ECON90033 – QUANTITATIVE ANALYSIS OF FINANCE I

# TUTORIAL 4

---

*Read this handout and complete the tutorial exercises before your tutorial class so that you can ask for help during the tutorial if necessary.*

## Stationary Stochastic Processes

As you learnt on the week 3 lectures, we treat an observed time series as a particular realization of the underlying sequence of random variables, called stochastic process. Working with stochastic processes, it is usually crucial to know whether they are stationary or not. We distinguish two concepts of stationarity, the so-called strong or strict stationarity and weak or covariance stationarity.

A stochastic process is said to be strongly or strictly stationary if its joint probability distribution at times $t_1$, $t_2$, ..., $t_m$ is the same as at times $t_1+k$, $t_2+k$, ..., $t_m+k$, where $m > 1$ and $k$ are integers. In plain language this means that no matter when we observe the time series, its properties do not change.

The joint probability distribution of a stochastic process $\{y_t\}$, in general, is characterized by its central moments, i.e., by

$$E\left[\left(y - E(y)\right)^i\right] \quad , \quad i = 0,1,2,...$$

The first and second (central) moments, however, are sufficient to describe the properties of $\{y_t\}$ if it has a normal joint probability distribution or if $y_t$ is a linear combination of its own lags $\{y_{t-1}, y_{t-2}, \ldots\}$ and maybe of current and past values of some other processes. Although in practice these conditions are not necessarily satisfied, or at least we do not know whether they are met, for the sake of simplicity we usually consider only the first and second central moments. This leads us to the weak form of stationarity.

A stochastic process is weakly or covariance stationary if it has constant and finite unconditional means, and autocovariances that do not depend on time. In symbols, $\{y_t\}$ is weakly stationary if

$$E(y_t) = \mu < \infty$$
$$Cov(y_t, y_{t-k}) = E\left[(y_t - \mu)(y_{t-k} - \mu)\right] = \gamma_k < \infty$$

and the second requirement implies,

$$\rho_{y_t, y_{t-k}} = \frac{Cov(y_t, y_{t-k})}{\sqrt{Var(y_t) \times Var(y_{t-k})}} = \rho_{y_{t-k}, y_t} = \rho_{y_t, y_{t+k}} = \frac{\gamma_k}{\gamma_0} = \rho_k$$

meaning that the autocorrelation coefficients as well can depend on *k* at most.

Usually, stationarity is meant to be weak stationarity, unless told otherwise, and we follow this practice.

One of the simplest stationary processes is the white noise, which is a purely random series $\{\varepsilon_t\}$ characterized by the following properties:

$$E(\varepsilon_t) = 0 \quad , \quad Var(\varepsilon_t) = \sigma^2 \quad , \quad \rho_k = 0 \ \ (k > 0)$$

By default, a white noise does not follow any pattern.

In Ex 1 of the week 3 lectures I simulated a white noise. You are going to do the same in the first exercise.

**Exercise 1**

a) Simulate a white noise by drawing a random sample of 150 from the standard normal distribution. Name this series *eps.*

*The*

*rnorm(n, mean, sd)*

*function generates a vector of n pseudo random numbers drawn from a normal distribution with mean expected value and sd standard deviation. The default distribution is the standard normal distribution.*

*To make the simulated series reproducible, this function must be preceded with*

*set.seed(x)*

*which initializes the pseudo random number generator algorithm. In this formula x is an arbitrary integer and, combined with nrnd() and fixed mean and sd, it always produces the same sequence of random numbers.*

Launch *RStudio*, create a new project and script, and name both *t4e1*. Execute[1]

```
set.seed(11072023)
eps = ts(rnorm(150, mean = 0, sd = 1), start = 1)
```

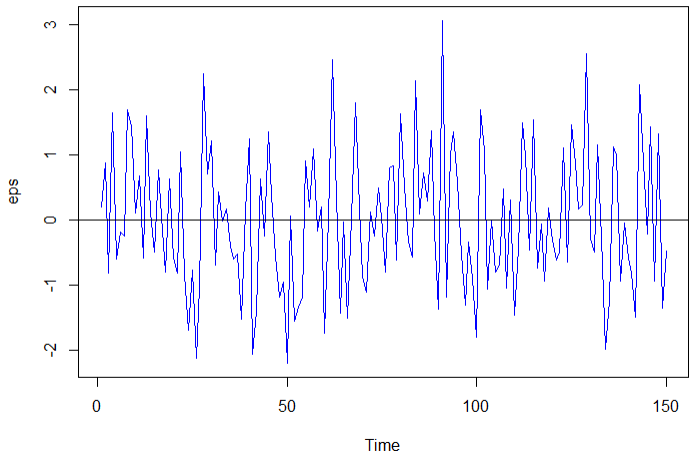
[1] To make sure that your results are exactly the same as mine, we use the same seed, 11072023. Otherwise each of us would draw a different set of random numbers. Also note that in the *rnorm()* function by default mean = 0 and sd = 1, so these arguments could be omitted this time.

b) By definition, a sequence of random numbers is stationary. Nevertheless, due to sampling error, finite samples might behave differently. To verify the properties of *eps*, first illustrate it with a time series plot.

The

```
ts.plot(eps, col = "blue")
abline(h = 0)
```

commands generate the following plot:




The simulated series appears to fluctuate randomly around zero with constant variance, and there is no obvious sign of autocorrelation either, so it looks like a white noise.

## The Autocorrelation and Partial Autocorrelation Functions

The autocorrelation and partial autocorrelation functions, *ACF* and *PACF*, are important tools for describing the dynamic properties of stochastic processes and for studying those of observed time series. They specify the relationships between the time gap between two different points in time on the one hand ($k$) and the corresponding autocorrelation and partial autocorrelation coefficients, respectively, on the other hand.

Assuming that $\{y_t\}$ is stationary, both its $k$th order autocorrelation and partial autocorrelation coefficients ($\rho_k$ and $\varphi_{k,k}$) show the direction and measure the strength of the linear relationship between $y_t$ and $y_{t-k}$. However, while $\rho_{t,t-k}$ does so without taking the intermediate lags (i.e., $y_{t-1}$, …, $y_{t-k+1}$) into consideration, $\varphi_{k,k}$ does account for them.
In terms of regression this means that $\rho_k$ is the slope parameter in the

$$y_t = \alpha + \rho_k y_{t-k} + \varepsilon_t$$

simple linear regression model, while $\varphi_{k,k}$ is the last slope parameter in the

$$y_t = \alpha + \varphi_{k,1} y_{t-1} + \varphi_{k,2} y_{t-2} + ... + \varphi_{k,k} y_{t-k} + \varepsilon_t$$

multiple linear regression model. These two population slope parameters are equal only if $y_{t-1}$, $y_{t-2}$, …, $y_{t-k}$ are perfectly uncorrelated, an unrealistic scenario in practice.

Returning to the $\{\varepsilon_t\}$ white noise, since by definition it is uncorrelated, its autocorrelation and partial autocorrelation functions are the same:

$$ACF: \ \rho_k = \begin{cases} 1 & if \quad k = 0 \\ 0 & if \quad k \geq 1 \end{cases} \qquad \text{and} \qquad PACF: \ \varphi_{kk} = \begin{cases} 1 & if \quad k = 0 \\ 0 & if \quad k \geq 1 \end{cases}$$

In practice, the *ACF* and *PACF* functions of the stochastic process that generated the data at hand are unknown, but assuming that the process is stationary and that the sample moments converge to the corresponding parameters (i.e., ergodicity), the autocorrelation and partial autocorrelation coefficients, and thus the *ACF* and *PACF* functions, can be estimated from a reasonably large sample of observations.

To obtain reasonable and reliable estimates of the autocorrelation and partial autocorrelation coefficients at various lags, the largest lag ($s$) should be relatively small compared to the sample size. There are several rules of thumb to this end. We are going to use $s = min(10, T/5)$ for non-seasonal data and $s = min(2S, T/5)$ for seasonal data, where $S$ is the number of seasons.

If the underlying data generating process is a white noise and the sample size is large, the sample autocorrelation and partial autocorrelation coefficients are approximately normally distributed,

$$r_k \sim N\left(0, \frac{1}{T}\right) \qquad \text{and} \qquad \hat{\varphi}_{kk} \sim N\left(0, \frac{1}{T}\right)$$

making possible to perform three hypothesis tests on them: the Bartlett's, Box-Pierce and Ljung-Box tests.

The Bartlett's test is for a single coefficient with the following hypotheses:

$$H_0 : \rho_k = 0 \ , \ H_A : \rho_k \neq 0 \qquad \text{and} \qquad H_0 : \varphi_{kk} = 0 \ , \ H_A : \varphi_{kk} \neq 0$$

The test statistics are

$$t = \sqrt{T}\, r_k \quad \text{and} \quad t = \sqrt{T}\, \hat{\varphi}_{kk}$$

Under $H_0$ they are distributed asymptotically as a $Z$ random variable, so the observed test statistics are to be compared to standard normal critical values.

The Box-Pierce and Ljung-Box tests are portmanteau tests for the first $k$ autocorrelation coefficients, meaning that they have a specific null hypothesis, but the alternative hypothesis comprises several scenarios. Namely,

$$H_0 : \rho_1 = \rho_2 = ... = \rho_k = 0 \qquad \text{and} \qquad H_A : \text{ at least one of } \rho_1, \rho_2, ..., \rho_k \neq 0$$

The test statistics are

$$Q_{BP} = T\sum_{k=1}^{s} r_k^2 \qquad \text{and} \qquad Q_{LB} = T(T+2)\sum_{k=1}^{s} \frac{r_k^2}{T-k}$$

where $T$ is the effective sample size. Under $H_0$ they are distributed as chi-square random variables with $s$ degrees of freedom when the test is performed on an observed time series or with $s$ - $m$ degrees of freedom when the test is performed on a residual series from a sample regression that has $m$ number of intercept and slope parameter estimates.

The sample autocorrelation function (*SACF*) can be used to assess the randomness and stationarity of a time series, to determine whether some trend and/or seasonal variations are present, and to identify moving average processes, while the sample partial autocorrelation function (*SPACF*) is more useful for autoregressive processes.[2]

*The functions*

*acf(x, lag.max = ) and pacf(x, lag.max = )*

*compute and by default plot the estimates of the autocorrelation and partial autocorrelation functions for the x numeric time series object. The second argument, lag.max is the lag length, i.e., the longest lag considered. If omitted, R sets it equal to 10log$_{10}$(T).*

**Exercise 1 (cont.)**

c) Calculate and plot *SACF* and *SPACF* of *eps*. Use the rule of thumb on the previous page for the lag length.

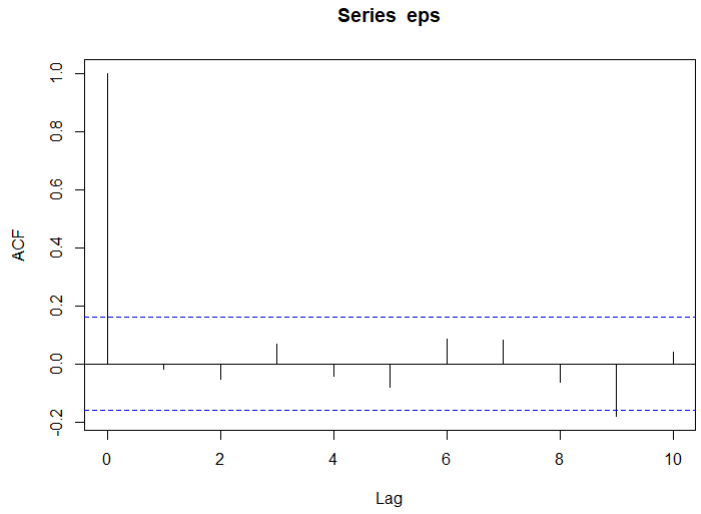
In this case $T$ = 150 and $s$ = *min*(10, $T$/5) = *min*(10, 30) = 10, so we are going to use 10 lags.[3]

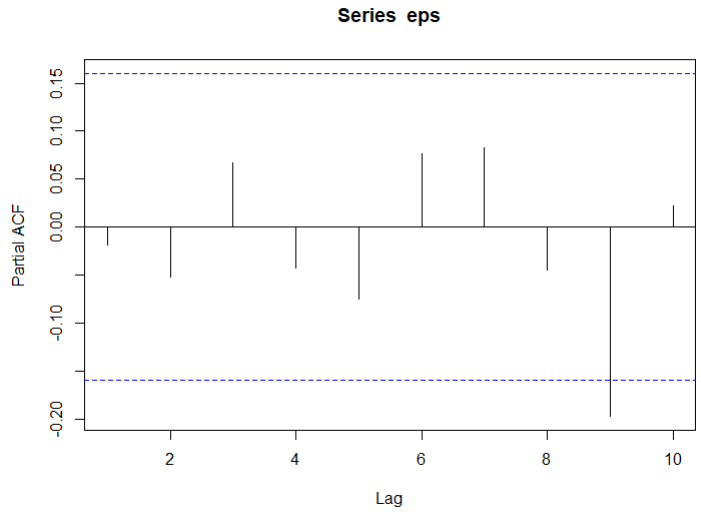
Execute the

```
acf(eps, lag.max = 10)
pacf(eps, lag.max = 10)
```

commands. They return the following plots of *SACF* and *SPACF*, called correlograms:

[2] We shall introduce moving average and autoregressive processes soon.
[3] You can easily confirm that this is far shorter than the default lag (10*log*$_{10}$(*T*) = 10*log*$_{10}$(150) = 21.76). In fact, since *T*/5 > 10 if *T* > 50, the rule of thumb suggests 10 lags whenever the sample size is above 50.




On these correlograms the blue dashed lines are the graphical representation of the approximate Bartlett's test. They depict

$$\pm\frac{2}{\sqrt{T}}=\pm\frac{2}{\sqrt{150}}=\pm 0.1632$$

and for a white noise the band between them is expected to contain 95% of the sample autocorrelation and partial autocorrelation coefficients (not accounting for $r_0$, which is by default equal to one). In other words, they illustrate the approximate 5% non-rejection region.

On each correlogram only the spike at lag 9 is outside the band, suggesting that $r_9$ and $\varphi_{99}$-hat are significantly different from zero. Although this suggests that at the 5% level there is significant autocorrelation and partial autocorrelation at lag 9, this can be contributed to type I error.

Otherwise, these correlograms basically confirm that the simulated *eps* series indeed behaves as a white noise.

d) Perform the Box-Pierce and Ljung-Box tests on *eps* at the 5% significance level.

*The BP and LB tests can be performed with the*

*Box.test(x, lag =, type =, fitdf = )*

*function, where x is a numeric vector or time series object, lag is the lag length, type is "Box-Pierce" or "Ljung-Box", and fitdf is the number of degrees of freedom to be subtracted if x is a series of residuals.*

For both tests the null hypothesis is that *eps* does not have 1st, 2nd, …, 10th order autocorrelation, i.e.,

$$H_0: \ \rho_1 = \rho_2 = ... = \rho_{10} = 0$$

while the alternative hypothesis is that *eps* has some autocorrelation of orders 1 – 10.

Since *eps* is just a simulated series and not a residual series, in this case *fitdf* = 0.

Execute

```
Box.test(eps, lag = 10, type = "Box-Pierce", fitdf = 0)
Box.test(eps, lag = 10, type = "Ljung-Box", fitdf = 0)
```

to get the following printouts:

```
        Box-Pierce test

data:  eps
X-squared = 10.174, df = 10, p-value = 0.4253


        Box-Ljung test

data:  eps
X-squared = 10.832, df = 10, p-value = 0.3708
```

Both *p*-values are quite large (> 0.10), so we can maintain the null hypothesis of no 1st, 2nd, …, 10th order autocorrelation.

**Exercise 2** (HMPY, p. 110, Ex 1)

In Exercise 2 of Tutorial 2 you studied monthly observations on US equity prices, dividends, earnings, consumer price index, and interest rate for the period January 1900 to September 2016. This time you will develop correlograms and perform hypothesis tests to further study their time series properties.

a) Launch *RStudio*, create a new project and script, and name both *t4e2*. Import the data set from *t2e2.RData* to your project, save it as *t4e2.RData*, and attach it to your *R* project. Convert the *PRICE* and *DIVIDEND* numeric vectors into *R* time series objects.

```
attach(t2e2)
PRICE = ts(PRICE, start = c(1900, 1), end = c(2016, 9), frequency = 12)
DIVIDEND = ts(DIVIDEND, start = c(1900, 1), end = c(2016, 9), frequency = 12)
```

Compute the percentage monthly log returns on equities and dividends, defined as

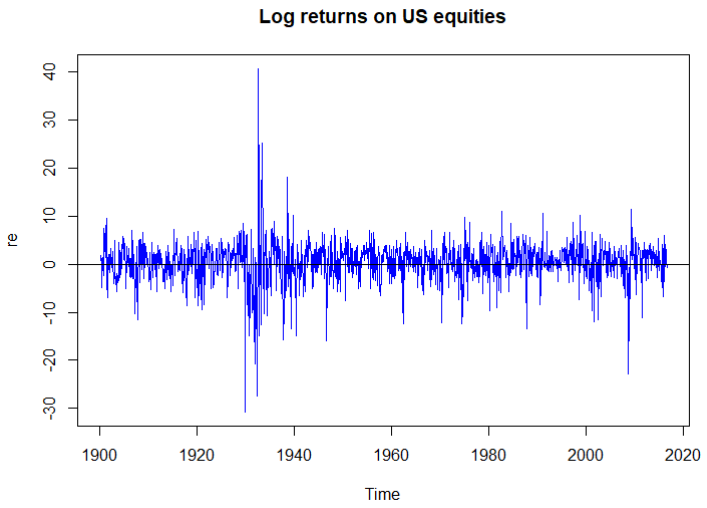
$$re_t = 100(\ln PRICE_t - \ln PRICE_{t-1})$$
$$rd_t = 100(\ln DIVIDEND_t - \ln DIVIDEND_{t-1})$$

by executing

```
re = 100*diff(log(PRICE))
rd = 100*diff(log(DIVIDEND))
```

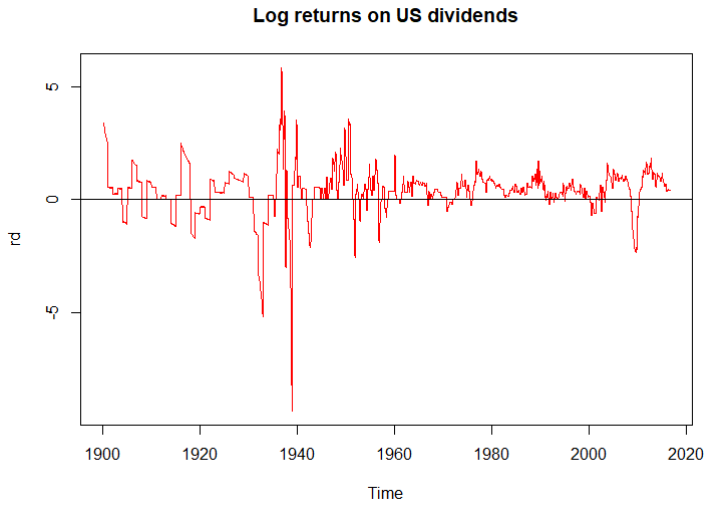
Plot these two series and interpret their time series patterns.

```
plot.ts(re, main = "Log returns on US equities", col = "blue")
abline(h = 0)
```



On this time series plot there is not any obvious sign of nonstationarity. In particular, the log returns on US equities tend to hover around some constant close to zero and, apart from some relatively large movements at around the start of the Great Depression in 1929, volatility seems to be steady.

```
plot.ts(rd, main = "Log returns on US dividends", col = "red")
abline(h = 0)
```




The log returns on US dividends also tend to hover around some constant. They seem to be less volatile in the first third of the sample than afterwards, but this is probably to some data manipulation.[4]

b) Compute and interpret the *SACF* and *SPACF* of equity returns and dividend returns for up to 6 lags.

The

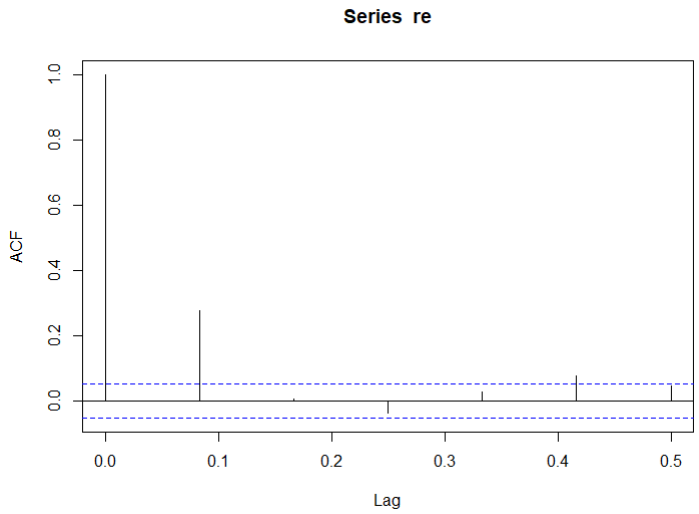
```
acf(re, lag.max = 6)
pacf(re, lag.max = 6)
```

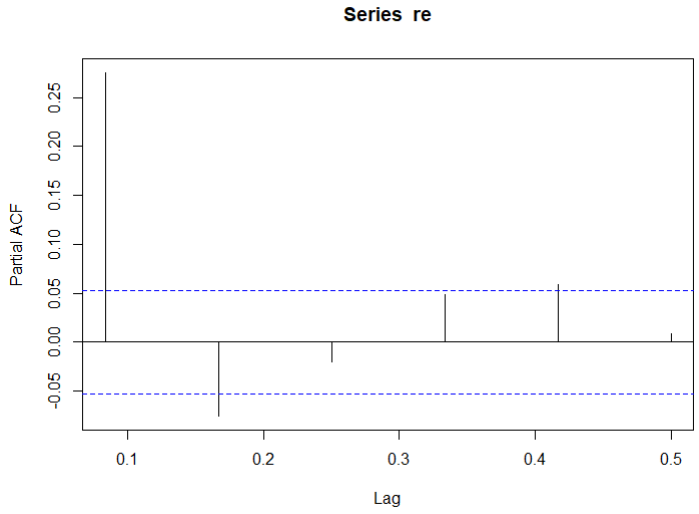
commands produce the correlograms displayed on the next page.

Have a look at the horizontal axes of these sample correlograms. Their scales are up 0.5. This might seem to be weird at the first glance, one would rather expect a scale from 0 or 1 to 6, as we intended to develop *SACF* and *SPACF* for up to 6 lags. The *acf()* and *pacf()*

[4] LK: I do not know for sure, but it seems to me that the pre-1939 monthly observations were obtained by interpolating quarterly observations.

functions, however, always measure the lags in terms of years, no matter what the actual data frequency is. Hence, if we had annual data, the scale would indeed be from 0 or 1 to 6. But we have monthly data, and one month is 1/12 year, so the scale is from 0 or 1/12 up to 6 / 12 = 0.5.



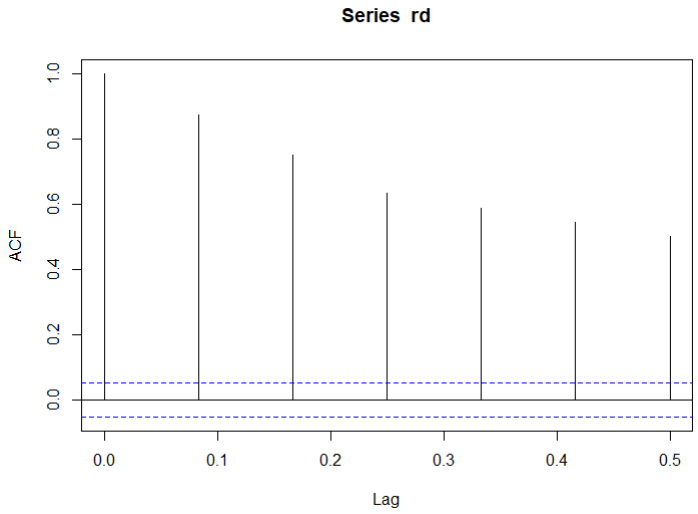



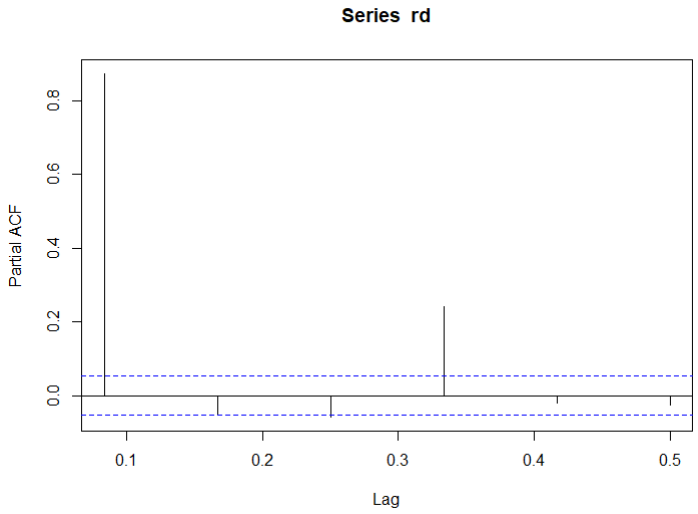

The *SACF* and *SPACF* of the log returns of US equities have a significant spike at 1 / 12, i.e., at the first lag. *SACF* also has a significant spike at lag 5, and *SPACF* at lags 2 and 5, but they are less important as they do not fit in a simple pattern.

The

*acf(rd, lag.max = 6)*
*pacf(rd, lag.max = 6)*

commands produce the following correlograms:




This time all six sample autocorrelation coefficients (lag = 1, 2, …, 6) are significant at the 5% level, and *SACF* has a seemingly exponentially decaying pattern. On the second correlogram the 1st and the 4th spikes are significant and the latter one might be related to the quarterly frequency of the time series.

c) Perform the Box-Pierce and Ljung-Box tests on equity returns and dividend returns for autocorrelation of orders 1-6 at the 5% significance level.

The null hypothesis is that there is not 1st, 2nd, …, 6th order autocorrelation, while the alternative hypothesis is that there is some autocorrelation of orders 1 to 6.

*Box.test(re, lag = 6, type = "Box-Pierce", fitdf = 0)*
*Box.test(re, lag = 6, type = "Ljung-Box", fitdf = 0)*

return

```
        Box-Pierce test

data:  re
X-squared = 120.68, df = 6, p-value < 2.2e-16


        Box-Ljung test

data:  re
X-squared = 120.98, df = 6, p-value < 2.2e-16
```

Both *p*-values are practically zero, so we reject the null hypothesis and conclude that the log returns of US equities have some autocorrelation of orders 1 to 6. Since we already acknowledged the significant spikes of the *SAFC*, this is a reasonable conclusion.

*Box.test(rd, lag = 6, type = "Box-Pierce", fitdf = 0)*
*Box.test(rd, lag = 6, type = "Ljung-Box", fitdf = 0)*

return

```
        Box-Pierce test

data:  rd
X-squared = 3666.7, df = 6, p-value < 2.2e-16


        Box-Ljung test

data:  rd
X-squared = 3679.4, df = 6, p-value < 2.2e-16
```

The reported *p*-values, and thus the statistical decisions too, are the same as before, so we conclude that the log returns of US dividends have some autocorrelation of orders 1 to 6. Again, this is logical in the light of the *SACF*.

## Univariate Time Series Models

On the week 3 lectures we introduced three dynamic models for stationary processes, the moving average (*MA*), the autoregressive (*AR*), and the mixed autoregressive moving average (*ARMA*) models.

1) A moving average model order *q*, *MA*(*q*), is defined by the following formulas:

$$y_t = \alpha + \theta_0\varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + ... + \theta_q\varepsilon_{t-q} = \sum_{i=0}^{q}\theta_i\varepsilon_{t-i}$$

$$\theta_0 \equiv 1 \ , \ \varepsilon_t \sim iid(0,\sigma^2)$$

The stochastic process described by this model has the following unconditional first and second moments:

$$E(y_t) = \alpha + E\left[\sum_{i=0}^{q}\theta_i\varepsilon_{t-i}\right] = \alpha + \sum_{i=0}^{q}\theta_i E(\varepsilon_{t-i}) = \alpha$$

$$Var(y_t) = Var\left[\sum_{i=0}^{q}\theta_i\varepsilon_{t-i}\right] = \sum_{i=0}^{q}\theta_i^2 Var(\varepsilon_{t-i}) = \sigma^2\sum_{i=0}^{q}\theta_i^2 = \gamma_0$$

$$Cov\,(y_t, y_{t-k}) = \gamma_k = \sigma^2\sum_{i=0}^{q-k}\theta_i\theta_{k+i} \text{ if } k \le q \text{ and } 0 \text{ if } k > q$$

Since these unconditional first and second moments do not depend on *t*, all finite-order *MA*(*q*) processes are stationary.

From the autocovariances the *ACF* of an *MA*(*q*) process is

$$\rho_k = \frac{\gamma_k}{\gamma_0} = \frac{\sum_{i=0}^{q-k}\theta_i\theta_{k+i}}{\sum_{i=0}^{q}\theta_i^2} \text{ if } k \le q \text{ and } 0 \text{ if } k > q$$

This formula shows that the autocorrelation coefficients are nonzero for lags $k \le q$, but they are zero for lags $k > q$. This implies that the memory of an *MA*(*q*) process is *q* periods long.

An *MA*(*q*) process is said to be invertible if it has an equivalent *AR*(∞) representation.

2) An autoregressive model of order *p*, *AR*(*p*), is defined by the following formulas:

$$y_t = \varphi_0 + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + ... + \varphi_p y_{t-p} + \varepsilon_t = \varphi_0 + \sum_{i=1}^{p}\varphi_i y_{t-i} + \varepsilon_t$$

$$\varepsilon_t \sim iid(0,\sigma^2)$$

The unconditional first and second moments are more complicated this time, but for *AR*(1), for example:

$$E(y) = \frac{\varphi_0}{1-\varphi_1} \ , \ \varphi_1 \neq 1$$

$$Var(y) = \gamma_0 = \frac{\sigma^2}{1-\varphi_1^2}$$

$$Cov\,(y_t, y_{t-k}) = \gamma_k = \varphi_1^k \frac{\sigma^2}{1-\varphi_1^2} \text{ granted that } |\varphi_1| < 1.$$

If this latter condition is satisfied, the *AR*(1) process is stationary.

From the autocovariances the *ACF* of an *AR*(1) process is

$$\rho_k = \frac{\gamma_k}{\gamma_0} = \varphi_1^k$$

This formula shows that the autocorrelation coefficients of a stationary *AR*(1) process are all nonzero and thus its *ACF* does not have a cut-off point, implying that the memory of this process is infinitely long. The *PACF* of a stationary *AR*(1) process, however, does have a cut-off point at lag 1 because $y_t$ depends directly only on $y_{t-1}$.

Moving on to the general case, *AR*(*p*) is stationary if its characteristic roots are all inside the unit circle.[5] The *ACF* of a stationary *AR*(*p*) process does not have a cut-off point, but its *PACF* does have one at lag *p*.

A stationary *AR*(*p*) process model has an equivalent *MA*(∞) representation.

3) A mixed autoregressive moving average model of orders *p* and *q*, *ARMA*(*p*, *q*), is defined by the following formulas:

$$y_t = \varphi_0 + \sum_{i=1}^{p} \varphi_i y_{t-i} + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

$$\varepsilon_t \sim iid(0, \sigma^2)$$

The underlying process is stationary and has an equivalent *MA*(∞) representation if its *AR* component is stationary, and it is invertible and has an equivalent *AR*(∞) representation if its *MA* component is invertible.

[5] The *AR* characteristic roots can be real or complex numbers and the unit circle is a circle in the real – complex coordinate system that is centered around the origin and has a unit radius. The moduli of the characteristic roots inside the unit circle are smaller than 1.

**Exercise 2 (cont.)**

d) Estimate an *AR*(2) model of equity returns

$$re_t = \varphi_0 + \varphi_1 re_{t-1} + \varphi_2 re_{t-2} + \varepsilon_t$$

and write out the estimated equation.

In principle, finite order *AR* models can be estimated with OLS like a standard multiple linear regression model. Using *R*, however, it is easier and more convenient to estimate *ARMA* and *ARIMA* models with a designated function, like the

*Arima(y, order = c(p,d,q), …)*

*function of the Forecast package, where y is the variable to be modelled, p is the order of the AR component, q is the order of the MA component, and d is the degree of differencing.*

In this case $p = 2$, $d = q = 0$, so after having installed the *Forecast* package on your computer, execute

```
library(forecast)
ar2ma0_re = Arima(re, c(2,0,0))
summary(ar2ma0_re)
```

You should get the following printout:[6]

```
Series: re
ARIMA(2,0,0) with non-zero mean

Coefficients:
         ar1      ar2    mean
      0.2958  -0.0751  0.4189
s.e.  0.0266   0.0266  0.1403

sigma^2 = 16.79:  log likelihood = -3959.51
AIC=7927.02   AICc=7927.04   BIC=7947.99
```

The top part of this printout shows the point estimates of the slope parameters of the autoregressive terms, *ar1* = $re_{t-1}$ and *ar2* = $re_{t-2}$, and the point estimate of 'mean'. But what is this 'mean'?

[6] This is just the top part of the printout; we ignore the bottom part for the time being.

To answer this question, it is important to understand that instead of estimating

$$re_t = \varphi_0 + \varphi_1 re_{t-1} + \varphi_2 re_{t-2} + \varepsilon_t$$

*Arima()* actually estimates

$$re_t = \tilde{\varphi}_0 + \eta_t$$
$$\eta_t = \varphi_1 \eta_{t-1} + \varphi_2 \eta_{t-2} + \varepsilon_t$$

where $\varepsilon_t$ is supposed to be a white noise, but $\eta_t$ is not as it is driven by an *AR*(2) process.

These two specifications are equivalent, but what is the relationship between them? Using the lag operator, the second equation can be solved for $\eta_t$ (assuming that $\varphi_1 + \varphi_2 \neq 1$),

$$\eta_t = \varphi_1 \eta_{t-1} + \varphi_2 \eta_{t-2} + \varepsilon_t \quad \rightarrow \quad (1 - \varphi_1 L - \varphi_2 L^2)\eta_t = \varepsilon_t \quad \rightarrow \quad \eta_t = \frac{\varepsilon_t}{1 - \varphi_1 L - \varphi_2 L^2}$$

The solution can be plugged in the first equation,

$$re_t = \tilde{\varphi}_0 + \eta_t = \tilde{\varphi}_0 + \frac{\varepsilon_t}{1 - \varphi_1 L - \varphi_2 L^2}$$

and finally the first equation can be rearranged as

$$(1 - \varphi_1 L - \varphi_2 L^2) re_t = (1 - \varphi_1 L - \varphi_2 L^2)\tilde{\varphi}_0 + \varepsilon_t \quad \rightarrow \quad re_t = \tilde{\varphi}_0 (1 - \varphi_1 - \varphi_2) + \varphi_1 re_{t-1} + \varphi_2 re_{t-2} + \varepsilon_t$$

Comparing this to the original *AR*(2) specification, you can see that the two specifications have the same lagged variables ($re_{t-1}$ and $re_{t-2}$), slope parameters ($\varphi_1$ and $\varphi_2$) and error term ($\varepsilon_t$). Hence, the intercepts must also be the same, implying

$$\varphi_0 = \tilde{\varphi}_0 (1 - \varphi_1 - \varphi_2)$$

Now we can write out the sample regression equation from the printout as follows:

$$\hat{\varphi}_0 = \hat{\tilde{\varphi}}_0 (1 - \hat{\varphi}_1 - \hat{\varphi}_2) = 0.4189(1 - 0.2958 + 0.0751) = 0.3264$$

and

$$\widehat{re}_t = \hat{\varphi}_0 + \hat{\varphi}_1 re_{t-1} + \hat{\varphi}_2 re_{t-2} = 0.3264 + 0.2958 re_{t-1} - 0.0751 re_{t-2}$$

e) Estimate an *MA*(1) model of equity returns

$$re_t = \alpha + \theta_1 \varepsilon_{t-1} + \varepsilon_t$$

and write out the estimated equation.

Unlike finite order *AR* models, *MA* and *ARMA* models cannot be estimated directly by OLS due to the unobservable lagged error term(s). Instead, they can be estimated in two steps using OLS or in one step using some nonlinear algorithm. The second solution is clearly more appealing and efficient, and that's what the *Arima()* function actually does.

In this case $p = d = 0$, $q = 1$, and the

*ar0ma1_re = Arima(re, c(0,0,1))*
*summary(ar0ma1_re)*

commands return

```
Series: re
ARIMA(0,0,1) with non-zero mean

Coefficients:
          ma1    mean
      0.2847  0.4192
s.e.  0.0246  0.1406

sigma^2 = 16.81:  log likelihood = -3960.82
AIC=7927.64   AICc=7927.66   BIC=7943.37
```

The sample regression equation can be obtained just like in part (d). Namely, this time *Arima()* estimated

$$re_t = \tilde{\varphi}_0 + \eta_t$$
$$\eta_t = \theta_1 \varepsilon_{t-1} + \varepsilon_t$$

Hence,

$$\eta_t = \theta_1 \varepsilon_{t-1} + \varepsilon_t = (1 + \theta_1 L)\varepsilon_t$$

and

$$re_t = \tilde{\varphi}_0 + \eta_t = \tilde{\varphi}_0 + (1 + \theta_1 L)\varepsilon_t = \tilde{\varphi}_0 + \theta_1 \varepsilon_{t-1} + \varepsilon_t \quad \rightarrow \quad \alpha = \tilde{\varphi}_0$$

Replacing the unobservable current and lagged error terms with the corresponding residuals, the sample regression is

$$\widehat{re}_t = \hat{\tilde{\varphi}}_0 + \hat{\theta}_1 e_{t-1} + e_t = 0.4192 + 0.2847 e_{t-1} + e_t$$

f) Consider the *ARMA*(1,1) model of equity returns

$$re_t = \varphi_0 + \varphi_1 re_{t-1} + \theta_1 \varepsilon_{t-1} + \varepsilon_t$$

Estimate this model and write out the estimated equation.

In this case $p = 1$, $d = 0$, $q = 1$, so execute

*ar1ma1_re = Arima(re, c(1,0,1))*
*summary(ar1ma1_re)*

to obtain the following printout:

```
Series: re
ARIMA(1,0,1) with non-zero mean

Coefficients:
          ar1     ma1    mean
      0.0843  0.2090  0.4181
s.e.  0.0818  0.0793  0.1445

sigma^2 = 16.81:  log likelihood = -3960.29
AIC=7928.59   AICc=7928.62   BIC=7949.56
```

In this case *Arima()* estimated

$$re_t = \tilde{\varphi}_0 + \eta_t$$
$$\eta_t = \varphi_1 \eta_{t-1} + \theta_1 \varepsilon_{t-1} + \varepsilon_t$$

From this (assuming that $\varphi_1 \neq 1$),

$$\eta_t = \varphi_1 \eta_{t-1} + \theta_1 \varepsilon_{t-1} + \varepsilon_t \quad \rightarrow \quad (1-\varphi_1 L)\eta_t = (1+\theta_1 L)\varepsilon_t \quad \rightarrow \quad \eta_t = \frac{1+\theta_1 L}{1-\varphi_1 L}\varepsilon_t$$

and

$$re_t = \tilde{\varphi}_0 + \eta_t = \tilde{\varphi}_0 + \frac{1+\theta_1 L}{1-\varphi_1 L}\varepsilon_t \quad \rightarrow \quad (1-\varphi_1 L)re_t = (1-\varphi_1 L)\tilde{\varphi}_0 + (1+\theta_1 L)\varepsilon_t$$

so

$$re_t = (1-\varphi_1)\tilde{\varphi}_0 + \varphi_1 re_{t-1} + \theta_1 \varepsilon_{t-1} + \varepsilon_t$$

Plugging in the point estimates, the sample regression equation is

$$\widehat{re}_t = \hat{\tilde{\varphi}}_0(1-\hat{\varphi}_1) + \hat{\varphi}_1 re_{t-1} + \hat{\theta}_1 e_{t-1} + e_t = 0.4181(1-0.0843) + 0.0843\widehat{re}_{t-1} + 0.2090 e_{t-1} + e_t$$
$$= 0.3829 + 0.0843\widehat{re}_{t-1} + 0.2090 e_{t-1} + e_t$$

g) You have estimated three models of equity returns. Given their printouts, which model do you think performs best?

This question can be answered on the basis of one of the model selection criteria reported on the printouts. They are the *Akaike Information Criterion* (*AIC*), the *Corrected Akaike Information Criterion* (*AICc*), and the *Bayesian information criterion* (*BIC*, also known as the *Schwartz Information Criterion, SIC*).

*AIC*, *AICc* and *BIC* can be used to rank alternative model specifications, granted that (a) each model has the same dependent variable and (b) is estimated from the same sample. These statistics are similar to the adjusted $R^2$ in the sense that they consider how well the models fit to the data and the cost in terms of the sample size and the number of parameters that have to be estimated from the sample, but there are two major differences. The first is that they weight

the goodness of fit, the sample size and the number of parameters differently, so they might rank the same specifications differently. The second is that while a higher adjusted $R^2$ is preferred to a smaller one, the decision rule is exactly the opposite for *AIC*, *AICc* and *BIC*, i.e., the smaller the better.

We do not discuss the details of these statistics, partly because *AIC*, *AICc* or *BIC* have several alternative definitions and formulas.[7] Fortunately, it does not matter which version of a given statistic is used, they all result in the same ranking. It is important to understand though that different software packages might use different formulas, so never compare an *AIC*, *AICc* or *BIC* statistic reported by one package to an *AIC*, *AICc* or *BIC* statistic reported by another package without checking the underlying formulas first.

In general, adjusted $R^2$ and *AIC* tend to favour larger models than *AICc* and *BIC*, especially when the sample size is small compared to the number of predictors in the model. *BIC* penalises the number of parameters more heavily than *AIC*, so the model chosen by *BIC* is either the same as that chosen by *AIC*, or a more parsimonious[8] one. The bottom line is that none of these statistics is superior under all conditions.

We have three competing specifications for the same dependent variable, $re_t$, and they were estimated from almost the same sample. Almost, because the lag length (i.e., the longest lag) of the *AR*(2) model is two while that of the *MA*(1) and *ARMA*(1,1) models is one. Since at each level of differencing one observation is sacrificed, this means that the effective sample size for the *AR*(2) model is one less than for the *MA*(1) and *ARMA*(1,1) models. This, however, most likely makes no real difference at all as we have 1400 $re_t$ observations.

Let's now compare the three printouts to each other. To make the comparison easier, the *AIC*, *AICc* or *BIC* statistics are summarised in the following table:

| | ***AR*(2)** | ***MA*(1)** | ***ARMA*(1,1)** |
|---|---|---|---|
| ***AIC*** | 7927.02 | 7927.64 | 7928.59 |
| ***AICc*** | 7927.04 | 7927.66 | 7928.62 |
| ***BIC*** | 7947.99 | 7943.37 | 7949.56 |

*AIC* and *AICc* favour the *AR*(2) specification, while *BIC* favours *MA*(1).

In this example you were told to estimate the *AR*(2), *MA*(1) and *ARMA*(1,1) models. In a real project, however, you have to decide yourselves which specification(s) might match best the unknown data generating process behind the observed time series. Traditionally, the decision

[7] Some of the alternative formulas might even alter the decision rule.
[8] In statistics a model is considered parsimonious if it accomplishes the desired level of explanation or prediction with as few predictor variables as possible.

is based on the correlograms, on the comparisons of the *SACF* and *SPACF* computed from the data to the *ACF*s and *PACF*s of various hypothetical data generating processes. This is, however, easier said than done because in practice *SACF* and *SPACF* often do not exhibit easily recognizable patterns, they do not help finding a suitable *ARMA*(*p*, *q*) model when *p* and *q* are both positive, and it takes a lot of practice to learn how to evaluate the correlograms.

Alternatively, one might experiment with a large number of different specifications and choose the one which is favoured by the favoured model selection criterion. There are several algorithms and computer programs that make this somewhat mechanical procedure easy to perform, like the *auto.arima()* function of the *Forecast* package.

*auto.arima(y, ic =, … )*

*where ic is one of "aic" (AIC), "aicc" (AICc) or "bic" (BIC), conducts a search over a set of possible ARIMA models*[9] *for a univariate time series and returns the best stationary and invertible model according to ic. Two further binary arguments are stepwise (if TRUE, a relatively fast stepwise selection is performed) and approximation (if TRUE, model selection is based on approximation). Hence, when this function is used for a large number of time series, the selection can be speed up by setting both of these arguments TRUE, otherwise it is better to set both FALSE. By default only the best model is reported, but if a third binary argument, trace, is set TRUE, the printout shows the full list of ARIMA models considered.*

h) Find the best model for *re* by running *auto.arima()* and minimizing *AICc*.

```
best.arima = auto.arima(re, ic = "aicc",
                        stepwise = FALSE, approximation = FALSE, trace = TRUE)
```

returns the printout displayed on the next page, where the last column shows the *AICc* values.

The algorithm considered 21 models with zero and non-zero mean alike, i.e., altogether 42 specifications. To make their comparisons easier, I coloured seven lines. The red one belongs to the *ARIMA*(3,0,2) = *ARMA*(3,2) model, which returned the smallest *AICc.* The blue ones belong to the specifications that rank 2nd, 3rd and 4th, and the green ones to the three specifications you estimated earlier.

To see some details of the best model, execute

*summary(best.arima)*

[9] We'll learn about *ARIMA* models soon.

```
ARIMA(0,0,0)            with zero mean     : 8052.615
ARIMA(0,0,0)            with non-zero mean : 8041.19
ARIMA(0,0,1)            with zero mean     : 7934.48
ARIMA(0,0,1)            with non-zero mean : 7927.657
ARIMA(0,0,2)            with zero mean     : 7934.46
ARIMA(0,0,2)            with non-zero mean : 7928.264
ARIMA(0,0,3)            with zero mean     : 7934.648
ARIMA(0,0,3)            with non-zero mean : 7927.84
ARIMA(0,0,4)            with zero mean     : 7936.627
ARIMA(0,0,4)            with non-zero mean : 7929.721
ARIMA(0,0,5)            with zero mean     : 7932.341
ARIMA(0,0,5)            with non-zero mean : 7926.256
ARIMA(1,0,0)            with zero mean     : 7938.549
ARIMA(1,0,0)            with non-zero mean : 7932.97
ARIMA(1,0,1)            with zero mean     : 7934.909
ARIMA(1,0,1)            with non-zero mean : 7928.617
ARIMA(1,0,2)            with zero mean     : 7935.643
ARIMA(1,0,2)            with non-zero mean : 7929.365
ARIMA(1,0,3)            with zero mean     : 7936.655
ARIMA(1,0,3)            with non-zero mean : 7929.825
ARIMA(1,0,4)            with zero mean     : 7932.001
ARIMA(1,0,4)            with non-zero mean : Inf
ARIMA(2,0,0)            with zero mean     : 7933.796
ARIMA(2,0,0)            with non-zero mean : 7927.045
ARIMA(2,0,1)            with zero mean     : 7935.713
ARIMA(2,0,1)            with non-zero mean : 7928.835
ARIMA(2,0,2)            with zero mean     : 7935.375
ARIMA(2,0,2)            with non-zero mean : 7928.209
ARIMA(2,0,3)            with zero mean     : 7935.837
ARIMA(2,0,3)            with non-zero mean : 7928.954
ARIMA(3,0,0)            with zero mean     : 7935.55
ARIMA(3,0,0)            with non-zero mean : 7928.509
ARIMA(3,0,1)            with zero mean     : 7937.072
ARIMA(3,0,1)            with non-zero mean : 7930.007
ARIMA(3,0,2)            with zero mean     : 7927.344
ARIMA(3,0,2)            with non-zero mean : 7916.296
ARIMA(4,0,0)            with zero mean     : 7933.445
ARIMA(4,0,0)            with non-zero mean : 7927.286
ARIMA(4,0,1)            with zero mean     : 7928.106
ARIMA(4,0,1)            with non-zero mean : 7923.948
ARIMA(5,0,0)            with zero mean     : 7929.565
ARIMA(5,0,0)            with non-zero mean : 7924.368
```

It returns

```
Series: re
ARIMA(3,0,2) with non-zero mean

Coefficients:
         ar1      ar2     ar3      ma1     ma2    mean
      1.2695  -1.2466  0.2862  -0.9905  0.9372  0.4309
s.e.  0.0327   0.0354  0.0263   0.0220  0.0343  0.1489

sigma^2 = 16.62:  log likelihood = -3951.11
AIC=7916.22   AICc=7916.3   BIC=7952.93
```

If you compare this printout to the previous ones, you can see that *ARMA*(3,2) indeed has the smallest *AICc* and also *AIC*, but its *BIC* is actually larger than those of the *AR*(2), *MA*(1) and *ARMA*(1,1) models.

g) Correctly specified *ARMA* models should have white noise residuals. Check whether the *ARMA*(3,2) model selected by *auto.arima()* satisfies this requirement.

Recall that a white noise has zero mean, constant variance and zero autocorrelation at all non-zero lags. Since there is an intercept (non-zero mean) in the model, only the second and third requirements are important. They can be checked with the *checkresiduals()* function of the *Forecast* package.

*checkresiduals(object, lag =, test = )*

*where object can be a previously estimated time series model or a residual series, lag is the number of lags to use in the portmanteau autocorrelation test, test is either "LB" (Ljung-Box) or "BG" (Breusch-Godfrey), checks whether the residuals from a time series model look like white noise.*

Execute

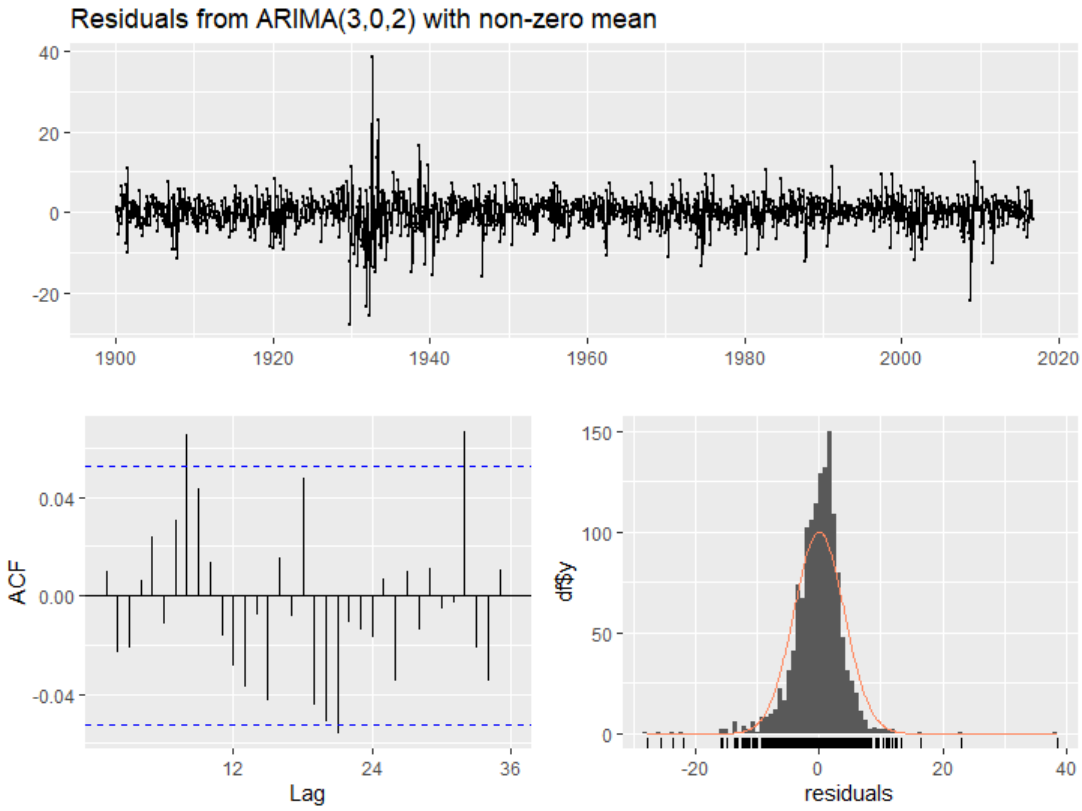
*checkresiduals(best.arima, lag = 10, test = "LB")*

You should get the printouts shown on the next page.[10] At the 5% significance level there is autocorrelation, so the error term might not be a white noise.

Although this means that the dynamics in the observed time series is not 'perfectly' captured with an *ARMA* model, we should not reject the *ARMA*(3,2) model automatically, it might still be our best option to model $re_t$.

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

[10] LK: By now you should be able to evaluate these printouts, so I skip the details this time. Note, however, that you cannot do the same in the assignments and on the exam.




```
        Ljung-Box test

data:  Residuals from ARIMA(3,0,2) with non-zero mean
Q* = 12.867, df = 5, p-value = 0.02466

Model df: 5.   Total lags used: 10
```