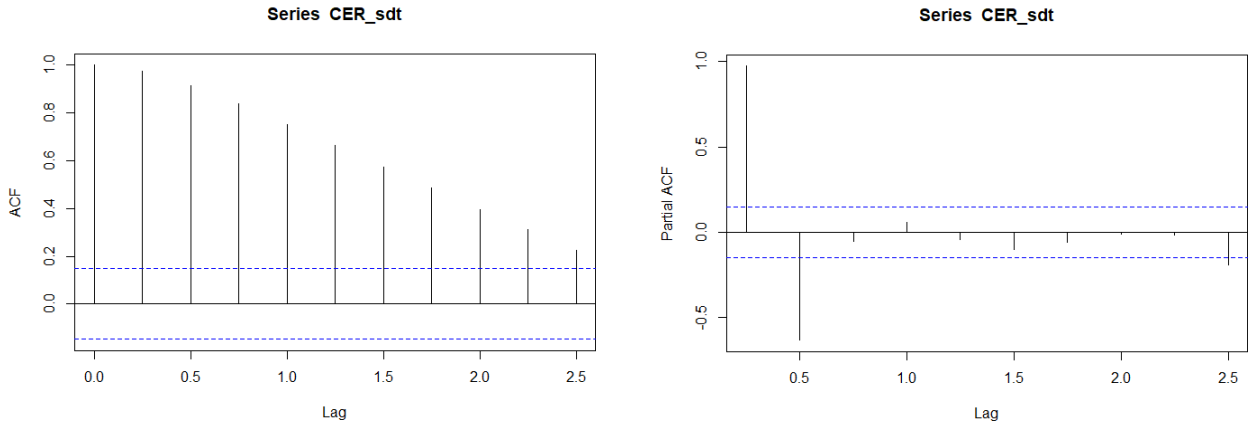
You should get:



b) Plot *SACF* and *SPACF* of *CER_sdt.*

ka = floor(min(10, length(CER_sdt)/5))
acf(CER_sdt, lag.max = ka, plot = TRUE)
pacf(CER_sdt, lag.max = ka, plot = TRUE)

return the following sample correlograms:



The sample autocorrelation coefficients decay slowly and are all significant, but only the first two sample partial autocorrelations are significant. In other words, the *SACF* displays slow one-sided (i.e., not oscillating) damping and decay, while the *SPACF* cuts off at lag (displacement) 2.[5]

c) Based on the sample correlograms, what *ARMA*(*p*,*q*) specification do you expect to model *CER-sdt* best?

If you return to the rules of thumb on page 1, you can see that the patterns of these sample correlograms probably best correspond to case (iii). Hence, our candidate is an *AR*(2) model.[6]

d) Estimate your preferred *ARMA*(*p*,*q*) models augmented with a deterministic linear trend for *CER_s*, i.e., for the shortened but not de-trended *CER* variable, and briefly evaluate the results.

[5] If, looking at the horizontal axes of these correlograms, you do not understand why at lag 2, revise Exercise 2 of Tutorial 4. You can also use the *as.numeric()* function and re-create the correlograms with integer lags. For the sake of illustration, execute *acf(as.numeric(CER_sdt), lag.max = ka, plot = TRUE)* and *pacf(as.numeric(CER_sdt), lag.max = ka, plot = TRUE).*

[6] In this case the sample correlograms do not really give a clear hint about a second candidate, but a low order mixed model like *ARMA*(1,1) is often a reasonably option.

Model estimation

As mentioned on page 2, in this course we estimate *ARMA* models with the *Arima()* function of the *forecast* package. Additional regressors can be added to the specifications with the optional *xreg* argument. It provides the list of the extra regressors.

For the tentatively selected *AR*(2) = *ARMA*(2,0) model, $p = 2$ and $q = 0$, and the t time variable is an additional regressor. Hence, execute the following commands:

```
library(forecast)
ar2ma0_CER_s = Arima(CER_s, order = c(2,0,0), xreg = t)
summary(ar2ma0_CER_s)
```

The top part of the printout is

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

Coefficients:
         ar1      ar2  intercept    xreg
      1.6489  -0.6905    57.9069  0.0275
s.e.  0.0535   0.0535     0.6696  0.0064

sigma^2 = 0.04134:  log likelihood = 30.49
AIC=-50.98   AICc=-50.62   BIC=-35.12
```

To understand this printout, it is important to mention that when there is some additional regressor (intercept, deterministic trend, some other independent variable), the *Arima()* function estimates the *ARMA* model with all the dynamics placed in the error term.

In this case, for example, the population *ARMA* model is

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

and *Arima()* estimates

$$y_t = \tilde{\varphi}_0 + \tilde{\lambda} t + \eta_t \quad \text{where} \quad \eta_t = \varphi_1 \eta_{t-1} + \varphi_2 \eta_{t-2} + \varepsilon_t$$

From the previous printout the corresponding sample regression model is

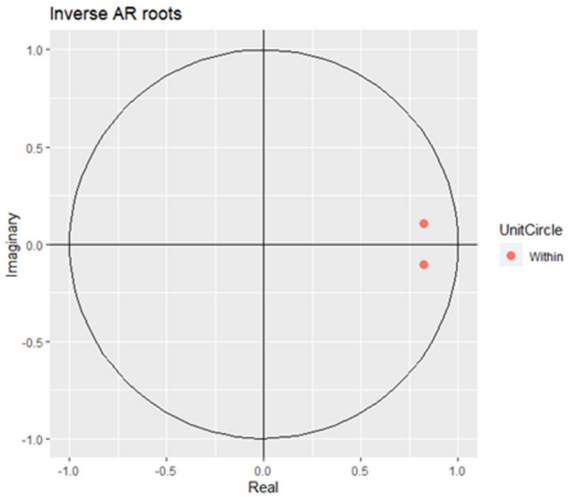
$$y = CER\_s$$
$$y_t = 57.9069 + 0.0275t + \eta_t$$
$$\eta_t = 1.6489\eta_{t-1} - 0.6905\eta_{t-2} + e_t$$

In the week 3 lectures you were told (see slide 26) that a finite order *AR*(p) process is stationary if all its characteristic roots are inside the unit circle. Although, due to lack of time, we did not discuss what these characteristic roots were and how to calculate them,

they can be easily obtained and illustrated by executing the

autoplot(ar2ma0_CER_s)

command. It returns the following graph:



It shows a two dimensional, real-imaginary coordinate system.[7] The circle, the so-called unit circle is centered around the origin and its radius is one. The red dots illustrate the *AR* characteristic roots, called inverse *AR* roots in *R*. They are not on the horizontal (real) axis, so they are complex numbers. Most importantly, they are within the unit circle, thus the estimated *AR*(2) process is stationary.

Diagnostic checks

First, get *t*-tests results for the coefficients of the *AR*(2) model by executing

library(lmtest)
coeftest(ar2ma0_CER_s, df = ar2ma0_CER_s$nobs - length(ar2ma0_CER_s$coef))

On the printout (see next page) every *p*-value is practically zero, so every coefficient is significant.

[7] LK: If you have never learnt about complex numbers, do not worry about this real-imaginary coordinate system, just focus on the circle and the two red dots inside it.

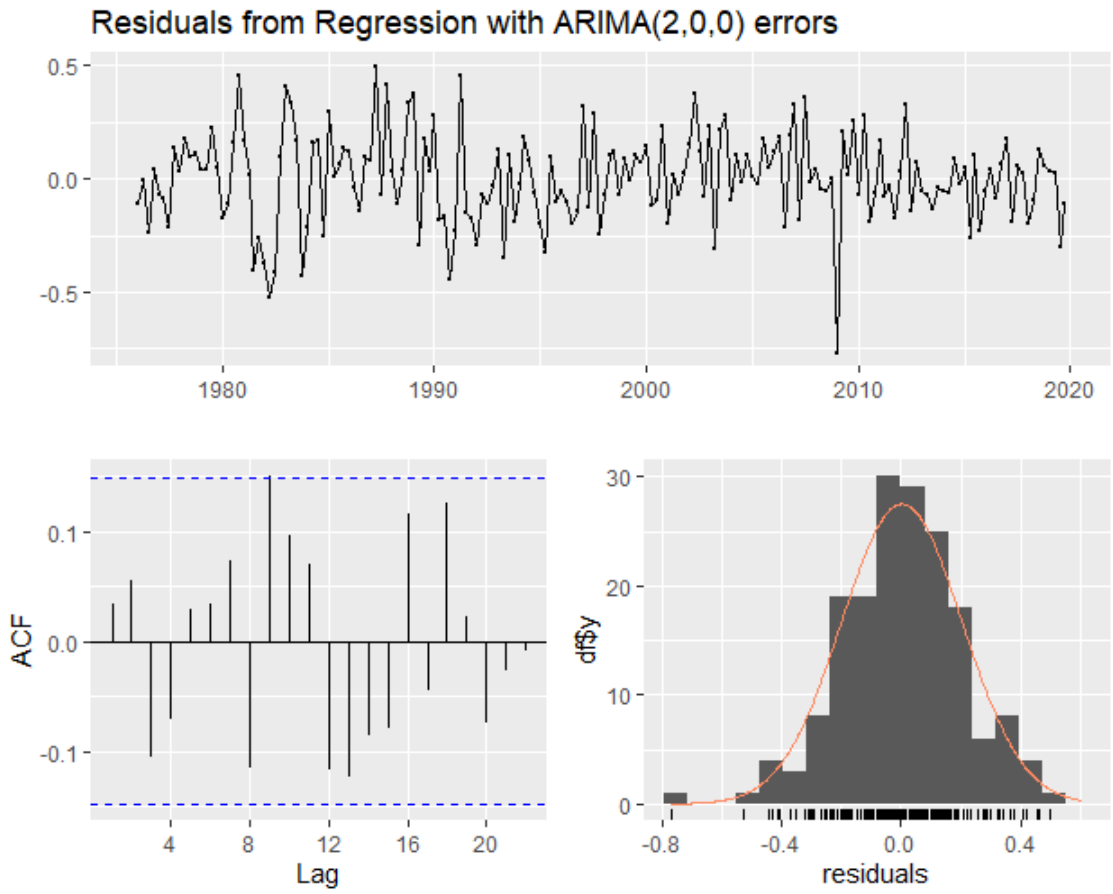
```
t test of coefficients:

             Estimate Std. Error  t value  Pr(>|t|)
ar1         1.6488850  0.0534867  30.8280 < 2.2e-16 ***
ar2        -0.6905421  0.0535226 -12.9019 < 2.2e-16 ***
intercept 57.9068562  0.6695797  86.4824 < 2.2e-16 ***
xreg       0.0275165  0.0064428   4.2709 3.214e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Then, study the residuals by executing the

checkresiduals(ar2ma0_CER_s)

command. It returns the following image:



It has three panels. The residual plot in the top panel seems random, though the pattern looks to change a bit after 1990. The sample correlogram in the bottom left panel does not have any significant spike. Finally, the histogram of the residuals in the bottom right panel is more or less symmetric like the superimposed normal curve.

The *checkresiduals()* function also provides the result of the LB test on the residuals:

```
        Ljung-Box test

data:  Residuals from Regression with ARIMA(2,0,0) errors
Q* = 7.4557, df = 6, p-value = 0.2808

Model df: 2.   Total lags used: 8
```

The reported *p*-value is 0.2808, so we maintain the null hypothesis of no autocorrelation of orders 1-8 in the residuals.

All in all, this *AR*(2) model seems adequate so far.

e) As mentioned on page 2, as a further diagnostic check, it is recommended to estimate larger and thus seemingly overfitted models to see whether the extra lags improve the quality of the model.

This check is actually automatically performed by the *auto.arima* function of the *forecast* package, so execute the

auto.arima(CER_s, xreg = t, ic = "aicc", seasonal = FALSE, trace = TRUE, stepwise = FALSE, approximation = FALSE)

command. It returns a long printout consisting of two parts. The top part, not displayed here, shows the considered specifications and the corresponding *AICc* values. The bottom part of the printout shows the 'best' model:

```
 Best model: Regression with ARIMA(2,0,0)

Series: CER_s
Regression with ARIMA(2,0,0) errors

Coefficients:
         ar1      ar2  intercept    xreg
      1.6489  -0.6905    57.9069  0.0275
s.e.  0.0535   0.0535     0.6696  0.0064

sigma^2 = 0.04134:  log likelihood = 30.49
AIC=-50.98   AICc=-50.62   BIC=-35.12
```

It is the same *AR*(2) model that we estimated and discussed in part (d).

It is also interesting to check whether the choice of the model selection criterion makes a real difference this time. To see this, execute

```
auto.arima(CER_s, xreg = t, ic = "aic", seasonal = FALSE, trace = FALSE,
      stepwise = FALSE, approximation = FALSE)
auto.arima(CER_s, xreg = t, ic = "bic", seasonal = FALSE, trace = FALSE,
      stepwise = FALSE, approximation = FALSE)
```

These commands rank the models on the basis of their *AIC* values and *BIC* values, respectively.[8] Still, *AR*(2) proves to be the best model.

f) So far, we have ignored the observations from 2020-2023 in order to avoid the unusually sharp dip of *CER* due to the Covid-19 outbreak and the accompanying downturn. How can we expand our model to make it capable of handling that turbulent period as well?

In general, unusually large or small observations can be handled by dummy variables, i.e., by augmenting the model by one dummy variable for each seemingly extreme observation. If you review the time series plot of *CER* on page 3, or open the *Excel* data file,

2019-07-01	62.30
2019-10-01	62.10
2020-01-01	60.87
2020-04-01	53.67
2020-07-01	58.37
2020-10-01	59.60
2021-01-01	59.63
2021-04-01	59.90
2021-07-01	60.87
2021-10-01	61.57
2022-01-01	61.83
2022-04-01	62.23

you can see that by the third quarter of 2021 *CER* returned to its value in the first quarter of 2020. Hence, we are going to use 7 dummy variables for the quarters from 2020 Q1 up to and including 2021 Q3.

These dummy variables can be generated in *R* by executing the following commands:

```
I1 = ts(ifelse(Date >= "2020-01-01" & Date < "2020-04-01", 1, 0),
      start = c(1976,1), end = c(2023,1), frequency = 4)
I2 = ts(ifelse(Date >= "2020-04-01" & Date < "2020-07-01", 1, 0),
      start = c(1976,1), end = c(2023,1), frequency = 4)
I3 = ts(ifelse(Date >= "2020-07-01" & Date < "2020-10-01", 1, 0),
      start = c(1976,1), end = c(2023,1), frequency = 4)
I4 = ts(ifelse(Date >= "2020-10-01" & Date < "2021-01-01", 1, 0),
      start = c(1976,1), end = c(2023,1), frequency = 4)
I5 = ts(ifelse(Date >= "2021-01-01" & Date < "2021-04-01", 1, 0),
      start = c(1976,1), end = c(2023,1), frequency = 4)
I6 = ts(ifelse(Date >= "2021-04-01" & Date < "2021-07-01", 1, 0),
      start = c(1976,1), end = c(2023,1), frequency = 4)
I7 = ts(ifelse(Date >= "2021-07-01" & Date < "2021-10-01", 1, 0),
      start = c(1976,1), end = c(2023,1), frequency = 4)
```

[8] To simplify the printouts, this time *trace* = *FALSE.*

To check these dummy variables, execute

```
options(max.print = 2000)
print(cbind(I1, I2, I3, I4, I5, I6, I7))
```

The top and bottom parts of the printout are below:

```
        I1 I2 I3 I4 I5 I6 I7
1976 Q1  0  0  0  0  0  0  0
1976 Q2  0  0  0  0  0  0  0
1976 Q3  0  0  0  0  0  0  0
1976 Q4  0  0  0  0  0  0  0
1977 Q1  0  0  0  0  0  0  0
1977 Q2  0  0  0  0  0  0  0
  ├
  ├
2019 Q1  0  0  0  0  0  0  0
2019 Q2  0  0  0  0  0  0  0
2019 Q3  0  0  0  0  0  0  0
2019 Q4  0  0  0  0  0  0  0
2020 Q1  1  0  0  0  0  0  0
2020 Q2  0  1  0  0  0  0  0
2020 Q3  0  0  1  0  0  0  0
2020 Q4  0  0  0  1  0  0  0
2021 Q1  0  0  0  0  1  0  0
2021 Q2  0  0  0  0  0  1  0
2021 Q3  0  0  0  0  0  0  1
2021 Q4  0  0  0  0  0  0  0
2022 Q1  0  0  0  0  0  0  0
2022 Q2  0  0  0  0  0  0  0
2022 Q3  0  0  0  0  0  0  0
2022 Q4  0  0  0  0  0  0  0
2023 Q1  0  0  0  0  0  0  0
```

As you can see, each dummy variable is equal to one in one quarter only and zero otherwise.

Since we intend to consider the full sample period, we also have to expand the time variable for 2020-2023. Hence, redefine *t* by executing

```
t = ts(1:length(CER), start = c(1976,1), end = c(2023,1), frequency = 4)
```

This time variable and the *I1*, …, *I7* dummy variables are additional regressors and they must be combined with the *cbind()* column bind function in order to include them in *xreg*. Hence, to find the best *ARMA* model over the whole sample period, execute

```
arima.new = auto.arima(CER, xreg = cbind(t, I1, I2, I3, I4, I5, I6, I7),
                       ic = "aicc", seasonal = FALSE, trace = TRUE,
                       stepwise = FALSE, approximation = FALSE)
summary(arima.new)
```

The bottom part of the new printout is at the top of the next page. As you can see, based on *AICc*, again *AR*(2) proved to be the best specification.

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

Coefficients:
         ar1      ar2  intercept       t       I1       I2       I3
      1.6398  -0.6812    57.9387  0.0269  -1.0509  -8.0912  -3.2523
s.e.  0.0529   0.0529     0.6520  0.0058   0.1865   0.3279   0.4215
          I4       I5       I6       I7
      -1.9092  -1.8007  -1.5006  -0.5683
s.e.   0.4541   0.4218   0.3283   0.1867

sigma^2 = 0.04294:  log likelihood = 32.84
AIC=-41.69   AICc=-39.92   BIC=-2.79
```

The new sample regression model is

$$y = CER$$

$$y_t = 57.9387 + 0.0269t - 1.0509 \times I1_t - 8.0912 \times I2_t - 3.2523 \times I3_t - 1.9092 \times I4_t - 1.8007 \times I5_t - 1.5006 \times I6_t - 0.5683 \times I7_t + \eta_t$$

$$\eta_t = 1.6398\eta_{t-1} - 0.6812\eta_{t-2} + e_t$$

If you compare this sample regression model to the previous sample regression model on page 10, you can see that the coefficients of *ar1*, *ar2*, *intercept* and *t* (*xreg* on the first printout) are very similar.

Note, however, that you cannot compare the *AIC*, *AICc* and *BIC* values of the two regressions to each other because they were estimated over different sample periods. Still, it is important to verify whether the augmented specification is supported by the data. We can do so by checking the significance of the dummy variables. The

coeftest(arima.new, df = arima.new$nobs - length(arima.new$coef))

command returns the printout shown on the next page. Since the reported two-tail *p*-values are all very small, the slope estimate of every dummy variable is significantly negative. This implies that these dummy variables are also significant as a group and the augmented *AR*(2) model is better than the original *AR*(2) model.

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

```
t test of coefficients:

             Estimate Std. Error  t value  Pr(>|t|)
ar1         1.6397582  0.0528508  31.0262 < 2.2e-16 ***
ar2        -0.6812266  0.0529387 -12.8682 < 2.2e-16 ***
intercept 57.9387437  0.6520019  88.8628 < 2.2e-16 ***
t          0.0268805  0.0058492   4.5956 8.151e-06 ***
I1        -1.0508586  0.1864714  -5.6355 6.725e-08 ***
I2        -8.0911512  0.3279393 -24.6727 < 2.2e-16 ***
I3        -3.2523184  0.4214880  -7.7163 8.247e-13 ***
I4        -1.9092089  0.4540848  -4.2045 4.137e-05 ***
I5        -1.8007260  0.4217865  -4.2693 3.184e-05 ***
I6        -1.5005958  0.3283248  -4.5705 9.076e-06 ***
I7        -0.5682738  0.1866835  -3.0440  0.002688 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Asset Price Bubbles

In the week 3 lectures you learnt that an *AR*(1) process,

$$y_t = a_0 + \varphi_1 y_{t-1} + \varepsilon_t$$

is stationary if $|\varphi_1| < 1$, and it is not stationary otherwise.

In the week 4 lectures we also considered the possibility of an *AR* unit root ($\varphi_1 = 1$) and discussed how to test for a unit root with the *ADF* test. In this test the hypotheses are

$$H_0 : \varphi_1 = 1 \quad vs \quad H_A : |\varphi_1| < 1$$

These hypotheses do not cover all possibilities since they ignore $\varphi_1 \leq -1$ and $\varphi_1 > 1$. Usually, this is not a problem because $|\varphi_1| > 1$ means that the process is explosive (a monotonic one if $\varphi_1 > 1$ or an oscillating one if $\varphi_1 < -1$), while $\varphi_1 = -1$ results in a random oscillation. These are rare scenarios in practice, and even if they occur, they are recognizable on time series plots.

Nevertheless, occasionally it is useful to perform formal hypothesis tests to find out whether a variable has an explosive pattern. In finance, for example, it is useful to study the possibility of asset price bubbles, i.e., situations when the actual price of an asset moves in excess of its market fundamental price.

A financial bubble is characterized by the rapid increase of the asset price to a point that is unsustainable, causing the asset to burst or contract in value. This scenario can be tested with a right-tail *ADF* test (*ADFrt*) that has the following hypotheses:

$$H_0 : \varphi_1 = 1 \quad vs \quad H_A : \varphi_1 > 1$$

The null hypothesis is the same as in the original left-tail *DF* test, but the alternative hypothesis is different.

Note that in this case the process is nonstationary under both hypotheses, under the null it is a random walk, while under the alternative it is explosive due to some bubble. The *ADFrt* test has the same test statistic as the original *ADF* test, but a different sampling distribution, and hence critical values.

The *ADFrt* test has two extensions. They have the same null hypothesis as the *ADF* and *ADFrt* tests, but different alternative hypotheses. Namely, the first extension, called supremum *ADF* (*SADF*) test, assumes under the alternative hypothesis that during the sample period there is a single collapsing bubble; while the second, called generalized supremum *ADF* (*GSADF*) test, assumes under the alternative hypothesis that during the sample period there are multiple periodically collapsing bubbles. Each of these tests consists of a set of *ADF* tests performed over some sequence of sub-samples and the *SADF* and *GSADF* test statistic value is the largest observed *ADF* test statistic.

Using *R*, these tests can be performed with the *adf.test()* function of the *tseries* package and the *radf()* function of the *exuber* package.

adf.test(y, alternative = , k =)

computes a left-tail (stationary) or a right-tail (explosive) ADF test for the null hypothesis of a unit root. It has three main arguments: y, alternative and k. y is a numeric vector or time series, alternative is "stationary" or "explosive", and k is the largest lag (i.e., lag length). If k is not specified by the user, it is set equal to trunc((length(y)-1)^(1/3))). The test regression always incorporates a constant and a linear trend.

radf(y, minw = , lag =)

performs the ADFrt, SADF and GSADF tests on variable y.[9] *It has three main arguments: y, minw and k. y is a numeric vector or time series, minw is a positive integer for the minimum window (sample) size, and lag is the largest lag in the ADF regression. If minw is not specified, it is set equal to (0.01+1.8/(T))T(0.01+1.8/(T))T*

Exercise 2

The *t6e2.xlsx* file contains monthly observations of the NASDAQ price index (*PR*) and the dividends (*DIV*) for the period January 1960 to December 1989. The present value model implies that the fundamental driver of the share price is the dividend payment.[10] A bubble occurs when the actual share price persistently deviates upward from the market fundamental price, which is the discounted present value of the stream of future cash flows attached to the asset.

[9] LK: This *R* function performs other recursive *ADF*-type tests as well, and *y* can also be a multivariate numeric time series object, but in this course we do not consider those options.
10 See slide 24 of the week 4 lecture notes.

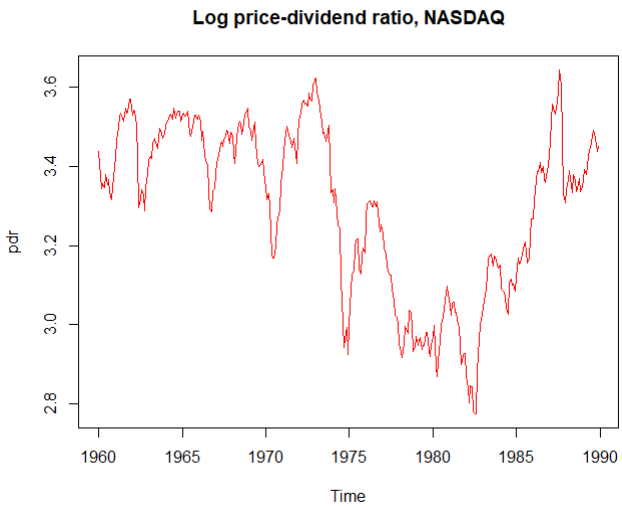
a) Launch *RStudio*, create a new project and script, and name both *t6e2*. Import the data and generate and plot the log price-dividend ratio,

$$pdr_t = \ln(PR_t) - \ln(DIV_t)$$

by executing the following commands:

```
attach(t6e2)
pdr = ts(log(PR) - log(DIV), start = c(1960, 1), end = c(1989, 12), frequency = 12)
plot.ts(pdr, main = "Log price-dividend ratio, NASDAQ", col = "red")
```

You get the following graph:



b) Perform the *ADF* and *ADFrt* tests on the log price-dividend ratio with *adf.test()* by setting its *alternative* argument to *stationary* and *explosive*, respectively.

If you do not have the *tseries* package on your computer, install it and then perform the following commands:

```
library(tseries)
adf.test(pdr, alternative = "stationary")
adf.test(pdr, alternative = "explosive")
```

They return the printouts shown on the next page. The last lines on these printouts tell that the first belongs to the original left-tail *ADF* test (*alternative hypothesis: stationarity*) and the second to the *ADFrt* test (*alternative hypothesis: explosive*).

The *p*-values are large (0.5229 and 0.4771), so both tests maintain the unit root null hypothesis.

```
        Augmented Dickey-Fuller Test

data:  pdr
Dickey-Fuller = -2.128, Lag order = 7, p-value = 0.5229
alternative hypothesis: stationary


        Augmented Dickey-Fuller Test

data:  pdr
Dickey-Fuller = -2.128, Lag order = 7, p-value = 0.4771
alternative hypothesis: explosive
```

c) Perform the *SADF* and *GSADF* test on the log price-dividend ratio.

The *ADF* and *ADFrt* test regressions had 7 lags. Let's use the same lag length in the *SADF* and *GSADF* tests.

Install the *exuber* package and then execute the following commands:

```
library(exuber)
RADF.pdr = radf(pdr, lag = 7)
summary(RADF.pdr)
```

They return the following printout:

```
── Summary (minw = 37, lag = 7) ──────────────── Monte Carlo (nrep = 2000) ──

series1 :
# A tibble: 3 × 5
  stat   tstat   `90`    `95`   `99`
  <fct>  <dbl>  <dbl>   <dbl> <dbl>
1 adf   -2.08  -0.399 -0.0745 0.560
2 sadf  -0.776  1.14   1.41   2.04
3 gsadf  1.86   1.90   2.14   2.60
```

The heading means that the minimum window size, i.e. the smallest number of observations used in the *ADF* tests, is 37; the lag length is 7; and the critical values were calculated with 2000 Monte Carlo simulations.[11] Below the heading you can see the *ADFrt*, *SADF* and *GSADF* test statistics and the 10%, 5% and 1% critical values.

Starting with the *ADFrt* test, you can see that the reported test statistic value is -2.08, different from the one returned by *adf.test()* in part (b). The discrepancy is due to the fact that *adf.test()* uses a constant and trend in the test regression, while *radf()* uses only a constant. In spite of this difference, this time none of them rejects the unit root null hypothesis.

[11] LK: If you do not know what Monte Carlo simulation is, do not worry, we do not have time to cover it.

Turning to the *SADF* and *GSADF* tests, you can see that the observed test statistic values (-0.776 and 1.86) are smaller than the critical values, so both tests maintain the null hypothesis of no bubble.

To see some details of the *SADF* test, execute

```
diagnostics(RADF.pdr, option = "sadf")
autoplot(RADF.pdr, option = "sadf", nonrejected = TRUE)
```

The first command confirms the decision drawn from the previous printout,

```
── Diagnostics (option = sadf) ──

series1:  Cannot reject H0
```

and the second displays a graph of the sequence of *ADF* test statistics (blue line) along with the 5% critical value (red dashed line).[12]

The observed *SADF* test statistic (-0.776) is at the highest point of the blue curve. The rejection region of the *SADF* and *GSADF* tests is under the right tail, so the fact that the blue line remains below the red dashed line indicates that the null hypothesis is maintained.

[12] 5% is the default significance level but it can be replaced with 10% or 1% by setting the optional *sig_lvl* argument equal to 90 or 99. Note that these values are not significance but the corresponding confidence levels.

Re-executing these commands for the *GSADF* test,

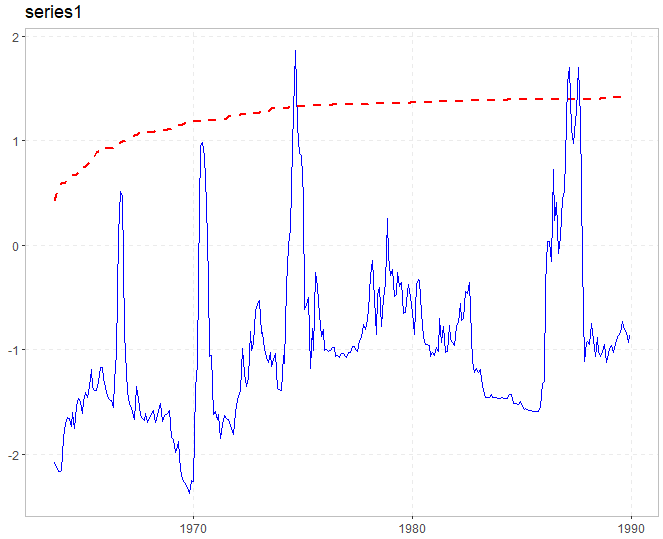
```
diagnostics(RADF.pdr, option = "gsadf")
autoplot(RADF.pdr, option = "gsadf", nonrejected = TRUE)
```

you get

```
── Diagnostics (option = gsadf) ──

series1:  Cannot reject H0
```

and



The statistical decision is the same as before (cannot reject H_0), though on the plot the blue curve moves above the red dashed curve three occasions. These explosive periods, however, were so short that they are not considered bubbles.