# Fitting and Selecting ARIMA models

FISH 550 – Applied Time Series Analysis  Download Rmd pdf

Eli Holmes

10 Apr 2025

# ARIMA

- I = Integrated: Refers to differencing the time series to make $z_t$ which is stationary (i.e., removing trends).
- A = AutoRegressive (AR): An additive weighted sum of past values to model the current value.
- MA = Moving Average: Uses an additive weighted sum past forecast errors to model the current value.

$$z_t = \phi_1 z_{t-1} + \cdots + \phi_p z_{t-p} + \epsilon_t$$

$$\epsilon_t = \theta e_t + \cdots + \theta_q e_{t-q}$$

# Box-Jenkins method

A. ARIMA(p,d,q) Model selection

1. Evaluate stationarity
2. Fix stationarity problems - Select the differencing level (d)
3. Selection of the AR level (p)
4. Selection of the MA level (q)

B. Parameter estimation

C. Model checking

1. Test model residuals for distribution assumptions (e.g. Normality)
2. Test model residuals for temporal correlation

# Notes

- For ARIMA models, much of the Box-Jenkins method will be automated with the **forecast** package functions, which you will use in the lab.
- For this lecture, I have removed much of the mathematical background. See ATSA 2023 lectures for more of that background.

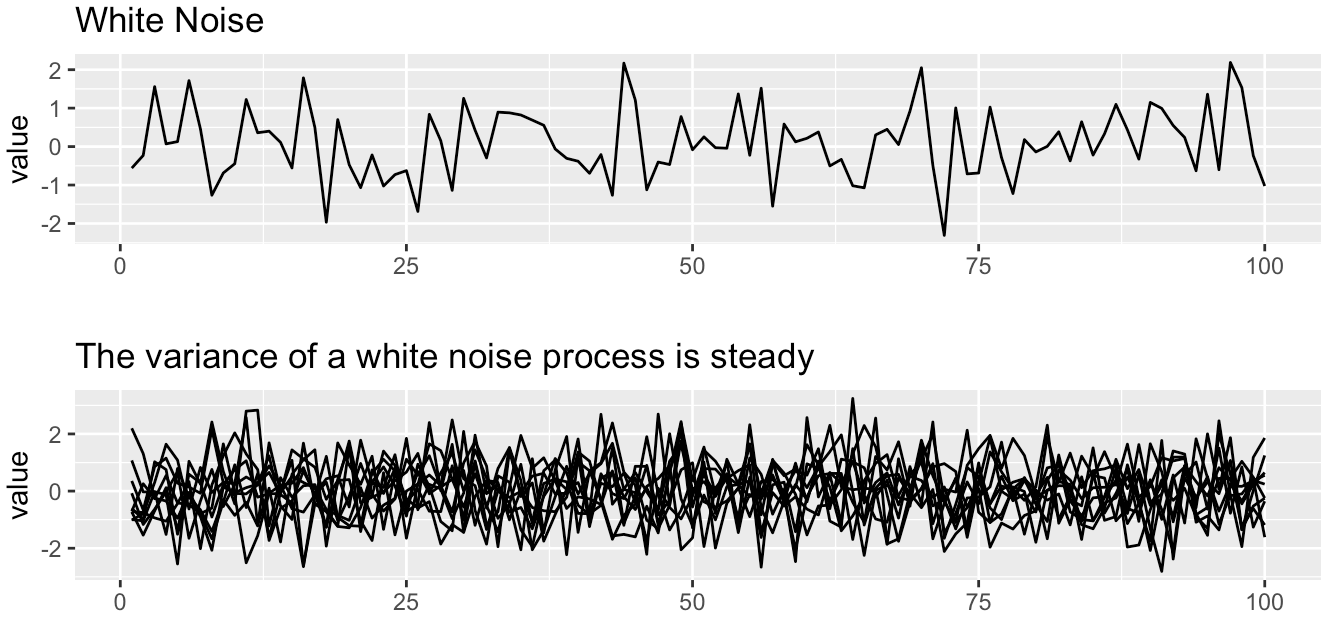
# Stationarity

Stationarity means ‘not changing in time’ in the context of time-series models. Typically we test the trend and variance, however more generally all statistical properties of a time-series is time-constant if the time series is ‘stationary’.

# Example

Many ARMA models exhibit stationarity. White noise is one type:

$$x_t = e_t, e_t \sim N(0, \sigma)$$

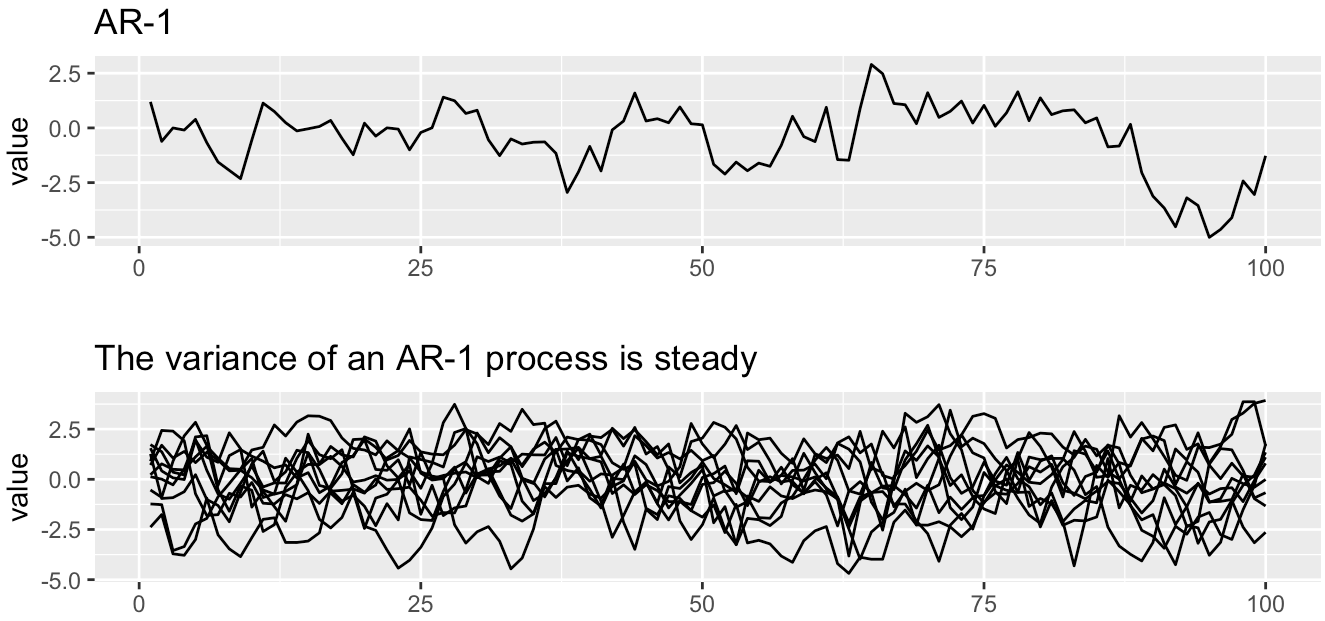


# Example

An AR-1 process with $-1 < \phi < 1$

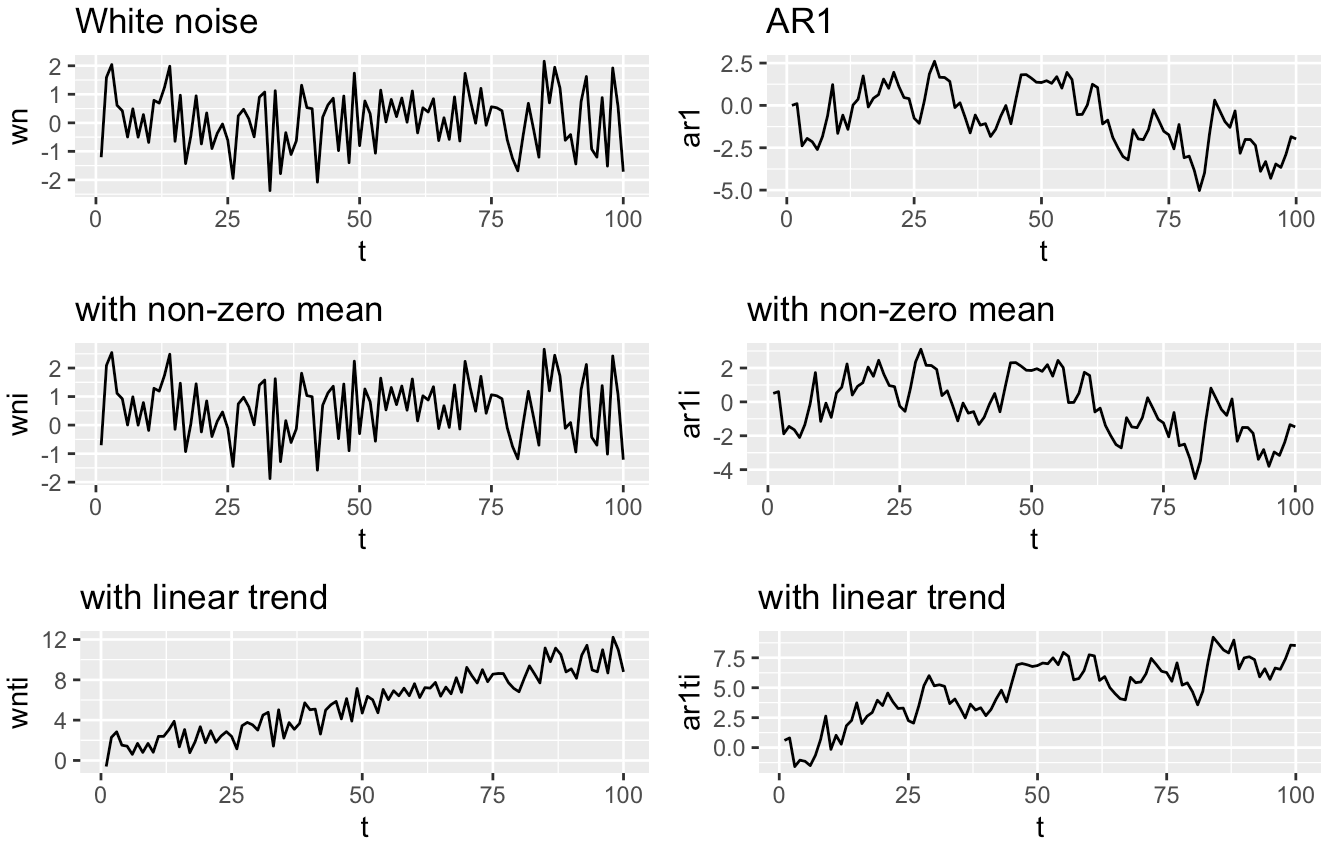
$$x_t = \phi x_{t-1} + e_t$$

is also stationary.



# Stationarity around non-zero mean

We can also have stationarity around a non-zero level or around a linear trend.

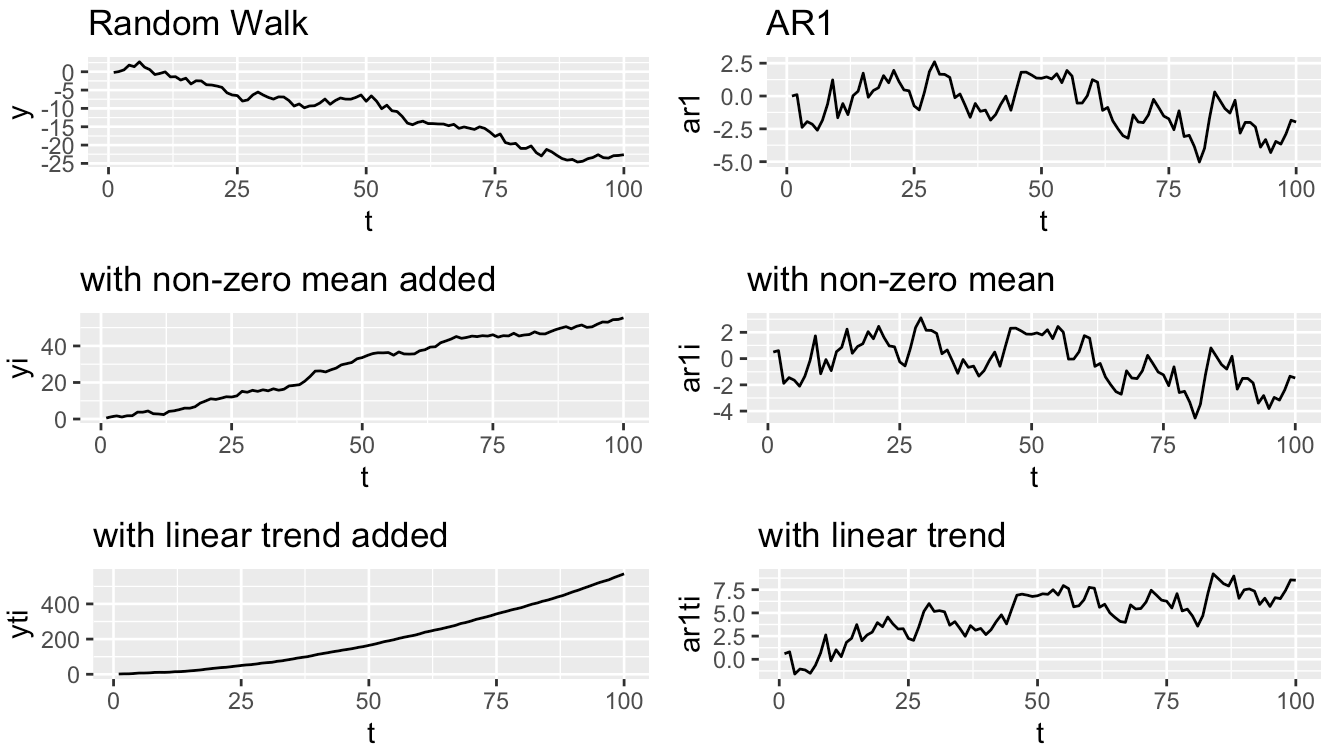


# Non-stationarity

One of the most common forms of non-stationarity that is tested for is that the process is a random walk $x_t = x_{t-1} + e_t$, aka a 'unit root' process. A test for an underlying random walk is called a 'unit root' test.

# Random walk with trends

There are two classic types of trends: a linear trend and an exponential trend.



# Testing for stationarity

Why is evaluating stationarity important?

For this lecture, we are using the Box-Jenkins method for forecasting. Step 1 is to create a transformed stationary time series that we can fit an ARMA model to (Wold Decomposition).

In a real data analysis, there are other reasons:

- Many standard algorithms for fitting ARMA models assume stationarity.
- Many AR models are stationary. If your data are not, you are fitting a model that is fundamentally inconsistent with your data.
- Many processes in environmental science are fundamentally random walks, i.e. non-stationary, e.g. movement, population growth, genetic drift.

# Testing for stationarity

We will discuss three common approaches to evaluating stationarity:

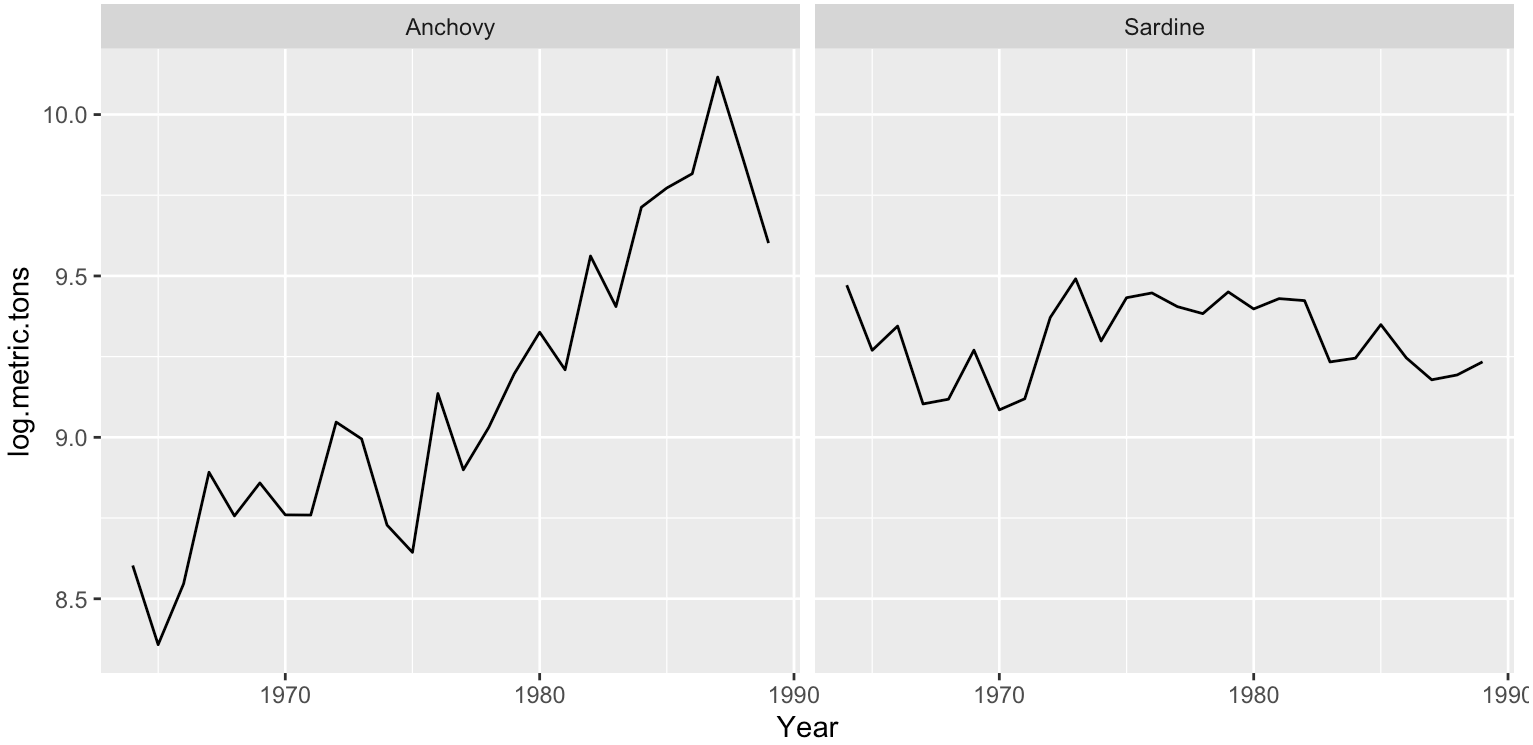
- Visual test
- (Augmented) Dickey-Fuller test
- KPSS test

# Visual test

The visual test is simply looking at a plot of the data versus time. Look for

- Change in the level over time. Is the time series increasing or decreasing? Does it appear to cycle?
- Change in the variance over time. Do deviations away from the mean change over time, increase or decrease?

# Anchovy and sardine catch in Greek waters



# Classic stationarity tests

## Dickey-Fuller (ADF) vs. KPSS Tests

| Aspect | Dickey-Fuller (ADF) Test | KPSS Test |
|---|---|---|
| **Purpose** | Tests for the presence of a unit root (non-stationarity). | Tests for stationarity (null hypothesis is stationary). |
| **Null Hypothesis ($H_0$)** | Series has a unit root (non-stationary). | Series is stationary. |
| **Alternative Hypothesis ($H_1$)** | Series is stationary. | Series is not stationary (has a unit root). |
| **Test Result Interpretation** | Reject $H_0$ → Stationary (no unit root). | Reject $H_0$ → Non-stationary (has a unit root). |
| **Type of Test** | Uses a regression model to check for unit root. | Compares the series' behavior against a stationary process. |

# Testing with the forecast package

`ndiffs()` uses a unit root test to determine the number of differences required for time series x to be made stationary. If test="kpss", the KPSS test is used with the null hypothesis that x has a stationary root against a unit-root alternative. Then the test returns the least number of differences required to pass the test at the level alpha.

```
library(forecast)
ndiffs(WWWusage)
ndiffs(diff(log(AirPassengers), 12))
```

# Fix stationarity problems

In this lecture we will use differencing, the I in ARIMA model refers to differencing.

# Differencing the data to make the mean stationary

Differencing means to create a new time series $z_t = x_t - x_{t-1}$. First order differencing means you do this once (so $z_t$) and second order differencing means you do this twice (so $z_t - z_{t-1}$).

The `diff()` function takes the first difference:

```
x <- diff(c(1,2,4,7,11))
x
```

```
## [1] 1 2 3 4
```

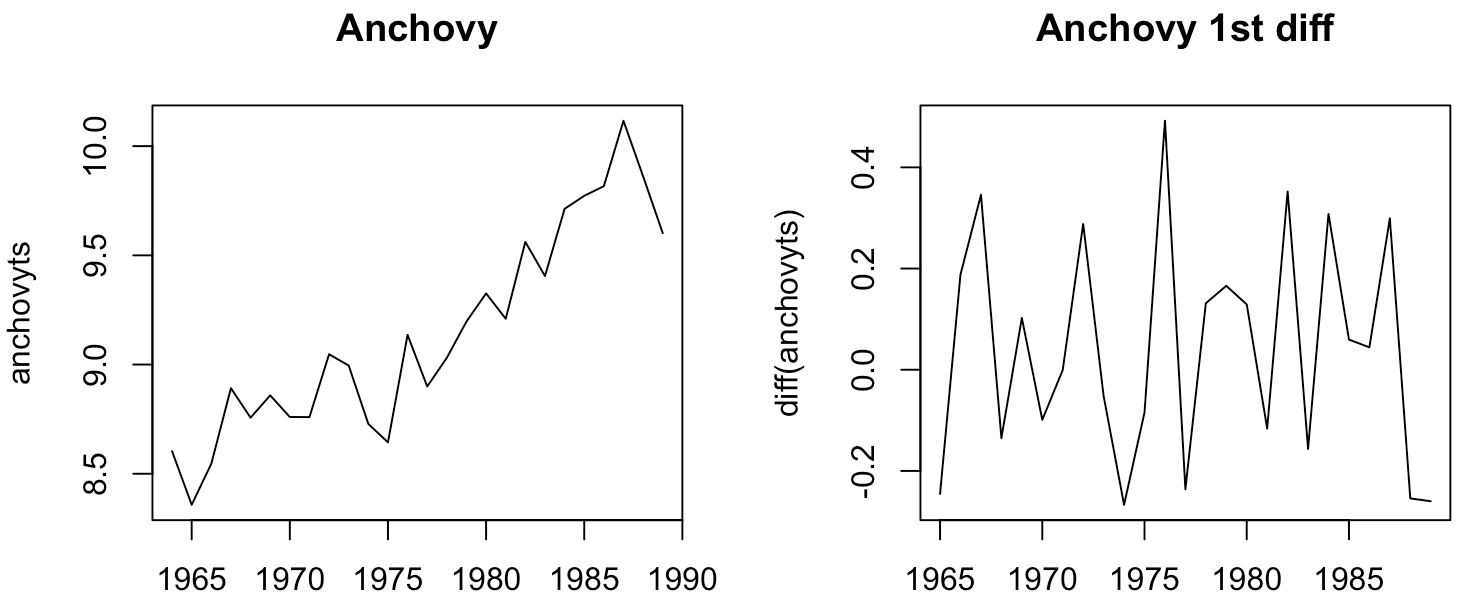
The second difference is the first difference of the first difference.

```
diff(x)
```

```
## [1] 1 1 1
```

# Anchovy catch first differenced

Here is a plot of the anchovy data and its first difference.



# `forecast::ndiffs()` function

You can use the `ndiffs()` function in the forecast package to find the number of differences needed.

```
forecast::ndiffs(anchovyts, test="kpss")
```

```
## [1] 1
```

```
forecast::ndiffs(anchovyts, test="adf")
```

```
## [1] 1
```

The test indicates that one difference ($x_t - x_{t-1}$) will lead to stationarity.

# Summary

Test stationarity before you fit a ARMA model.

**Visual test**: Do the data fluctuate around a level or do they have a trend or look like a random walk?

*Yes or maybe?* -> Apply a “unit root” test. ADF or KPSS

*No or fails the unit root test?* -> Apply differencing and re-test.

*Still not passing?* -> Try a second difference or you may need to transform the data (if say it has an exponential trend).

*Still not passing?* -> ARMA model might not be the best choice. Or you may need to use an adhoc detrend.

**These steps are automated by the forecast package**

# Box-Jenkins method

A. Model form selection

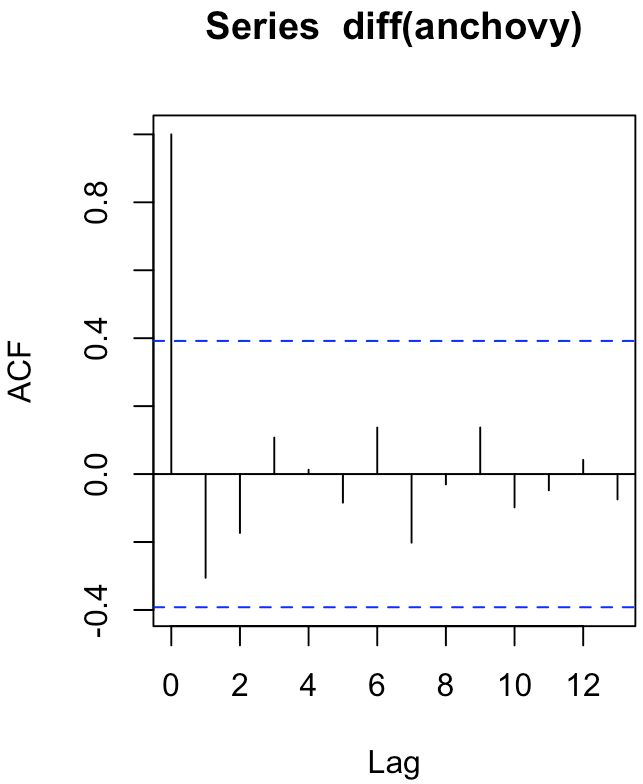
1. Evaluate stationarity
2. Selection of the differencing level (d) – to fix stationarity problems
3. **Selection of the AR level (p)**
4. **Selection of the MA level (q)**

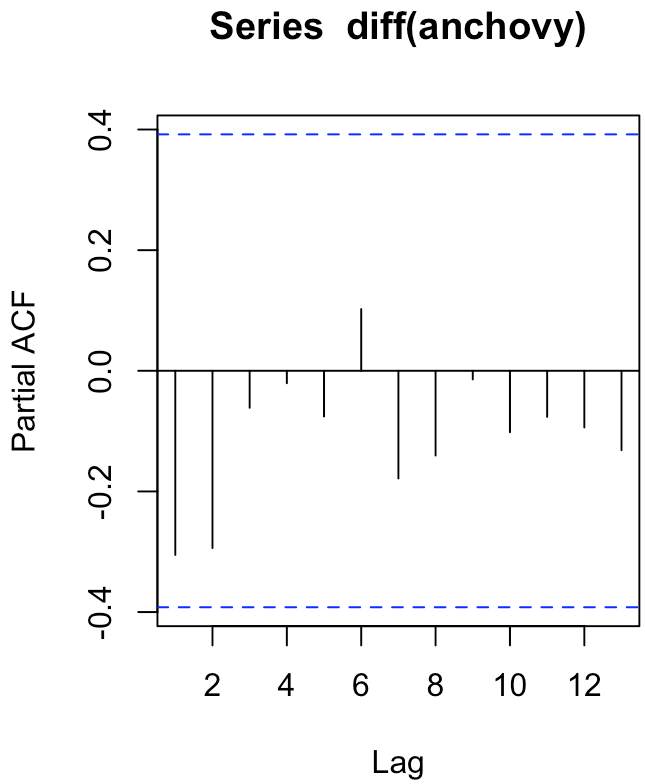
**B. Parameter estimation**

C. Model checking

# ACF and PACF

On Tuesday, you learned how to use ACF and PACF to visually infer the AR and MA lags for a ARMA model. Here is the ACF and PACF of the differenced anchovy time series.



# Formal model selection

This weighs how well the model fits against how many parameters your model has. Basic idea is to fit (many) models and use AIC, AICc or BIC to select. Smallest AIC or BIC is best model.

The `auto.arima()` function in the forecast package in R allows you to easily do this and will also select the level of differencing (via ADF or KPSS tests).

```
forecast::auto.arima(anchovy)
```

Type `?forecast::auto.arima` to see a full description of the function.

# Model selection with `auto.arima()`

```
forecast::auto.arima(anchovy)
```

```
## Series: anchovy
## ARIMA(0,1,1) with drift
##
## Coefficients:
##           ma1   drift
##       -0.6685  0.0542
## s.e.   0.1977  0.0142
##
## sigma^2 = 0.04037:  log likelihood = 5.39
## AIC=-4.79   AICc=-3.65   BIC=-1.13
```

The output indicates that the 'best' model is a MA(1) with first difference. "with drift" means that the mean of the anchovy first differences (the data for the model) is not zero.

# Trace = TRUE

By default, step-wise selection is used for the model search. You can see what models that `auto.arima()` tried using `trace=TRUE`. The models are selected on AICc by default.

```
forecast::auto.arima(anchovy, trace=TRUE)
```

```
##
##  ARIMA(2,1,2) with drift         : 4.765453
##  ARIMA(0,1,0) with drift         : 0.01359746
##  ARIMA(1,1,0) with drift         : -0.1662165
##  ARIMA(0,1,1) with drift         : -3.647076
##  ARIMA(0,1,0)                    : -1.554413
##  ARIMA(1,1,1) with drift         : Inf
##  ARIMA(0,1,2) with drift         : Inf
##  ARIMA(1,1,2) with drift         : 1.653078
##  ARIMA(0,1,1)                    : -1.372929
##
##  Best model: ARIMA(0,1,1) with drift
```

```
## Series: anchovy
## ARIMA(0,1,1) with drift
##
```

# Selected model

First difference of the data is MA(1) with drift

$$x_t - x_{t-1} = \mu + w_t + \theta_1 w_{t-1}$$

aka

$$x_t = x_{t-1} + \mu + w_t + \theta_1 w_{t-1}$$

"Today's value is yesterday's value plus a constant, and influenced by yesterday's shocks/errors"

$w_t$ is white noise.

# Fit to simulated AR(2) data

```
set.seed(100)
a1 = arima.sim(n=100, model=list(ar=c(.8,.1)))
forecast::auto.arima(a1, seasonal=FALSE, max.d=0)
```

```
## Series: a1
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1     mean
##       0.6928  -0.5343
## s.e.  0.0732   0.2774
##
## sigma^2 = 0.7703:  log likelihood = -128.16
## AIC=262.33   AICc=262.58   BIC=270.14
```

The 'best-fit' model is AR(1) not AR(2).

# How often is the 'true' model is chosen

Let's run 100 simulations of a AR(2) process and record the best fits.

```
save.fits = rep(NA,100)
for(i in 1:100){
  a1 = arima.sim(n=100, model=list(ar=c(.8,.1)))
  fit = forecast::auto.arima(a1, seasonal=FALSE, max.d=0, max.q=0)
  save.fits[i] = paste0(fit$arma[1], "-", fit$arma[2])
}
```

Overwhelmingly the correct type of model (AR) is selected, but usually a simpler model of AR(1) is chosen over AR(2).

Table heading is AR order - MA order.

```
table(save.fits)
```

```
## save.fits
## 1-0 2-0 3-0 4-0
##  74  20   5   1
```

# stepwise=FALSE

By default, step-wise selection is used and an approximation is used for the models tried in the model selection step. For a final model selection, you should turn these off to fit a large set of models.

```
forecast::auto.arima(anchovy, stepwise=FALSE,
                     approximation=FALSE)
```

```
## Series: anchovy
## ARIMA(0,1,1) with drift
##
## Coefficients:
##           ma1   drift
##       -0.6685  0.0542
## s.e.   0.1977  0.0142
##
## sigma^2 = 0.04037:  log likelihood = 5.39
## AIC=-4.79   AICc=-3.65   BIC=-1.13
```

# Summary: model selection and fitting

- Once you have dealt with stationarity, you need to determine the order of the model: the AR part and the MA part.
- Also consider if there are reasons to assume a particular structure.
  - Are you using an established model form, from say another paper?
  - Are you fitting to a process that is fundamentally AR only or AR + MA?

# Box-Jenkins method

A. Model form selection

1. Evaluate stationarity
2. Selection of the differencing level (d) – to fix stationarity problems
3. Selection of the AR level (p)
4. Selection of the MA level (q)

B. Parameter estimation

**C. Model checking**

# Check the residuals

Residuals = difference between the expected (fitted) value of $x_t$ and the data

**There is no observation error in an ARMA model.** The expected value is the $x_t$ expected from data up to $t-1$.

For example, the residual for an AR(2) model is $y_t - \hat{x_t}$.

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + w_t$$

$$\hat{x_t} = \phi_1 x_{t-1} + \phi_2 x_{t-2}$$

# `residuals()` function in R

The `residuals()` function will return the residuals for fitted models.

```
fit <- forecast::auto.arima(anchovy)
residuals(fit)
```

```
## Time Series:
## Start = 1
## End = 26
## Frequency = 1
##  [1]  0.008549039 -0.249032308 -0.004098059  0.281393071
##  [5] -0.006015194  0.043859685 -0.123711732 -0.137125900
##  [9]  0.142098844 -0.011246624 -0.328608840 -0.358310373
## [13]  0.198311913 -0.157824727 -0.028321380  0.092732171
## [17]  0.136826748 -0.078995675  0.245238274 -0.046755189
## [21]  0.222279848  0.153983301  0.093036353  0.307250228
## [25] -0.103051063 -0.383026466
```

# `fitted()` function in R

The `fitted()` function will return the expected values. Remember that for a ARMA model, these are the expected values conditioned on the data up to time $t-1$.

```
fitted(fit)
```

```
## Time Series:
## Start = 1
## End = 26
## Frequency = 1
##  [1] 8.594675 8.606878 8.550151 8.610325 8.762619 8.814978
##  [7] 8.883569 8.896418 8.905111 9.006436 9.056857 9.002066
## [13] 8.937456 9.057378 9.059236 9.104026 9.188947 9.288486
## [19] 9.316477 9.451955 9.490634 9.618501 9.723727 9.808749
## [25] 9.964784 9.984801
```

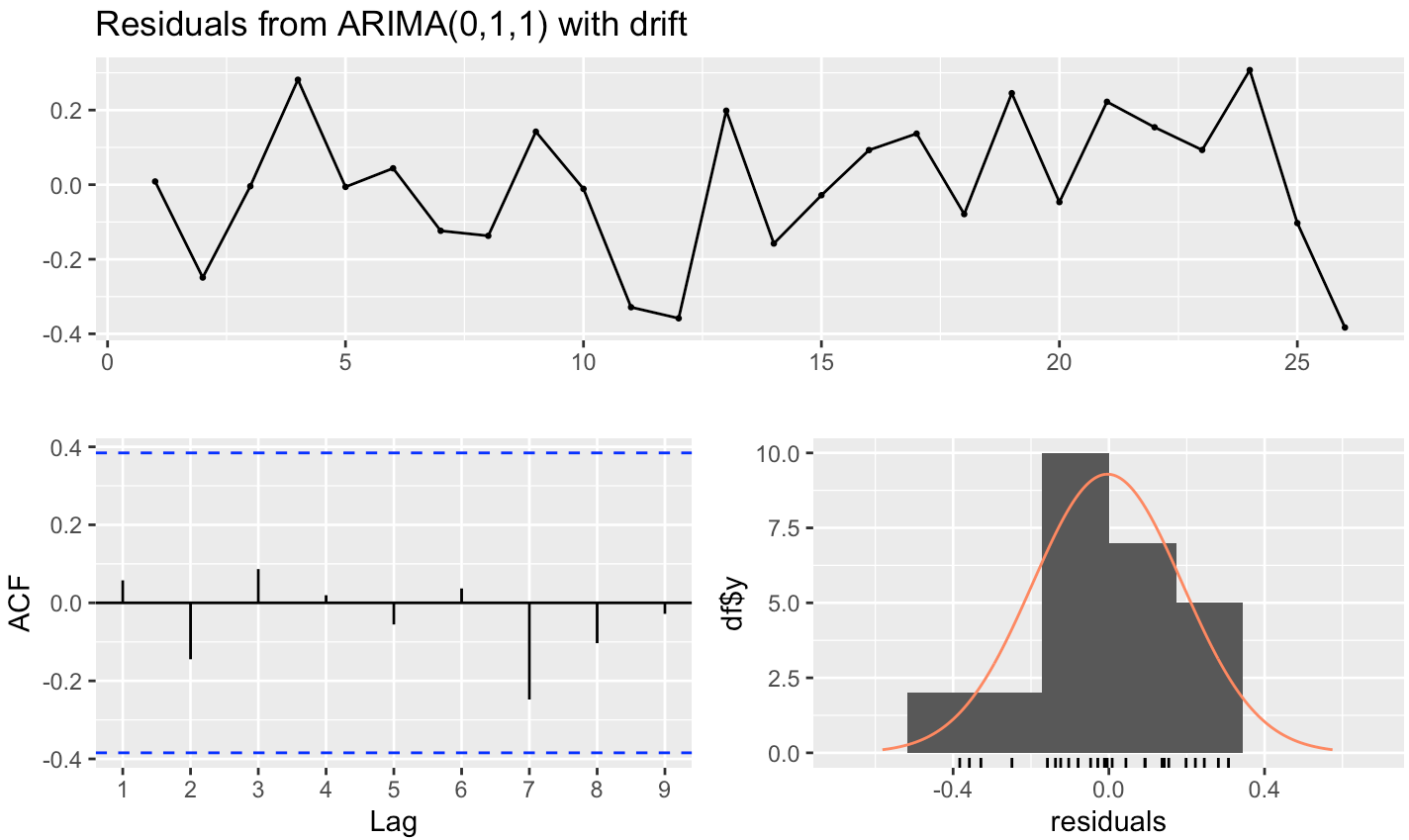
The residuals are `data` minus `fitted`.

# Standard residuals tests

- Plot the residuals. They should look roughly like white noise.
- Look at the ACF of the residuals. They should be uncorrelated.
- Look at the histogram. They should be normally distributed (if that is your error assumption).

# Residuals check with forecast package

```
forecast::checkresiduals(fit)
```



# Test for autocorrelation

The standard test for autocorrelated time-series residuals is the Ljung-Box test. The null hypothesis for this test is **no autocorrelation**. We do not want to reject the null.

```
forecast::checkresiduals(fit, plot=FALSE)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1) with drift
## Q* = 1.0902, df = 4, p-value = 0.8958
##
## Model df: 1.   Total lags used: 5
```

$p > 0.05$ would be interpreted as not enough statistical evidence to reject the null hypothesis.

# That concludes Fitting and Selecting