# Lecture 9: ARIMA models Pt 1 (Chapter 8)

## Forecasting: principles and practice (2nd edition)

book by Rob Hyndman and George Athanasopoulos
Parts of the slides are taken from the lecture notes of Robert Nau, Duke University
(https://people.duke.edu/~rnau/411home.htm)

```r
library(fpp2)
library(tseries)
```

## Review

### What ARIMA stands for:

- A series that need to be differenced to made stationary is an “integrated” (I) series.
- Lags of stationarized series are called “autoregressive” (AR) terms
- Lags of the forecast errors are called “moving average” (MA) terms

## Construction of an ARIMA model for forecasting

1. Plot the data. Identify any unusual observations.
2. If necessary, transform the data (using a Box-Cox transformation) to stabilize the variance.
3. If the data are non-stationary: take first differences of the data until the data are stationary.
4. Examine the ACF/PACF: Is an AR($p$) or MA($q$) model appropriate?
5. Try your chosen model(s), and use the AICc to search for a better model.
6. Check the residuals from your chosen model by plotting the ACF of the residuals, and doing a portmanteau test of the residuals. If they do not look like white noise, try a modified model.
7. Once the residuals look like white noise, calculate forecasts.

## ARIMA terminology

A non-seasonal ARIMA model can be (almost) completely summarized by three numbers:

p = the number of autoregressive terms

d = the number of nonseasonal differences

q = the number of moving-average terms

- This is called an "ARIMA(p,d,q)" model
- The model may also include a constant term (or not)

## Stationarity

A stationary time series is one whose properties do not depend on the time at which the series is observed.

So time series with trends, or with seasonality, are not stationary.

Some cases can be confusing – a time series with cyclic behaviour (but not trend or seasonality) is stationary. That is because the cycles are not of fixed length, so before we observe the series we cannot be sure where the peaks and troughs of the cycles will be.

In general, a stationary time series will have no obvious observable patterns in the long-term.

## Continued from Lecture 8

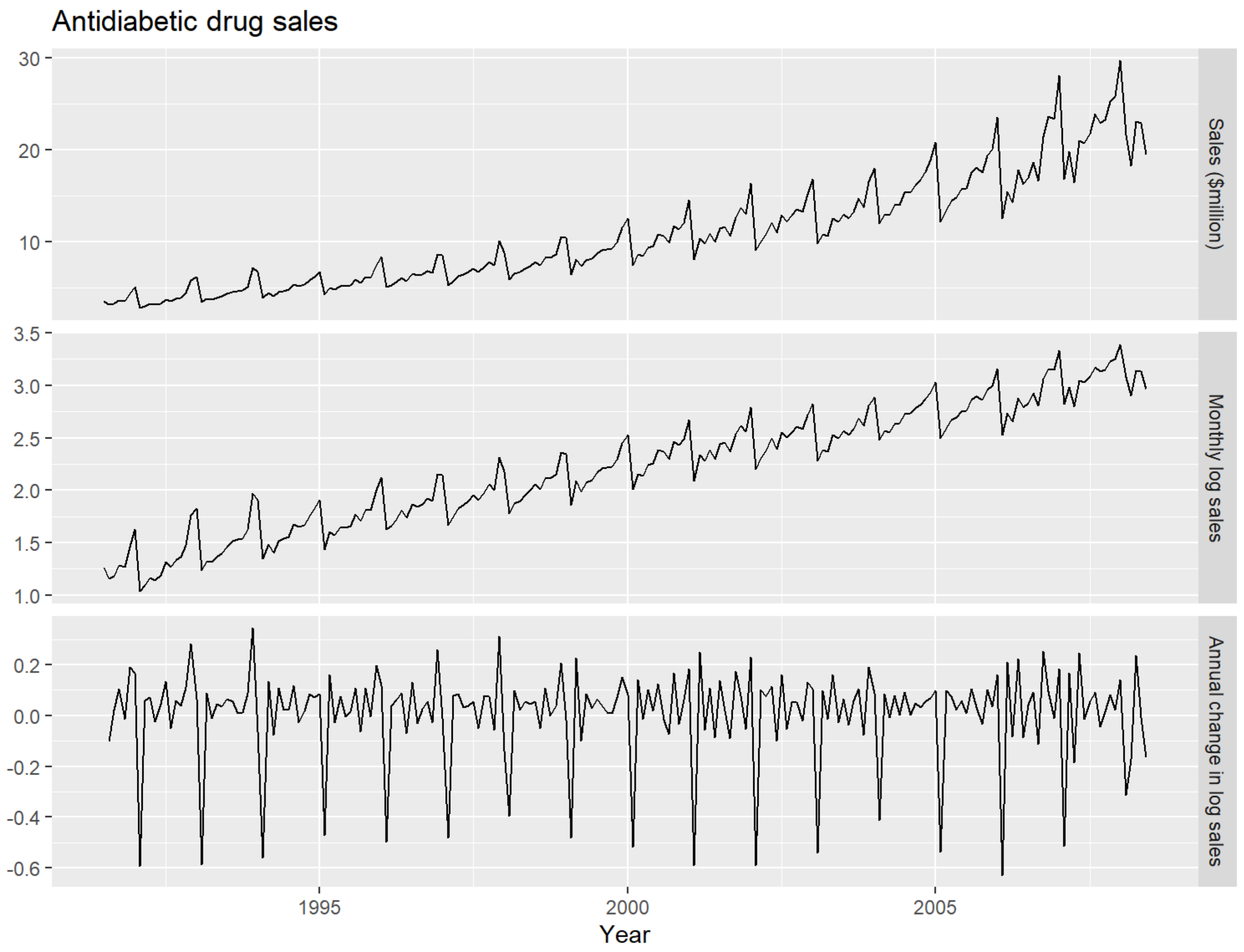
### Differencing

One way to make a time series stationary is to compute the differences between consecutive observations. This is known as differencing.

Transformations such as logarithms can help to stabilize the variance of a time series.

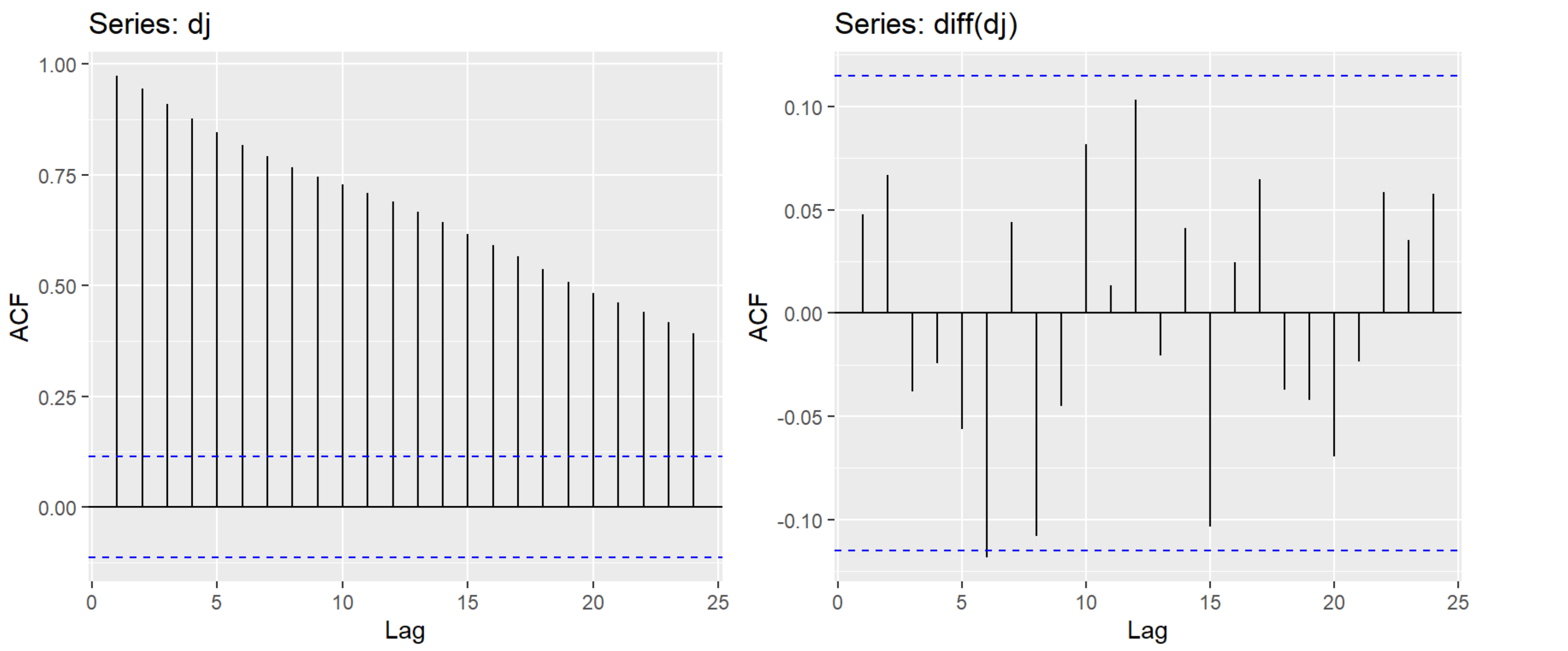
Differencing can help stabilize the mean of a time series by removing changes in the level of a time series, and eliminating or reducing the trend and the seasonality.

```r
cbind("Sales ($million)" = a10,
"Monthly log sales" = log(a10),
"Annual change in log sales" = diff(log(a10),1)) %>%
autoplot(facets=TRUE) +
xlab("Year") + ylab("") +
ggtitle("Antidiabetic drug sales")
```


Antidiabetic drug sales
30
20
10
Sales ($million)
3.5
3.0
2.5
2.0
1.5
1.0
Monthly log sales
0.2
0.0
-0.2
-0.4
-0.6
Annual change in log sales
1995
2000
2005
Year

Besides looking at the time plot of the data, the ACF plot is also useful for identifying non-stationary time series. ACF of Dow in levels and diffs

```r
library(gridExtra)
grid.arrange(ggAcf(dj), ggAcf(diff(dj)), ncol=2)
```



For a stationary time series, the ACF will drop to zero relatively quickly, while the ACF of non-stationary data decreases slowly. Also, for non-stationary data, the value of autocorrelation coefficient $r_1$ is often large and positive.

```r
Box.test(diff(dj), lag=10, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  diff(dj)
## X-squared = 14.461, df = 10, p-value = 0.153
```

The ACF of the differenced Dow-Jones index looks just like that of a white noise series. There is only one autocorrelation lying just outside the 95% limits, and the Ljung-Box $Q^*$ statistic has a p-value of 0.153 (for $h = 10$). This suggests that the daily change in the Dow-Jones index is essentially a random amount which is uncorrelated with that of previous days.

## Random walk model

The differenced series is the change between consecutive observations in the original series, and can be written as

$$y'_t = y_t - y_{t-1}.$$

The differenced series will have only $T - 1$ values since it is not possible to calculate a difference $y'_1$ for the first observation.

When the differenced series is white noise, the model for the original series can be written as

$$y_t - y_{t-1} = e_t \quad \text{or} \quad y_t = y_{t-1} + e_t\ .$$

A random walk model is very widely used for non-stationary data, particularly financial and economic data. Random walks typically have:

- long periods of apparent trends up or down
- sudden and unpredictable changes in direction.

The forecasts from a random walk model are equal to the last observation, as future movements are unpredictable, and are equally likely to be up or down. Thus, the random walk model underpins naive forecasts.

A closely related model allows the differences to have a non-zero mean. Then

$$y_t - y_{t-1} = c + e_t \quad \text{or} \quad y_t = c + y_{t-1} + e_t \ .$$

The value of $c$ is the average of the changes between consecutive observations. If $c$ is positive, then the average change is an increase in the value of $y_t$. Thus $y_t$ will tend to drift upwards. But if $c$ is negative, $y_t$ will tend to drift downwards.

This is the model behind the drift method.

## Second-order differencing

Occasionally the differenced data will not appear stationary and it may be necessary to difference the data a second time to obtain a stationary series:

$$\begin{aligned} y''_t &= y'_t - y'_{t-1} \\ &= (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) \\ &= y_t - 2y_{t-1} + y_{t-2}. \end{aligned}$$

In this case, $y''_t$ will have $T-2$ values. Then we would model the *change in the changes* of the original data. In practice, it is almost never necessary to go beyond second-order differences.

## Seasonal differencing

A seasonal difference is the difference between an observation and the corresponding observation from the previous year. So

$$y'_t = y_t - y_{t-m} \qquad \text{where } m = \text{number of seasons.}$$

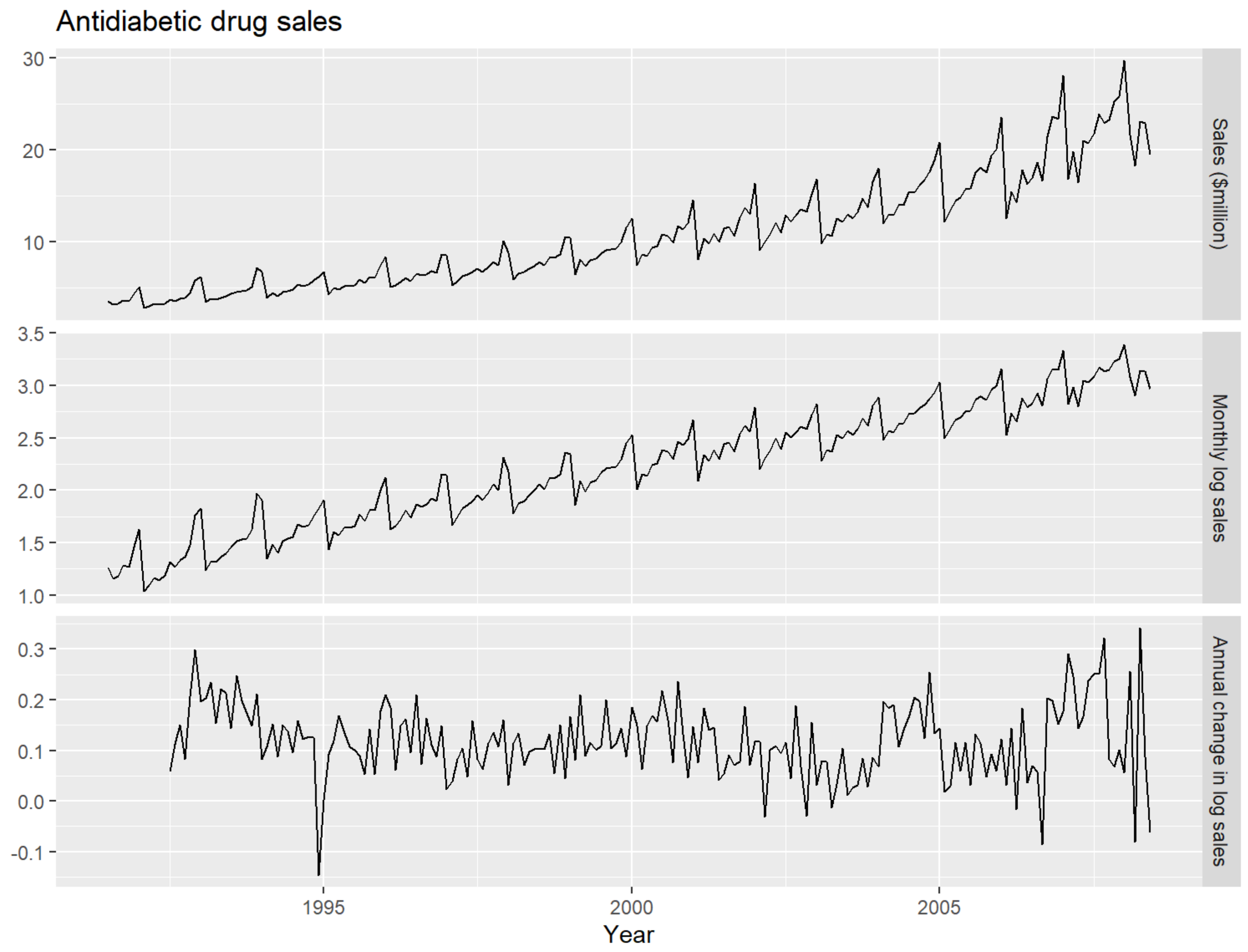
These are also called *lag-$m$ differences* as we subtract the observation after a lag of $m$ periods.

If seasonally differenced data appear to be white noise, then an appropriate model for the original data is

$$y_t = y_{t-m} + e_t.$$

Forecasts from this model are equal to the last observation from the relevant season. That is, this model gives seasonal naive forecasts.

```
cbind("Sales ($million)" = a10,
"Monthly log sales" = log(a10),
"Annual change in log sales" = diff(log(a10),12)) %>%
autoplot(facets=TRUE) +
xlab("Year") + ylab("") +
ggtitle("Antidiabetic drug sales")
```

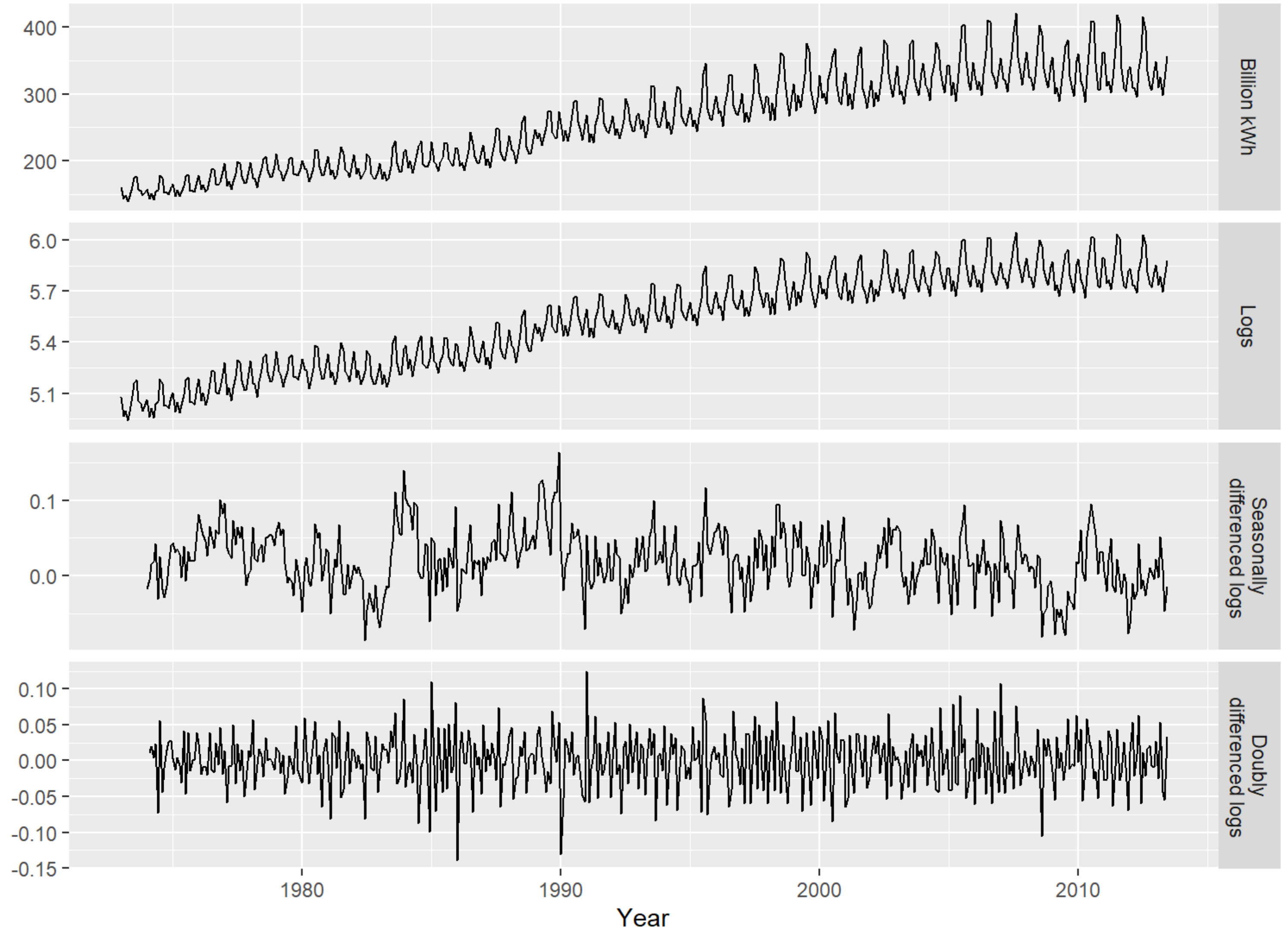


Logs and seasonal differences of the A10 (antidiabetic) sales data. The logarithms stabilize the variance, while the seasonal differences remove the seasonality and trend.

Sometimes it is necessary to do both a seasonal difference and a first difference to obtain stationary data, as is shown in figure in the next slide. Here, the data are first transformed using logarithms (second panel), then seasonal differences are calculated (third panel). The data still may seem a little non-stationary, and so further first differences are computed (bottom panel).

```r
cbind("Billion kWh" = usmelec,
"Logs" = log(usmelec),
"Seasonally\n differenced logs" = diff(log(usmelec),12),
"Doubly\n differenced logs" = diff(diff(log(usmelec),12),1)) %>%
autoplot(facets=TRUE) +
xlab("Year") + ylab("") +
ggtitle("Monthly US net electricity generation")
```

Monthly US net electricity generation



There is a degree of subjectivity in selecting which differences to apply. The seasonally differenced data in Figure on “Antidiabetic drug sales” do not show substantially different behaviour from the seasonally differenced data in Figure “Monthly US net electricity generation”.

In the latter case, we could have decided to stop with the seasonally differenced data, and not done an extra round of differencing. In the former case, we could have decided that the data were not sufficiently stationary and taken an extra round of differencing. Some formal tests for differencing will be discussed later, but there are always some choices to be made in the modelling process, and different analysts may make different choices.

If $y'_t = y_t - y_{t-m}$ denotes a seasonally differenced series, then the twice-differenced series is

$$\begin{aligned} y''_t &= y'_t - y'_{t-1} \\ &= (y_t - y_{t-m}) - (y_{t-1} - y_{t-m-1}) \\ &= y_t - y_{t-1} - y_{t-m} + y_{t-m-1}\,. \end{aligned}$$

Or

If $y'_t = y_t - y_{t-1}$ denotes first differenced series, then the twice-(seasonally) differenced series is

$$\begin{aligned} y''_t &= y'_t - y'_{t-m} \\ &= (y_t - y_{t-1}) - (y_{t-m} - y_{t-m-1}) \\ &= y_t - y_{t-1} - y_{t-m} + y_{t-m-1}\,. \end{aligned}$$

When both seasonal and first differences are applied, it makes no difference which is done first the result will be the same.

However, if the data have a strong seasonal pattern, it is recommended that seasonal differencing be done first because sometimes the resulting series will be stationary and there will be no need for a further first difference. If first differencing is done first, there will still be seasonality present.

It is important that if differencing is used, the differences are interpretable. First differences are the change between one observation and the next. Seasonal differences are the change between one year to the next. Other lags are unlikely to make much interpretable sense and should be avoided.

## Unit root tests

One way to determine more objectively if differencing is required is to use a unit root test. These are statistical hypothesis tests of stationarity that are designed for determining whether differencing is required.

The usual hypothesis tests for regression coefficients do not work when the data are non-stationary.

A number of unit root tests are available, and they are based on different assumptions and may lead to conflicting answers.

## The ADF test

One of the most popular tests is the Augmented Dickey-Fuller (ADF) test.

For this test, the following regression model is estimated:

$$y'_t = \alpha + \beta t + \phi y_{t-1} + \beta_1 y'_{t-1} + \beta_2 y'_{t-2} + \cdots + \beta_k y'_{t-k},$$

where $y'_t$ denotes the first-differenced series, $y'_t = y_t - y_{t-1}$ and $k$ is the number of lags to include in the regression.

If the original series, $y_t$, needs differencing, then the coefficient $\hat{\phi}$ should be approximately zero. If $y_t$ is already stationary, then $\hat{\phi} < 0$.

The following R function from the tseries package carries out the ADF test

```
adf.test(x, alternative = "stationary")
```

In R, the default value of $k$ is set to $\lfloor (T-1)^{1/3} \rfloor$, where $T$ is set to the length of the time series $\lfloor x \rfloor$ means the largest integer not greater than $x$.

The null-hypothesis for an ADF test is that the data are non-stationary. So large p-values are indicative of non-stationarity, and small p-values suggest stationarity. Using the usual 5% threshold, differencing is required if the p-value is greater than 0.05.

Details of the adf.test (along with other commands in tseries package) could be found by downloading (https://cran.r-project.org/web/packages/tseries/tseries.pdf)

One of the main criticisms of the ADF test is its low power when $\phi$ is less than but close to $0$, i.e., $1 - \phi = 0.95$

## The KPSS test

Another popular unit root test is the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test. This reverses the hypotheses, so the null-hypothesis is that the data are stationary. In this case, small p-values (e.g., less than 0.05) suggest that differencing is required. The kpss.test is also from the tseries package:

```
kpss.test(x)
```

## Backshift notation

The backward shift operator $B$ is a useful notational device when working with time series lags:

$$By_t = y_{t-1}.$$

(Some references use $L$ for lag instead of $B$ for backshift.) In other words, $B$, operating on $y_t$, has the effect of shifting the data back one period. Two applications of $B$ to $y_t$ shifts the data back two periods:

$$B(By_t) = B^2 y_t = y_{t-2}.$$

For monthly data, if we wish to consider *the same month last year*, the notation is $B^{12} y_t$ = $y_{t-12}$.

The backward shift operator is convenient for describing the process of differencing. A first difference can be written as

$$y'_t = y_t - y_{t-1} = y_t - By_t = (1 - B)y_t \; .$$

Note that a first difference is represented by $(1 - B)$. Similarly, if second-order differences have to be computed, then:

$$y''_t = y_t - 2y_{t-1} + y_{t-2} = (1 - 2B + B^2)y_t = (1 - B)^2 y_t \; .$$

In general, a $d$th-order difference can be written as

$$(1 - B)^d y_t.$$

Backshift notation is very useful when combining differences as the operator can be treated using ordinary algebraic rules. In particular, terms involving $B$ can be multiplied together.

For example, a seasonal difference followed by a first difference can be written as

$$\begin{aligned}(1-B)(1-B^m)y_t &= (1-B-B^m+B^{m+1})y_t \\ &= y_t - y_{t-1} - y_{t-m} + y_{t-m-1},\end{aligned}$$

the same result we obtained earlier.

## Autoregressive models

In a multiple regression model, we forecast the variable of interest using a linear combination of predictors. In an autoregression model, we forecast the variable of interest using a linear combination of past values of the variable.

The term autoregression indicates that it is a regression of the variable against itself.

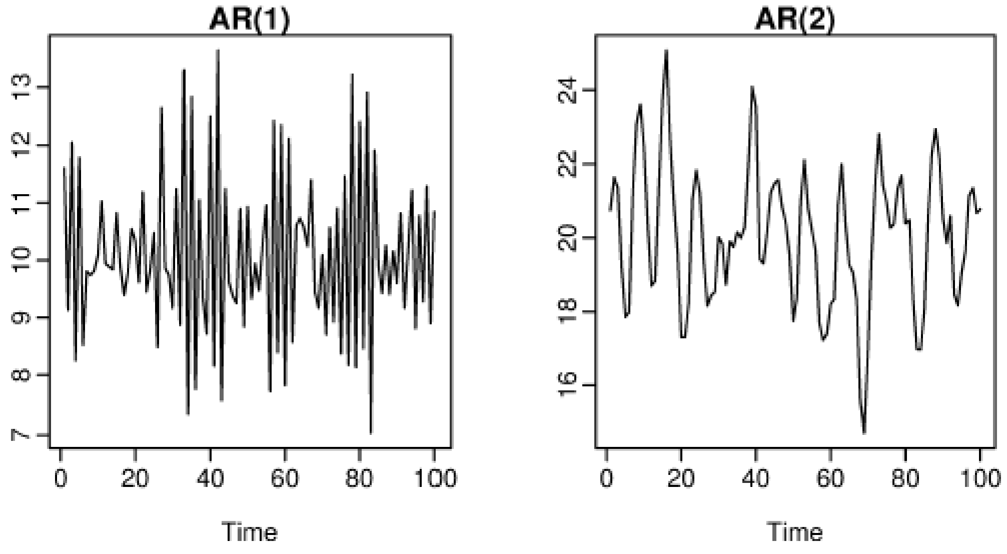
An autoregressive model of order $p$, denoted AR($p$) model, can be written as

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t,$$

where $c$ is a constant and $e_t$ is white noise. This is like a multiple regression but with lagged values of $y_t$ as predictors.

Autoregressive models are remarkably flexible at handling a wide range of different time series patterns. The two series in the next slide show series from an AR(1) model and an AR(2) model. Changing the parameters $\phi_1 \dots \phi_p$ results in different time series patterns. The variance of the error term $e_t$ will only change the scale of the series, not the patterns.

Left: AR(1) with $y_t = 18 - 0.8y_{t-1} + e_t$. Right: AR(2) with $y_t = 8 + 1.3y_{t-1} - 0.7y_{t-2} + e_t$. $e_t$ is normally distributed white noise with mean zero and variance one.



***

For an AR(1) model:

- When $\phi_1 = 0$, $y_t$ is equivalent to white noise.
- When $\phi_1 = 1$ and $c = 0$, $y_t$ is equivalent to a random walk.
- When $\phi_1 = 1$ and $c \neq 0$, $y_t$ is equivalent to a random walk with drift
- When $\phi_1 < 0$, $y_t$ tends to oscillate between positive and negative values.

We normally restrict autoregressive models to stationary data, in which case some constraints on the values of the parameters are required. For example,

- For an AR(1) model: $-1 < \phi_1 < 1$
- For an AR(2) model: $-1 < \phi_1 < 1, \phi_1 + \phi_2 < 1$

When $p \geq 3$, the restrictions are much more complicated. R takes care of these restrictions when estimating a model

## Interpretation of AR terms

- A series displays autoregressive (AR) behavior if it apparently feels a “restoring force” that tends to pull it back toward its mean.
- In an AR(1) model, the AR(1) coefficient determines how fast the series tends to return to its mean. If the coefficient is near zero, the series returns to its mean quickly; if the coefficient is near 1, the series returns to its mean slowly.
- In a model with 2 or more AR coefficients, the sum of the coefficients determines the speed of mean reversion, and the series may also show an oscillatory pattern.

## Moving average models

Rather than use past values of the forecast variable in a regression, a moving average model uses past forecast errors in a regression-like model.

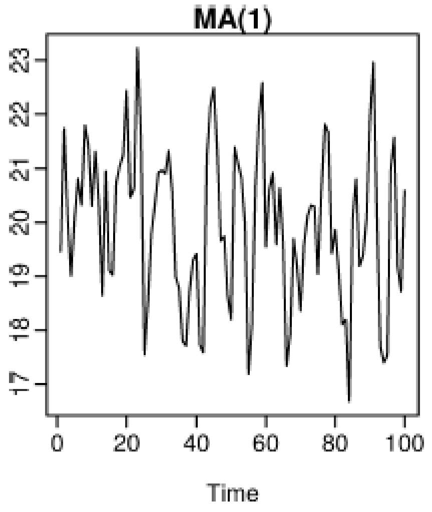
$$y_t = c + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q},$$

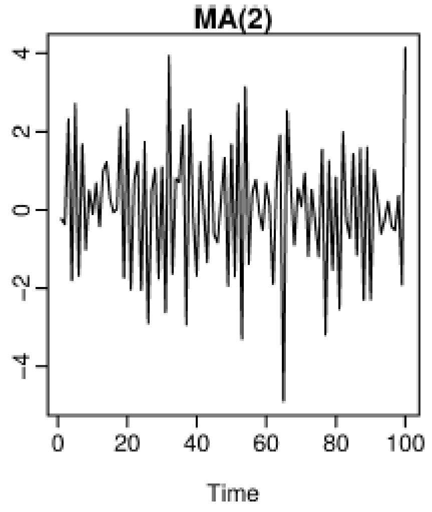
where $e_t$ is white noise. We refer to this as an MA($q$) model.

We do not observe the values of $e_t$, so it is not really regression in the usual sense.

Moving average models should not be confused with moving average smoothing. A moving average model is used for forecasting future values while moving average smoothing is used for estimating the trend-cycle of past values.

Left: MA(1) with $y_t = 20 + e_t + 0.8e_{t-1}$. Right: MA(2) with $y_t = e_t - e_{t-1} + 0.8e_{t-2}$. $e_t$ is normally distributed white noise with mean zero and variance one.





It is possible to write any stationary AR($p$) model as an MA($\infty$) model. For example, using repeated substitution, we can demonstrate this for an AR(1) model:

$$\begin{aligned}
y_t &= \phi_1 y_{t-1} + e_t \\
&= \phi_1(\phi_1 y_{t-2} + e_{t-1}) + e_t \\
&= \phi_1^2 y_{t-2} + \phi_1 e_{t-1} + e_t \\
&= \phi_1^3 y_{t-3} + \phi_1^2 e_{t-2} + \phi_1 e_{t-1} + e_t \\
&\text{etc.}
\end{aligned}$$

Provided $-1 < \phi_1 < 1$, the value of $\phi_1^k$ will get smaller as $k$ gets larger. So eventually we obtain

$$y_t = e_t + \phi_1 e_{t-1} + \phi_1^2 e_{t-2} + \phi_1^3 e_{t-3} + \cdots,$$

an MA($\infty$) process.

The reverse result holds if we impose some constraints on the MA parameters. Then the MA model is called "invertible". That is, we can write any invertible MA(q) process as an AR($\infty$) process. Invertible models are not simply introduced to enable us to convert from MA models to AR models. They also have some mathematical properties that make them easier to use in practice.

The invertibility constraints are similar to the stationarity constraints.

- For an MA(1) model: $-1 < \theta_1 < 1$
- For an MA(2) model: $-1 < \theta_1 < 1, \theta_1 + \theta_2 < 1$

When $q \geq 3$, the restrictions are much more complicated.

If you are curious, there are many good references including (https://eml.berkeley.edu/~powell/e241b_f06/TS-StatInv.pdf)

R takes care of these restrictions when estimating a model

## Interpretation of MA terms

- A series displays moving-average (MA) behavior if it apparently undergoes random “shocks” whose effects are felt in two or more consecutive periods.
- The MA(1) coefficient is (minus) the fraction of last period’s shock that is still felt in the current period.
- The MA(2) coefficient, if any, is (minus) the fraction of the shock two periods ago that is still felt in the current

## Do we need both AR and MA terms?

- In general, we don’t: usually it suffices to use only one type or the other.
- Some series are better fitted by AR terms, others are better fitted by MA terms (at a given level of differencing).
- Rough rules of thumb:

–If the stationarized series has positive autocorrelation at lag 1, AR terms often work best. If it has negative autocorrelation at lag 1, MA terms often work best.

– An MA(1) term often works well to fine-tune the effect of a nonseasonal difference, while an AR(1) term often works well to compensate for the lack of a nonseasonal difference, so the choice between them may depend on whether a difference has been used.

(Note: Robert Nau from Duke has an informative website on ARIMA (https://people.duke.edu/~rnau/411home.htm))

## Non-seasonal ARIMA models

If we combine differencing with autoregression and a moving average model, we obtain a non-seasonal ARIMA model. ARIMA is an acronym for AutoRegressive Integrated Moving Average model (*integration* in this context is the reverse of differencing). The full model can be written as

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t,$$

where $y'_t$ is the differenced series (it may have been differenced more than once). The *predictors* on the right hand side include both lagged values of $y_t$ and lagged errors.

We call this an *ARIMA($p, d, q$) model*, where $p =$ order of the autoregressive part; $d =$ degree of first differencing involved; $q =$ order of the moving average part.

Once we start combining components in this way to form more complicated models, it is much easier to work with the backshift notation.

$$\begin{aligned}\underset{\text{AR}(p)}{\overset{\downarrow}{}}\ \ \underbrace{(1-\phi_1 B-\cdots-\phi_p B^p)}_{}\ &\ \ (1-B)^d y_t\\ &= c+(1+\theta_1 B+\cdots+\theta_q B^q)e_t\end{aligned}$$

$$\begin{array}{ccc} \text{AR}(p) & & d\text{ diffs} \\ \downarrow & & \downarrow \\ (1-\phi_1 B-\cdots-\phi_p B^p) & & (1-B)^d y_t \\ & = & c+(1+\theta_1 B+\cdots+\theta_q B^q)e_t \\ & & \uparrow \\ & & \text{MA}(q) \end{array}$$

R uses a slightly different parametrization:

$$(1-\phi_1 B-\cdots-\phi_p B^p)({y'}_t-\mu)=(1+\theta_1 B+\cdots+\theta_q B^q)e_t$$

where ${y'}_t=(1-B)^d y_t$ and $\mu$ is the mean of ${y'}_t$ so to convert the bottom equation to the top equation, we move $(1-\phi_1 B-\cdots-\phi_p B^p)(-\mu)$ to the right hand of the equality in bottom equation. Hence we have $c=(1-\phi_1 B-\cdots-\phi_p B^p)\mu$

The same stationarity and invertibility conditions that are used for autoregressive and moving average models also apply to this ARIMA model.

Selecting appropriate values for $p$, $d$ and $q$ can be difficult. The `auto.arima()` function in R will do it for us automatically. Later we will learn how to use *auto.arima()* and also some methods to choose the methods ourselves.

First, we try to select values for $p$, $d$ and $q$ as outlined in “Constructing an ARIMA model”

## Constructing non-seasonal ARIMA model manually

1. Plot the data. Identify any unusual observations.
2. If necessary, transform the data (using a Box-Cox transformation) to stabilize the variance.
3. If the data are non-stationary: take first differences of the data until the data are stationary.
4. Examine the ACF/PACF: Is an AR($p$) or MA($q$) model appropriate?
5. Try your chosen model(s), and use the AICc to search for a better model.
6. Check the residuals from your chosen model by plotting the ACF of the residuals, and doing a portmanteau test of the residuals. If they do not look like white noise, try a modified model.
7. Once the residuals look like white noise, calculate forecasts.

## ACF and PACF plots

It is usually not possible to tell, simply from a time plot, what values of p and q are appropriate for the data. However, it is sometimes possible to use the ACF plot, and the closely related PACF plot, to determine appropriate values for $p$ and $q$.

Recall that an ACF plot shows the autocorrelations which measure the relationship between $y_t$ and $y_{t-k}$ for different values of $k$. Autocorrelation coefficients of a time series $y$ could be calculated and plotted in R with

```
Acf (y); ggAcf(y);
```

Now if $y_t$ and $y_{t-1}$ are correlated, then $y_{t-1}$ and $y_{t-2}$ must also be correlated. But then $y_t$ and $y_{t-2}$ might be correlated, simply because they are both connected to $y_{t-1}$.

The *partial autocorrelations* measure the relationship between $y_t$ and $y_{t-k}$ after removing the effects of other time lags: $1, 2, 3, \dots, k-1$.

So the first partial autocorrelation is identical to the first autocorrelation, because there is nothing between them to remove.

The partial autocorrelation of the $k^{th}$ lag could be *approximated* by running OLS regression up to the $k^{th}$ lag as follows:

$$\begin{aligned}\alpha_k &= k\text{th partial autocorrelation coefficient}\\ &= \text{the estimate of } \phi_k \text{ in the autoregression model}\\ y_t &= c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_k y_{t-k} + e_t.\end{aligned}$$

Chapter 4 of Hamilton, J. D. (1994) (Time Series Analysis) discusses the proof based on Durbin (1960).

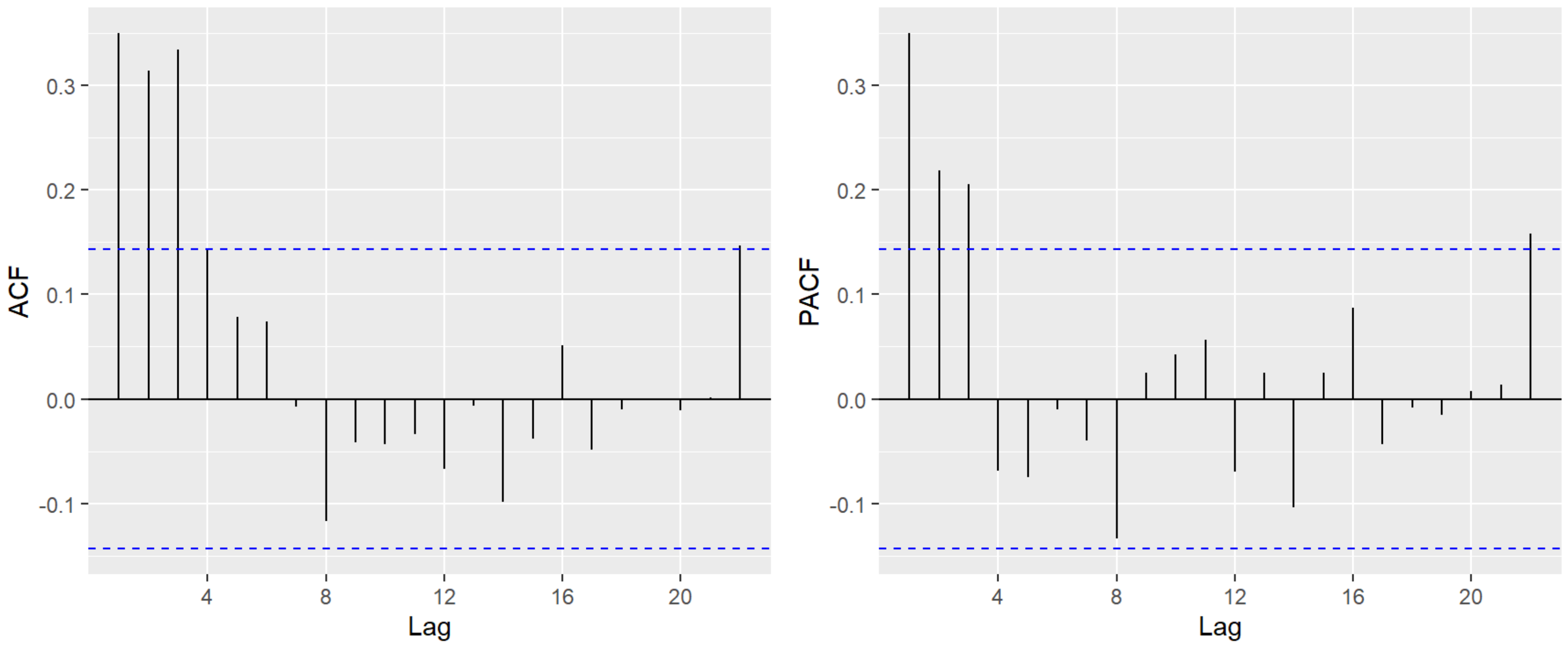
A step by step calculation of PACF at $k^{th}$ lag is shown in (https://www.empiwifo.uni-freiburg.de/lehre-teaching-1/winter-term/dateien-financial-data-analysis/handout-pacf.pdf)

PACF could be calculated and plotted for time serie $y$ in R:

```
Pacf (y); ggPacf(y);
```

The partial autocorrelations have the same critical values of $\pm 1.96/\sqrt{T}$ as for ordinary autocorrelations, and these are shown on the plot

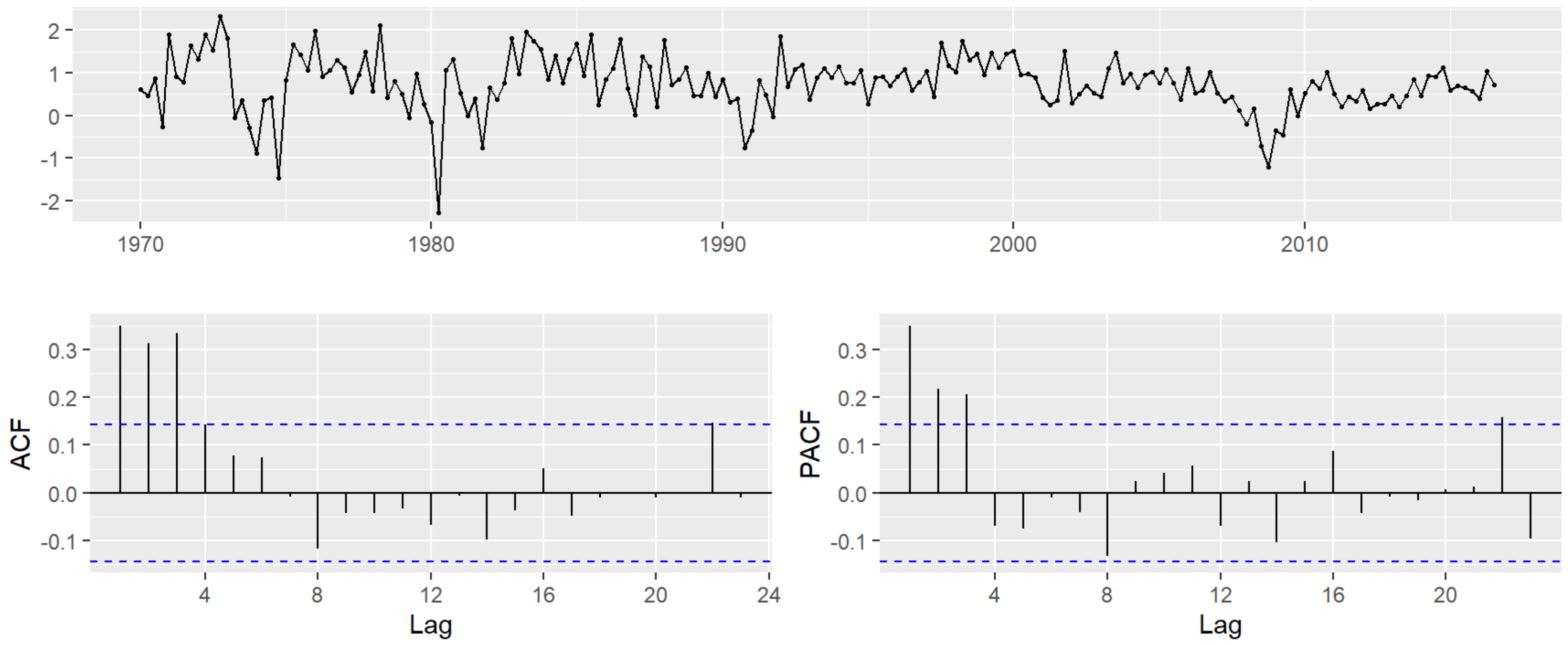
```
grid.arrange(ggAcf(uschange[,"Consumption"],main=""), ggPacf( uschange[,"Consumption"],main=""), ncol=2)
```



ACF and PCF of quarterly percentage change in US consumption.

## or a time plot, ACF plot and PACF plot in one command:

```
ggtsdisplay(uschange[,"Consumption"],main="")
```



If the data are from an ARIMA($p$, $d$,0) or ARIMA(0,$d$,$q$) model, then the ACF and PACF plots can be helpful in determining the value of $p$ or $q$. If $p$ and $q$ are both positive, then the plots do not help in finding suitable values of $p$ and $q$.

The data may follow an ARIMA($p, d, 0$) model if the ACF and PACF plots of the differenced data show the following patterns:

- the ACF is exponentially decaying or sinusoidal;
- there is a significant spike at lag $p$ in PACF, but none beyond lag $p$.

The data may follow an ARIMA($0, d, q$) model if the ACF and PACF plots of the differenced data show the following patterns:

- the PACF is exponentially decaying or sinusoidal;
- there is a significant spike at lag $q$ in ACF, but none beyond lag $q$.

From the ACF and PACF plots of US consumption, we see:

- ACF : there are three spikes followed by an almost significant spike at lag 4 (apart from one just outside the bounds at lag 22). We can ignore one significant spike in each plot if it is just outside the limits, and not in the first few lags. After all, the probability of a spike being significant by chance is about one in twenty, and we are plotting 22 spikes in each plot.

- PACF, there are three significant spikes, and then no significant spikes thereafter

We can ignore one significant spike in each plot if it is just outside the limits

The pattern in the first three spikes is what we would expect from an ARIMA(3,0,0), as the PACF tends to decrease.

## Maximum likelihood estimation

Once the model order has been identified (i.e., the values of $p$, $d$ and $q$), we need to estimate the parameters $c$, $\phi_1, \dots, \phi_p$, $\theta_1, \dots, \theta_q$.

When R estimates the ARIMA model, it uses *maximum likelihood estimation* (MLE).

This technique finds the values of the parameters which maximize the probability of obtaining the data that we have observed.

For ARIMA models, MLE is very similar to the *least squares* estimates that would be obtained by minimizing

$$\sum_{t=1}^{T} e_t^2.$$

## Information Criteria

Akaike's Information Criterion (AIC), which was useful in selecting predictors for regression, is also useful for determining the order of an ARIMA model. It can be written as

$$\text{AIC} = -2log(L) + 2(p + q + k + 1),$$

where $L$ is the likelihood of the data, $k = 1$ if $c = 0$ that the last term in the parenthesis is the number of parameters in the model (including $\sigma^2$, the variance of the residuals)

The corrected Akaike's IC for ARIMA:

$$\text{AIC}_\text{c} = \text{AIC} + \frac{2(p + q + k + 1)(p + q + k + 2)}{T - p - q - k - 2}.$$

The Bayesian IC

$$\text{BIC} = \text{AIC} + [log(T) - 2](p + q + k + 1).$$

Good models are obtained by minimizing the AIC, AICc or BIC. Our preference is to use the AICc.

It is important to note that these information criteria tend not to be good guides to selecting the appropriate order of differencing ( ) of a model, but only for selecting the values of $p$ and $q$.

This is because the differencing changes the data on which the likelihood is computed, making the AIC values between models with different orders of differencing not comparable.

So we need to use some other approach to choose $d$, and then we can use the AICc to select $p$ and $q$.

## ARIMA modelling in R

If you want to choose the model yourself, use the `Arima()` function in R.

```
fit2 <- Arima(uschange[,"Consumption"], order=c(3,0,0))
summary(fit2)
```

```
## Series: uschange[, "Consumption"]
## ARIMA(3,0,0) with non-zero mean
##
## Coefficients:
##          ar1     ar2     ar3    mean
##       0.2274  0.1604  0.2027  0.7449
## s.e.  0.0713  0.0723  0.0712  0.1029
##
## sigma^2 estimated as 0.3494:  log likelihood=-165.17
## AIC=340.34   AICc=340.67   BIC=356.5
##
## Training set error measures:
##                       ME      RMSE       MAE      MPE     MAPE      MASE
## Training set 0.001106571 0.5847286 0.4294193 47.28068 171.0212 0.6728123
##                    ACF1
## Training set 0.01499451
```

If we are unsure if another specification may give a better fit, we can run *Arima()* on the alternative model and compare AICc. For example, our analysis may lead us to consider ARIMA$(2,0,2)$.

```
fit21 <- Arima(uschange[,"Consumption"], order=c(2,0,2))
summary(fit21)
```

```
## Series: uschange[, "Consumption"]
## ARIMA(2,0,2) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ma1     ma2    mean
##       1.3908  -0.5813  -1.1800  0.5584  0.7463
## s.e.  0.2553   0.2078   0.2381  0.1403  0.0845
##
## sigma^2 estimated as 0.3511:  log likelihood=-165.14
## AIC=342.28   AICc=342.75   BIC=361.67
##
## Training set error measures:
##                        ME     RMSE       MAE      MPE    MAPE      MASE
## Training set 0.0007456313 0.584581 0.432925 49.21967 170.739 0.678305
##                     ACF1
## Training set 0.009876658
```

Hence, AICc is smaller for ARIMA$(3, 0, 0)$ than ARIMA$(2, 0, 2)$.

Now we try the *auto.arima()* function. Make sure to make the arguments "work harder" by specfiying *stepwise=FALSE* and *approximate=FALSE*. Otherwise, *auto.arima()* may miss some ARIMA specifications for a quick approximation.

The `auto.arima()` function in R uses a variation of the Hyndman and Khandakar algorithm, which combines unit root tests, minimization of the AICc and MLE to obtain an ARIMA model.

## *auto.arima()*

```
fit3 <- auto.arima(uschange[,"Consumption"], seasonal=FALSE,
stepwise=FALSE, approximation=FALSE)
summary(fit3)
```

```
## Series: uschange[, "Consumption"]
## ARIMA(3,0,0) with non-zero mean
##
## Coefficients:
##          ar1     ar2     ar3    mean
##       0.2274  0.1604  0.2027  0.7449
## s.e.  0.0713  0.0723  0.0712  0.1029
##
## sigma^2 estimated as 0.3494:  log likelihood=-165.17
## AIC=340.34   AICc=340.67   BIC=356.5
##
## Training set error measures:
##                       ME      RMSE       MAE      MPE     MAPE      MASE
## Training set 0.001106571 0.5847286 0.4294193 47.28068 171.0212 0.6728123
##                    ACF1
## Training set 0.01499451
```

## How does auto.arima() work?

Hyndman-Khandakar algorithm for automatic ARIMA modelling

1. The number of differences $0 \leq d \leq 2$ is determined using repeated KPSS tests.
2. The values of $p$ and $q$ are then chosen by minimizing the AICc after differencing the data $d$ times.

Rather than considering every possible combination of $p$ and $q$, the algorithm uses a stepwise search to traverse the model space.

1. Four initial models are fitted:
   - *ARIMA* $(0, d, 0)$ ,
   - *ARIMA* $(2, d, 2)$,
   - *ARIMA* $(1, d, 0)$,
   - *ARIMA* $(0, d, 1)$.

   A constant is included unless $d = 2$. If $d \leq 1$, an additional model-ARIMA without a constant is also fitted.

2. The best model (with the smallest AICc value) fitted in step (a) is set to be the “current model”.

3. Variations on the current model are considered:

- vary $p$ and/or $q$ from the current model by $\pm 1$;
- include/exclude $c$ from the current model. The best model considered so far (either the current model or one of these variations) becomes the new current model.

4. Repeat Step 2(c) until no lower AICc can be found.

The arguments to auto.arima() provide for many variations on the algorithm.

The default procedure uses some approximations to speed up the search. These approximations can be avoided with the *argument approximation=FALSE*. It is possible that the minimum AICc model will not be found due to these approximations, or because of the use of a stepwise procedure. A much larger set of models will be searched if the argument *stepwise=FALSE* is used. See the help file for a full description of the arguments (https://cran.r-project.org/web/packages/forecast/forecast.pdf).

Additional notes:

The auto.arima() function automates the inclusion of a constant.

The auto.arima() function automates the differentiation of integrated series and do not select a model with roots close to the unit circle.

## Recall: Modelling procedure

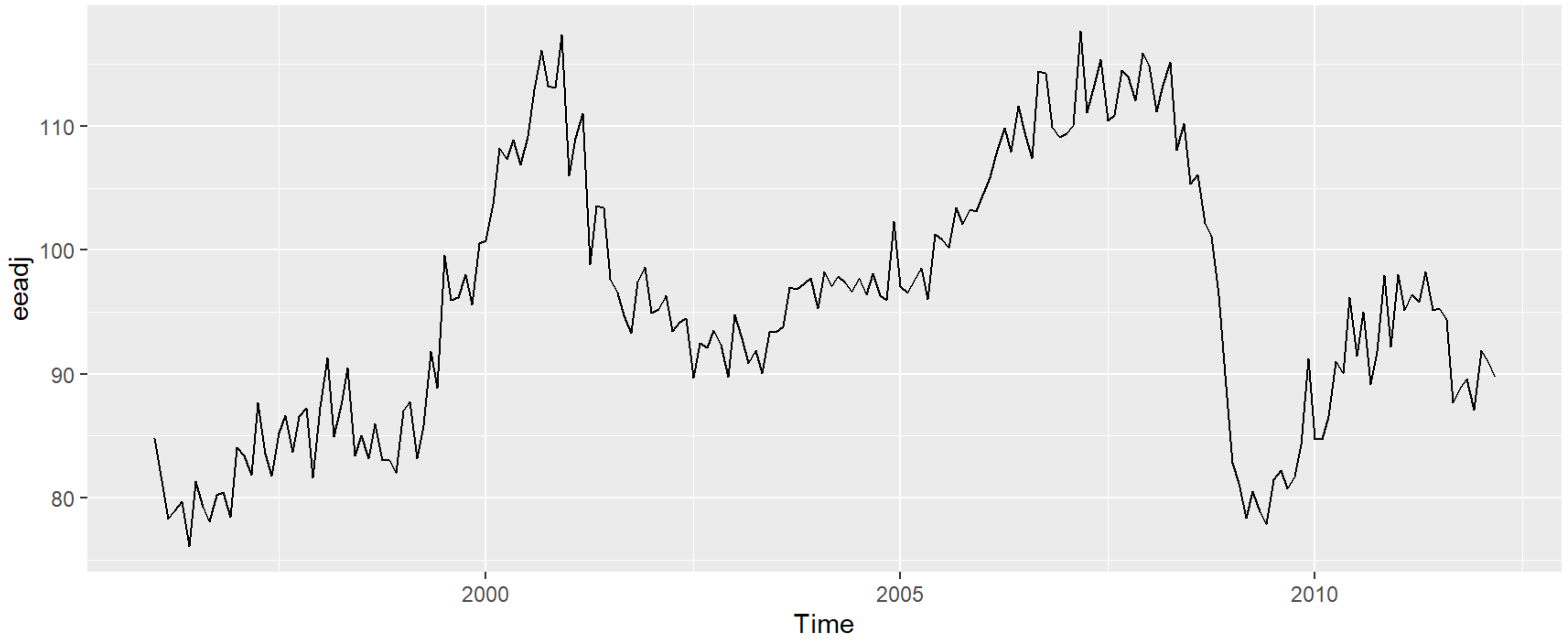
1. Plot the data. Identify any unusual observations.
2. If necessary, transform the data (using a Box-Cox transformation) to stabilize the variance.
3. If the data are non-stationary: take first differences of the data until the data are stationary.
4. Examine the ACF/PACF: Is an AR($p$) or MA($q$) model appropriate?
5. Try your chosen model(s), and use the AICc to search for a better model.
6. Check the residuals from your chosen model by plotting the ACF of the residuals, and doing a portmanteau test of the residuals. If they do not look like white noise, try a modified model.
7. Once the residuals look like white noise, calculate forecasts.

auto.arima() could do steps 3 to 5 for us

## Example: Seasonally adjusted electrical equipment orders

We will apply this procedure to the seasonally adjusted electrical equipment orders data *elecequip*

```
elecequip %>% stl(s.window='periodic') %>% seasadj() -> eeadj
autoplot(eeadj)
```



Monthly seasonally adjusted electrical equipment orders index in the Euro area (1996-2011)

1. The time plot shows some sudden changes, particularly the big drop in 2008/2009. These changes are due to the global economic environment. Otherwise there is nothing unusual about the time plot and there appears to be no need to do any data adjustments.
2. There is no evidence of changing variance, so we will not do a Box-Cox transformation.
3. The data are clearly non-stationary, as the series wanders up and down for long periods.

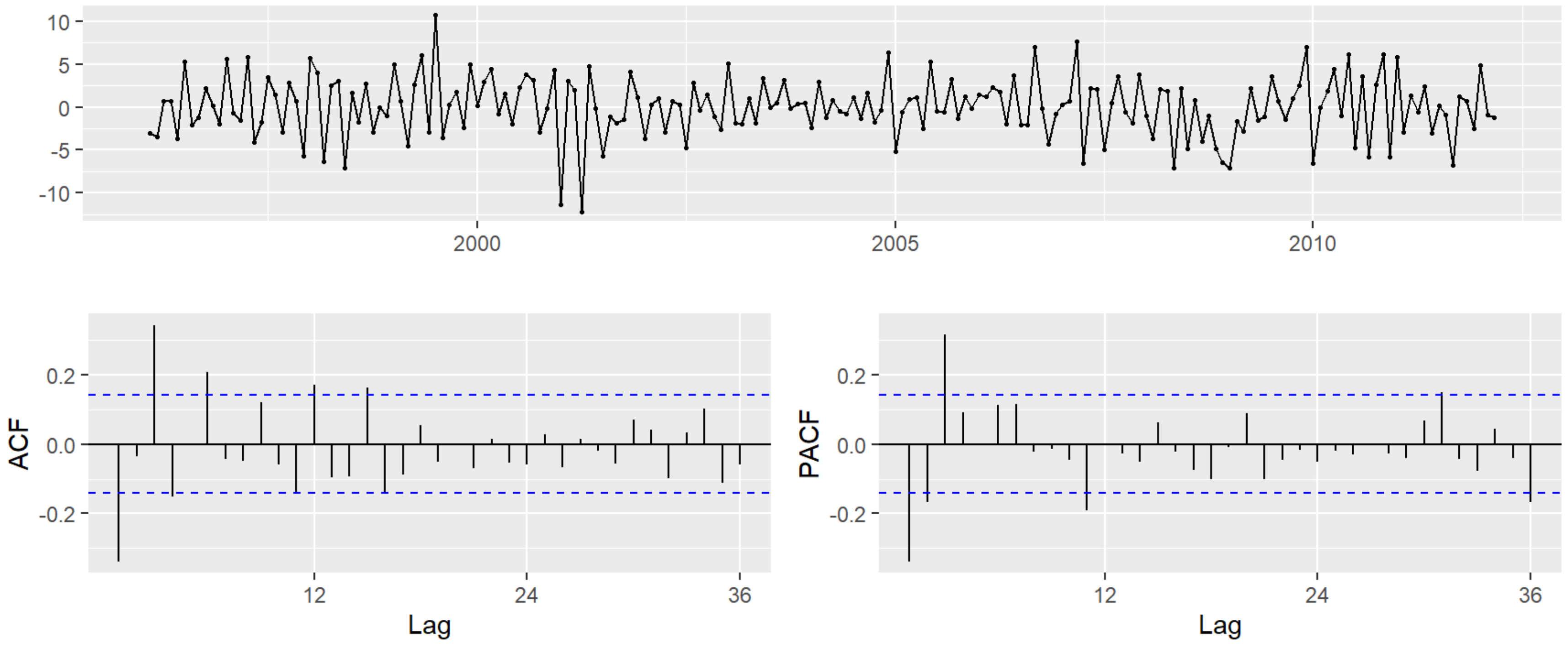
```
adf.test(eeadj)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  eeadj
## Dickey-Fuller = -2.4033, Lag order = 5, p-value = 0.4073
## alternative hypothesis: stationary
```

```
kpss.test(eeadj)
```

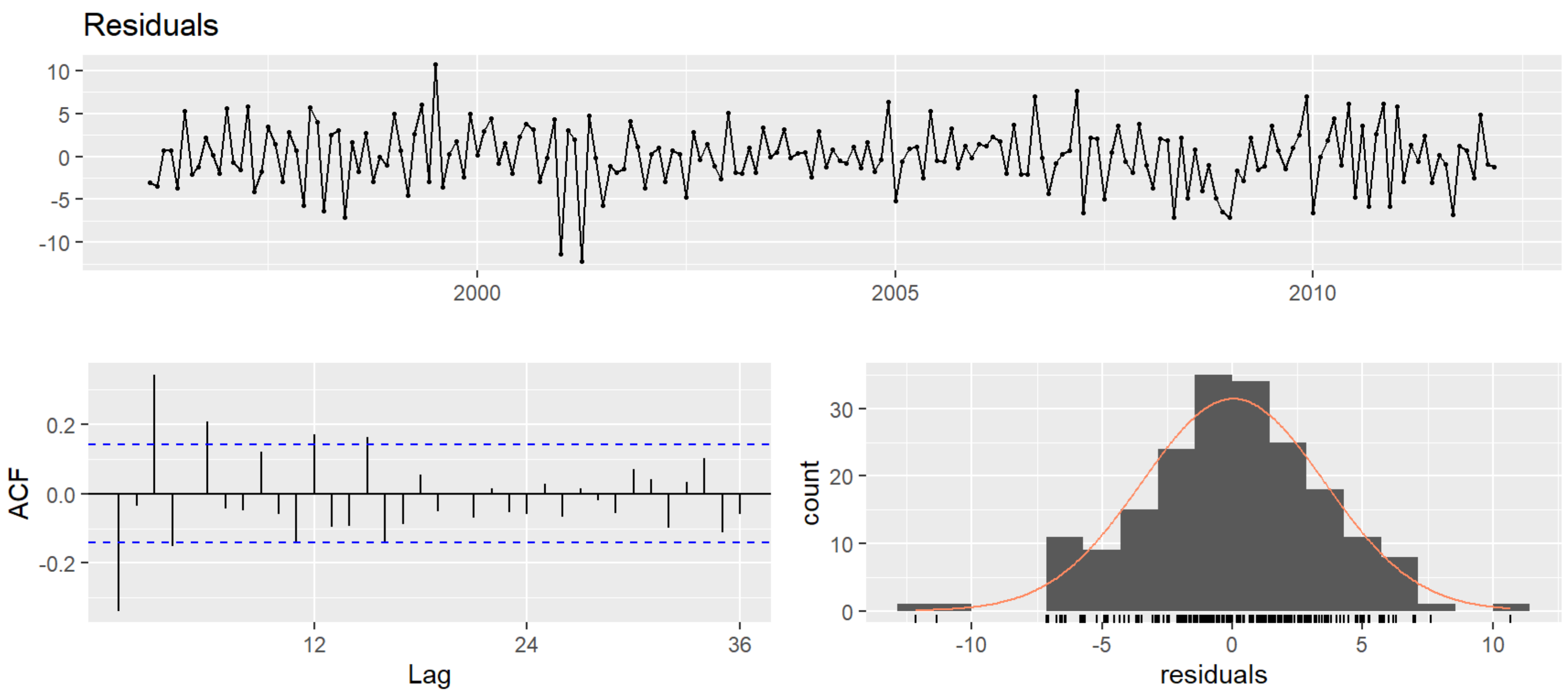
```
##
##  KPSS Test for Level Stationarity
##
## data:  eeadj
## KPSS Level = 0.70165, Truncation lag parameter = 4, p-value =
## 0.0134
```

```r
eeadj %>% diff() %>% ggtsdisplay(main="")
```



Time plot and ACF and PACF plots for the differenced seasonally adjusted electrical equipment data.

```
checkresiduals(diff(eeadj))
```



From the figures, the differenced seasonally adjusted data look stationary.

From the ACF and PACF shown, an AR(3) model with an MA$(1)$ or ARIMA$(3,1,1)$ component seem to be a likely candidate model. We also estimate ARIMA$(3,1,0)$ and ARIMA$(4,1,0)$ and ARIMA$(2,1,0)$ and based on the next few slides, ARIMA$(3,1,1)$ has the lowest AICc. After confirmation that ARIMA$(3,1,1)$ is the better model based on the AICc, we then run checkresiduals on ARIMA$(3,1,1)$.

```r
fit <- Arima(eeadj, order=c(3,1,1))
summary(fit)
```

```
## Series: eeadj
## ARIMA(3,1,1)
##
## Coefficients:
##          ar1     ar2     ar3      ma1
##       0.0044  0.0916  0.3698  -0.3921
## s.e.  0.2201  0.0984  0.0669   0.2426
##
## sigma^2 estimated as 9.577:  log likelihood=-492.69
## AIC=995.38   AICc=995.7   BIC=1011.72
##
## Training set error measures:
##                      ME     RMSE      MAE         MPE     MAPE    MASE
## Training set 0.03288179 3.054718 2.357169 -0.00647009 2.481603 0.28844
##                     ACF1
## Training set 0.008980578
```

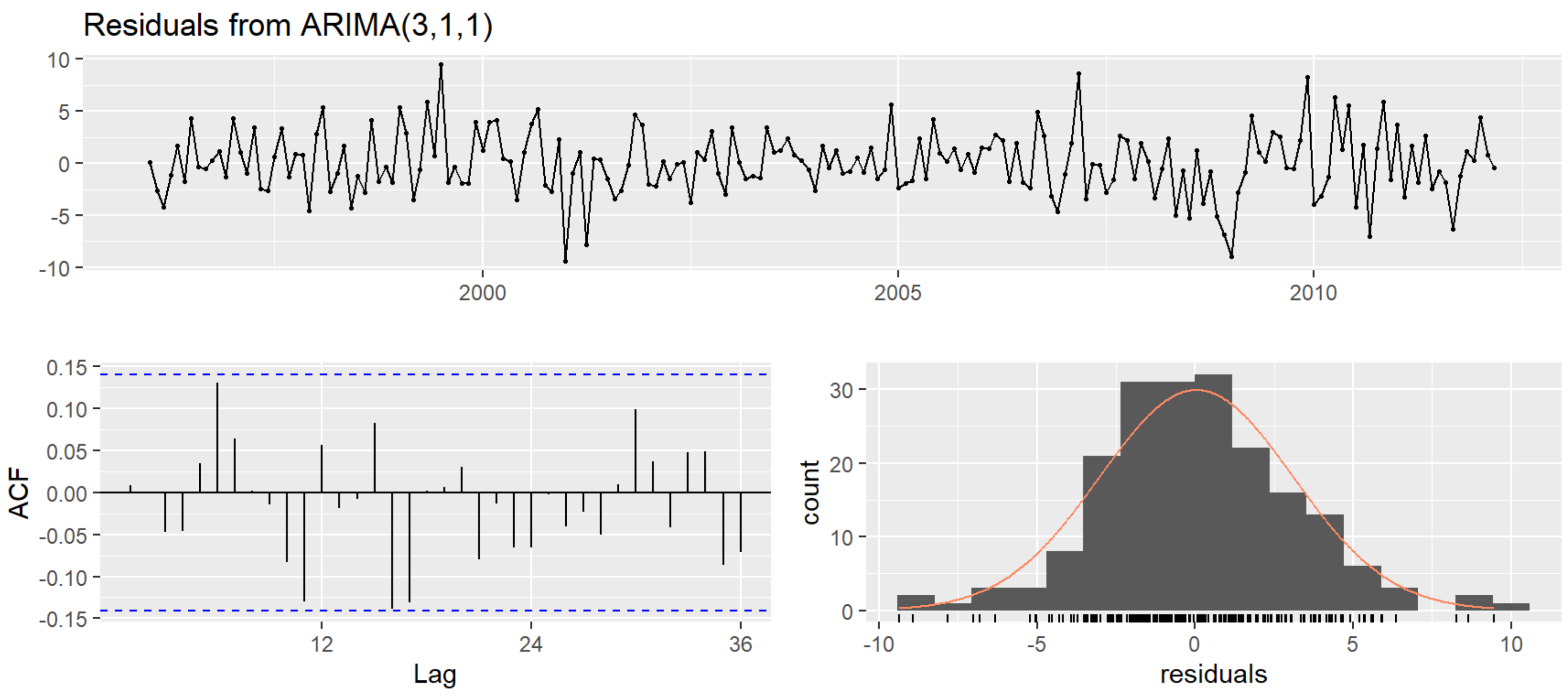
```r
fit1 <- Arima(eeadj, order=c(3,1,0))
summary(fit1)
```

```
## Series: eeadj
## ARIMA(3,1,0)
##
## Coefficients:
##           ar1      ar2     ar3
##       -0.3418  -0.0426  0.3185
## s.e.   0.0681   0.0725  0.0682
##
## sigma^2 estimated as 9.639:  log likelihood=-493.8
## AIC=995.6   AICc=995.81   BIC=1008.67
##
## Training set error measures:
##                      ME     RMSE      MAE         MPE     MAPE     MASE
## Training set 0.03546433 3.072732 2.387717 -0.01504649 2.513219 0.292178
##                     ACF1
## Training set -0.03026898
```

```r
fit2 <- Arima(eeadj, order=c(4,1,0))
summary(fit2)
```

```
## Series: eeadj
## ARIMA(4,1,0)
##
## Coefficients:
##           ar1      ar2     ar3     ar4
##       -0.3731  -0.0388  0.3520  0.0967
## s.e.   0.0717   0.0721  0.0722  0.0716
##
## sigma^2 estimated as 9.598:  log likelihood=-492.89
## AIC=995.79   AICc=996.11   BIC=1012.13
##
## Training set error measures:
##                      ME     RMSE      MAE         MPE    MAPE      MASE
## Training set 0.03397165 3.058099 2.365728 -0.01074768 2.48889 0.2894873
##                      ACF1
## Training set 0.0008203733
```

```r
checkresiduals(fit)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,1,1)
## Q* = 24.034, df = 20, p-value = 0.2409
##
## Model df: 4.   Total lags used: 24
```

The ACF plot of the residuals from the ARIMA$(3, 1, 1)$ model shows that all correlations are within the threshold limits, indicating that the residuals are behaving like white noise. A portmanteau test returns a large p-value, also suggesting that the residuals are white noise.

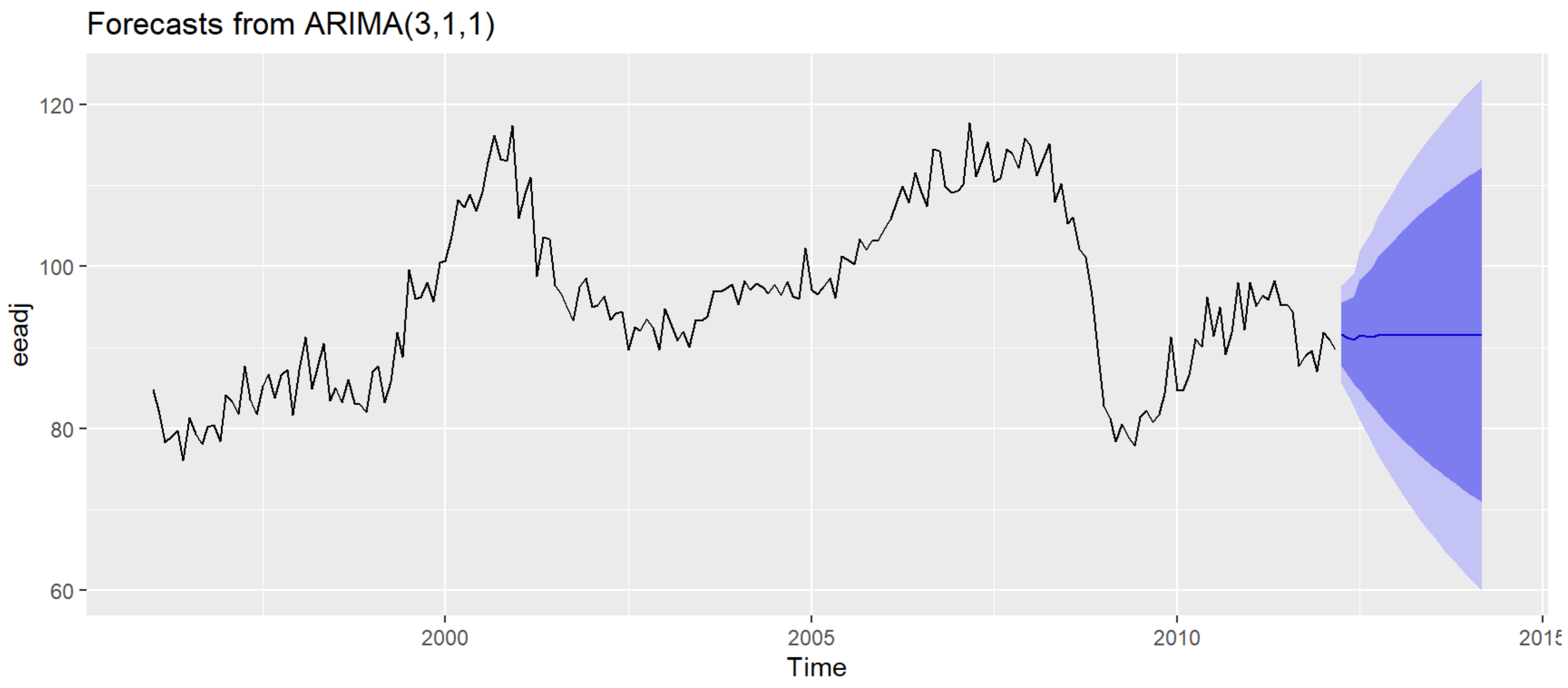
## Using auto.arima() instead

```
fit4 <- auto.arima(eeadj, seasonal=FALSE,
stepwise=FALSE, approximation=FALSE)
summary(fit4)
```

```
## Series: eeadj
## ARIMA(3,1,1)
##
## Coefficients:
##          ar1     ar2     ar3      ma1
##       0.0044  0.0916  0.3698  -0.3921
## s.e.  0.2201  0.0984  0.0669   0.2426
##
## sigma^2 estimated as 9.577:  log likelihood=-492.69
## AIC=995.38   AICc=995.7   BIC=1011.72
##
## Training set error measures:
##                      ME     RMSE      MAE         MPE     MAPE    MASE
## Training set 0.03288179 3.054718 2.357169 -0.00647009 2.481603 0.28844
##                     ACF1
## Training set 0.008980578
```

## Forecast

```
autoplot(forecast(fit))
```



## Point forecasts

Point forecasts can be calculated using the following three steps.

- Expand the ARIMA equation so that $y_t$ is on the left hand side and all other terms are on the right.
- Rewrite the equation by replacing $t$ by $T + h$.
- On the right hand side of the equation, replace future observations by their forecasts, future errors by zero, and past errors by the corresponding residuals.

Beginning with $h = 1$, these steps are then repeated for $h = 2, 3, \ldots$ until all forecasts have been calculated.

## Example

The ARIMA(3,1,1) model fitted in previously can be written as follows

$$(1-\hat{\phi}_1 B-\hat{\phi}_2 B^2-\hat{\phi}_3 B^3)(1-B)y_t=(1+\hat{\theta}_1 B)e_t,$$

where $\hat{\phi}_1=0.0044$, $\hat{\phi}_2=0.0916$, $\hat{\phi}_3=0.3698$ and $\hat{\theta}_1=-0.3921$.

Then we expand the left hand side to obtain

$$\left[1-(1+\hat{\phi}_1)B+(\hat{\phi}_1-\hat{\phi}_2)B^2+(\hat{\phi}_2-\hat{\phi}_3)B^3+\hat{\phi}_3 B^4\right]y_t,$$

and applying the backshift operator gives

$$\begin{aligned} y_t-(1+\hat{\phi}_1)y_{t-1}+(\hat{\phi}_1-\hat{\phi}_2)y_{t-2}+(\hat{\phi}_2-\hat{\phi}_3)y_{t-3}+\hat{\phi}_3 y_{t-4} \\ =e_t+\hat{\theta}_1 e_{t-1}. \end{aligned}$$

Finally, we move all terms other than $y_t$ to the right hand side:

$$\begin{aligned} y_t = (1+\hat{\phi}_1)y_{t-1} - (\hat{\phi}_1 - \hat{\phi}_2)y_{t-2} - (\hat{\phi}_2 - \hat{\phi}_3)y_{t-3} - \hat{\phi}_3 y_{t-4} \\ + e_t + \hat{\theta}_1 e_{t-1}. \end{aligned}$$

This completes the first step. While the equation now looks like an ARIMA(4,0,1), it is still the same ARIMA(3,1,1) model we started with.

For the second step, we replace $t$ by $T+1$:

$$\begin{aligned} \hat{y}_{T+1} = (1+\hat{\phi}_1)y_T - (\hat{\phi}_1 - \hat{\phi}_2)y_{T-1} - (\hat{\phi}_2 - \hat{\phi}_3)y_{T-2} - \hat{\phi}_3 y_{T-3} \\ + e_{T+1} + \hat{\theta}_1 \hat{e}_T. \end{aligned}$$

Assuming we have observations up to time $T$, all values on the right hand side are known except for $e_{T+1}$ which we replace by zero and $e_T$ which we replace by the last observed residual $\hat{e}_T$.

$$\hat{y}_{T+1|T} = (1+\hat{\phi}_1)y_T - (\hat{\phi}_1 - \hat{\phi}_2)y_{T-1} - (\hat{\phi}_2 - \hat{\phi}_3)y_{T-2} \\ - \hat{\phi}_3 y_{T-3} + \hat{\theta}_1 \hat{e}_T.$$

A forecast of $y_{T+2}$ is obtained by replacing $t$ by $T+2$. All values on the right hand side will be known at time $T$ except $y_{T+1}$ which we replace by $\hat{y}_{T+1|T}$, and $e_{T+2}$ and $e_{T+1}$, both of which we replace by zero:

$$\hat{y}_{T+2|T} = (1+\hat{\phi}_1)\hat{y}_{T+1|T} - (\hat{\phi}_1 - \hat{\phi}_2)y_T - (\hat{\phi}_2 - \hat{\phi}_3)y_{T-1} \\ - \hat{\phi}_3 y_{T-2}.$$

The process continues in this manner for all future time periods. In this way, any number of point forecasts can be obtained.

## Prediction intervals

These are generated in function forecast()

The actual calculations of prediction intervals are more difficult.

Examples from simplier models

The first forecast interval is easily calculated. If $\hat{\sigma}$ is the standard deviation of the residuals, then a 95% forecast interval is given by $\hat{y}_{T+1|T} \pm 1.96\hat{\sigma}$.

## Multi-step forecast intervals for ARIMA$(0, 0, q)$ models

$$y_t = e_t + \sum_{i=1}^{q} \theta_i e_{t-i}.$$

are based on the estimated forecast variance

$$\sigma_h = \hat{\sigma}^2 \left[1 + \sum_{i=1}^{h-1} \theta_i^2\right], \qquad \text{for } h = 2, 3, \ldots.$$

A 95% forecast interval is given by $\hat{y}_{T+h|T} \pm 1.96\sqrt{\sigma_h}$.

An AR(1) model can be written as an MA($\infty$) model. Using this equivalence, the above result for MA($q$) models can also be used to obtain prediction intervals for AR(1) models.

More general results are discussed in advanced textbooks (Brockwell and Davis(2016)).

## More importantly for us …

The prediction intervals for ARIMA models are based on assumptions that the residuals are uncorrelated and normally distributed. If either of these are assumptions do not hold, then the forecast intervals may be incorrect.

For this reason, always plot the ACF and histogram of the residuals to check the assumptions before producing forecast intervals.

In general, prediction intervals from ARIMA models will increase as the forecast horizon increases.

For stationary models (i.e., with $d = 0$), they will converge so forecast intervals for long horizons are all essentially the same.

For $d > 1$, the forecast intervals will continue to grow into the future.

ARIMA-based intervals tend to be too narrow as only the variation in the errors has been accounted for.

There may also be variations in the parameter estimates, and in the model order, that have not been included in the calculation.

The calculation assumes that the historical patterns that have been modelled will continue into the forecast period.

## Thank you for your attention