Module 5: Moving Average and ARIMA models

Andrew Parnell, School of Mathematics and Statistics,
University College Dublin

Learning outcomes

- Recognise and understand the basic theory behind MA(1) and MA(q) models
- Recognise the basic ARMA(p,q) formulation
- Understand what an ARIMA(p,d,q) model is
- How to fit all of the above in JAGS

The files `jags_moving_average.R` and `jags_ARIMA.R` are relevant to this module

Reminder: AR models

- An Autoregressive (AR) model works by making the current data point dependent on the previous value, dampened by a parameter
- The usual likelihood used is:

$$y_t \sim N(\alpha + \phi y_{t-1}, \sigma^2)$$

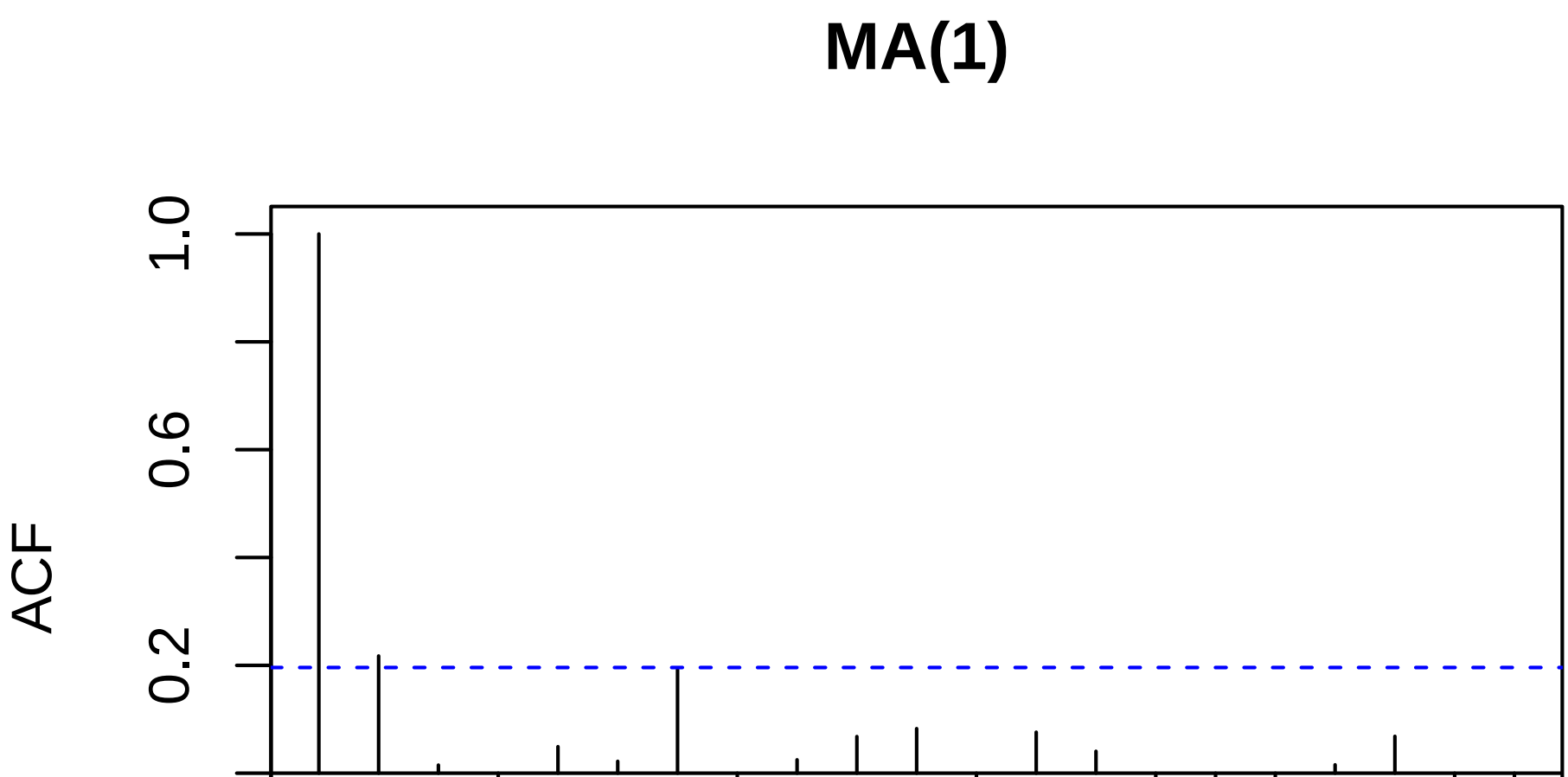
- ϕ is usually constrained (via the prior distribution) to lie between -1 and 1. Outside that range the process blows up
- The sample PACF is often a good way of diagnosing if an AR model might be appropriate

Intro to Moving Average Models

- Moving Average (MA) models are similar to AR models but they depend on the previous residual of the series rather than the value itself
- If the previous residual was large then we want to make a big change to the next prediction
- If the previous residual was small then we might not want to make much of a change
- An MA model is very similar to the exponential smoothing heuristic models which are sometimes used for forecasting. They are also sometimes known as ‘error correction models’

Moving average models and the ACF/PACF

- Recall that the sample partial autocorrelation function (PACF) can be used to diagnose whether an AR model is appropriate (and also suggest the order p)
- For the MA model, it is the sample autocorrelation function (ACF) helps determine the order of the model



Example 1: MA(1)

- The MA(1) model is defined as:

$$y_t = \alpha + \theta\epsilon_{t-1} + \epsilon_t$$

  where $\epsilon_t \sim N(0, \sigma^2)$ as usual
- Parameter α represents the overall mean, whilst θ controls the amount of weight placed on previous residuals
- Unlike the AR model there is no restriction on the value of θ, though negative values can sometimes be physically unrealistic
- The likelihood version of the model is:

$$y_t \sim N(\alpha + \theta\epsilon_{t-1}, \sigma^2)$$

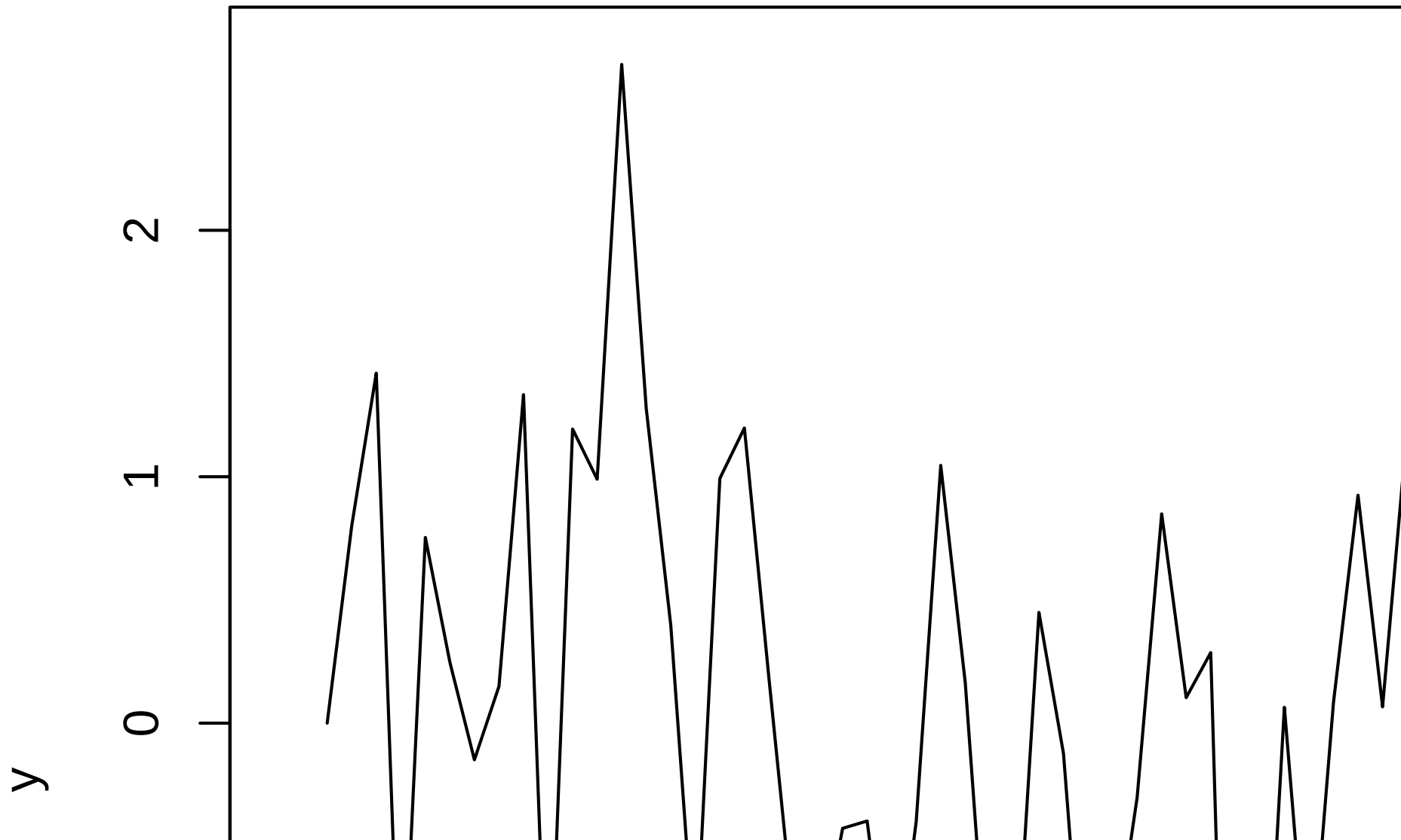
Simulating from the MA(1) process

Below is some simple code to simulate from an MA(1) process. Note that the first values of `y` and `eps` need to be initialised

```
T = 100 # Number of observations
sigma = 1 # Residual sd
alpha = 0 # Mean
theta = runif(1) # Choose a positive value
y = eps = rep(NA,T)
y[1] = alpha
eps[1] = 0
for(t in 2:T) {
  y[t] = rnorm(1, mean = alpha + theta * eps[t-1], sd = sig
  eps[t] = y[t] - alpha - theta * eps[t-1]
}
```

Resulting plot

```
plot(1:T,y,type='l')
```



JAGS code for the MA(1) model with vague priors

```
model_code = '
model
{
  # Set up residuals
  eps[1] <- y[1] - alpha
  # Likelihood
  for (t in 2:T) {
    y[t] ~ dnorm(alpha + theta * eps[t-1], tau)
    eps[t] <- y[t] - alpha - theta * eps[t-1]
  }
  # Priors
  alpha ~ dnorm(0, 0.01)
  theta ~ dnorm(0, 0.01)
  tau <- 1/pow(sigma, 2)
  sigma ~ dunif(0, 10)
}
'
```

Extending to MA(q)

- It's reasonably straightforward to extend this model to have the current value of y depending on more than one previous residual
- The model becomes an MA(q) model with:

$$y_t \sim N(\alpha + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q}, \sigma^2)$$

- The parameters are as before, except there are now q values of θ.
- Usually when estimated they will decrease with q, the older residuals matter less
- In JAGS it's usually help to re-write the long sum in the mean as: `inprod(theta, eps[(t-q):(t-1)])` just like we did with the AR model

JAGS code for MA(q) (with vague priors)

```
model_code = '
model
{
  # Set up residuals
  for(t in 1:q) {
    eps[t] <- y[t] - alpha
  }
  # Likelihood
  for (t in (q+1):T) {
    y[t] ~ dnorm(mean[t], tau)
    mean[t] <- alpha + inprod(theta, eps[(t-q):(t-1)])
    eps[t] <- y[t] - alpha - inprod(theta, eps[(t-q):(t-1)]
  }
  # Priors
  alpha ~ dnorm(0.0,0.01)
  for (i in 1:q) {
    theta[i] ~ dnorm(0.0,0.01)
  }
```

Combining AR and MA into ARMA

- There is no reason why we have to use just AR or MA on their own
- It's possible to combine them together, for example:

$$y_t = \alpha + \phi y_{t-1} + \theta \epsilon_{t-1} + \epsilon_t$$

  This is an Autoregressive Moving Average (ARMA) model
- It's often written as ARMA(p,q) where p is the number of AR terms (here 1) and q the number of MA terms (here also 1)
- ARMA models can deal with a very wide variety of flexible time series behaviour, though they remain stationary
- The likelihood format is:

$$y_t \sim N(\alpha + \phi y_{t-1} + \theta \epsilon_{t-1}, \sigma^2)$$

Combining ARMA with the random walk to produce ARIMA

- There is one other time series model we have already met, that of the random walk:

$$y_t - y_{t-1} \sim N(0, \sigma^2)$$

  This type of time series model works on the *differences* of the data
- It's possible to difference more than once, e.g. $y_t - 2y_{t-1} + y_{t-2}$ is the second difference of y_t
- Again we can combine these ideas into the ARMA framework to produce an ARIMA model (the I stands for integrated, i.e. differenced)

Example: the ARIMA(1,1,1) model

- The ARIMA model is written as ARIMA(p,d,q) where p and q are as before and d is the number of differences.
- If we want to fit an ARIMA(1,1,1) model we first let $z_t = y_t - y_{t-1}$ then run the model:

$$z_t \sim N(\alpha + \phi z_{t-1} + \theta \epsilon_{t-1}, \sigma^2)$$

- This is an ARMA model on the first differences

General format: the ARIMA(p,d,q) model

- First take the dth difference of the series y_t, and call this z_t. You can do this in R with the `diff` function, e.g. `diff(y, differences = 2)`
- Then fit the model:

$$z_t \sim N(\alpha + \phi_1 z_{t-1} + \ldots + \phi_p z_{t-p} + \theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q}, \sigma^2)$$

- In JAGS we use the `inprod` function to get round all the long sums
- There's no obvious way of choosing the values of p, d, and q. In later modules we look at some model choice options

Priors for ARIMA models

There are four broad classes of parameters in the model:

- α which controls the overall mean, and is pretty well informed by the data. Often people remove this parameter by just mean centering the process y
- ϕ which controls the AR terms. There are complicated constraints on the parameter values to make the series stationary. However, the posterior distribution will usually be fine as it will be informed by the data, so usually just restricting them to (-1,1) is adequate
- θ which controls the MA terms. There are no restrictions on the values of these but sometimes it makes sense if the values decrease with lag.
- σ controls the residual standard deviation (sometimes known as the white noise standard deviation). If we have an idea on the likely range of the data we can usually place at least a good upper limit on the value of σ

Full ARIMA code

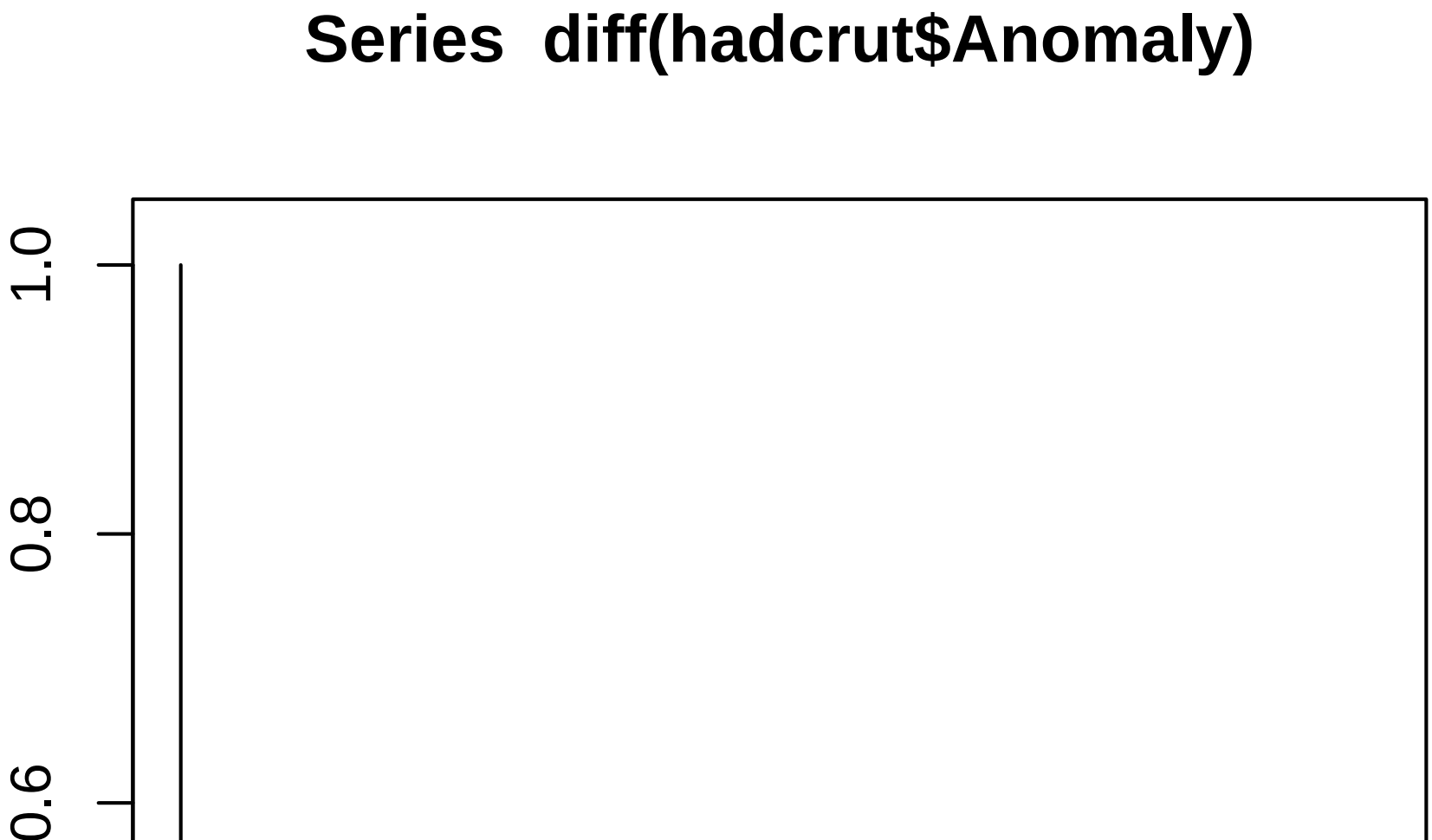
```
model_code = '
model
{
  # Set up residuals
  for(t in 1:q) {
    eps[t] <- z[t] - alpha
  }
  # Likelihood
  for (t in (q+1):T) {
    z[t] ~ dnorm(alpha + ar_mean[t] + ma_mean[t], tau)
    ma_mean[t] <- inprod(theta, eps[(t-q):(t-1)])
    ar_mean[t] <- inprod(phi, z[(t-p):(t-1)])
    eps[t] <- z[t] - alpha - ar_mean[t] - ma_mean[t]
  }

  # Priors
  alpha ~ dnorm(0.0,0.01)
  for (i in 1:q) {
```

Example: hadcrut data

```
hadcrut = read.csv('https://raw.githubusercontent.com/andre
par(mfrow=c(1,2))
with(hadcrut,plot(Year,Anomaly,type='l'))
with(hadcrut,plot(Year[-1],diff(Anomaly),type='l'))
```

Look at the ACF/PACF

```
par(mfrow=c(1,2))
acf(diff(hadcrut$Anomaly))
pacf(diff(hadcrut$Anomaly))
```



Fitting the model ARIMA(3,1,3)

```
d = 1
real_data = with(hadcrut,
                 list(T = nrow(hadcrut)-d,
                      z = diff(hadcrut$Anomaly, differences
                      q = 3,
                      p = 3))

model_parameters =  c("alpha","theta","phi","sigma")

real_data_run = jags(data = real_data,
                     parameters.to.save = model_parameters,
                     model.file=textConnection(model_code),
                     n.chains=4,
                     n.iter=1000,
                     n.burnin=200,
                     n.thin=2)
```

Checking the output

```
print(real_data_run)
```

```
## Inference for Bugs model at "5", fit using jags,
##  4 chains, each with 1000 iterations (first 200 discarde
##  n.sims = 1600 iterations saved
##              mu.vect sd.vect     2.5%      25%      50%
## alpha          0.010   0.007   -0.003    0.005    0.009
## phi[1]        -0.087   0.205   -0.467   -0.238   -0.086
## phi[2]        -0.305   0.285   -0.831   -0.504   -0.317
## phi[3]        -0.335   0.428   -1.026   -0.645   -0.388
## sigma          0.104   0.006    0.093    0.100    0.104
## theta[1]      -0.184   0.226   -0.613   -0.348   -0.175
## theta[2]      -0.070   0.382   -0.684   -0.398   -0.080
## theta[3]      -0.023   0.438   -1.043   -0.268    0.030
## deviance -274.736   4.353 -282.659 -277.362 -275.047 -27
##              Rhat n.eff
## alpha       1.008   440
## phi[1]      1.006  1600
```

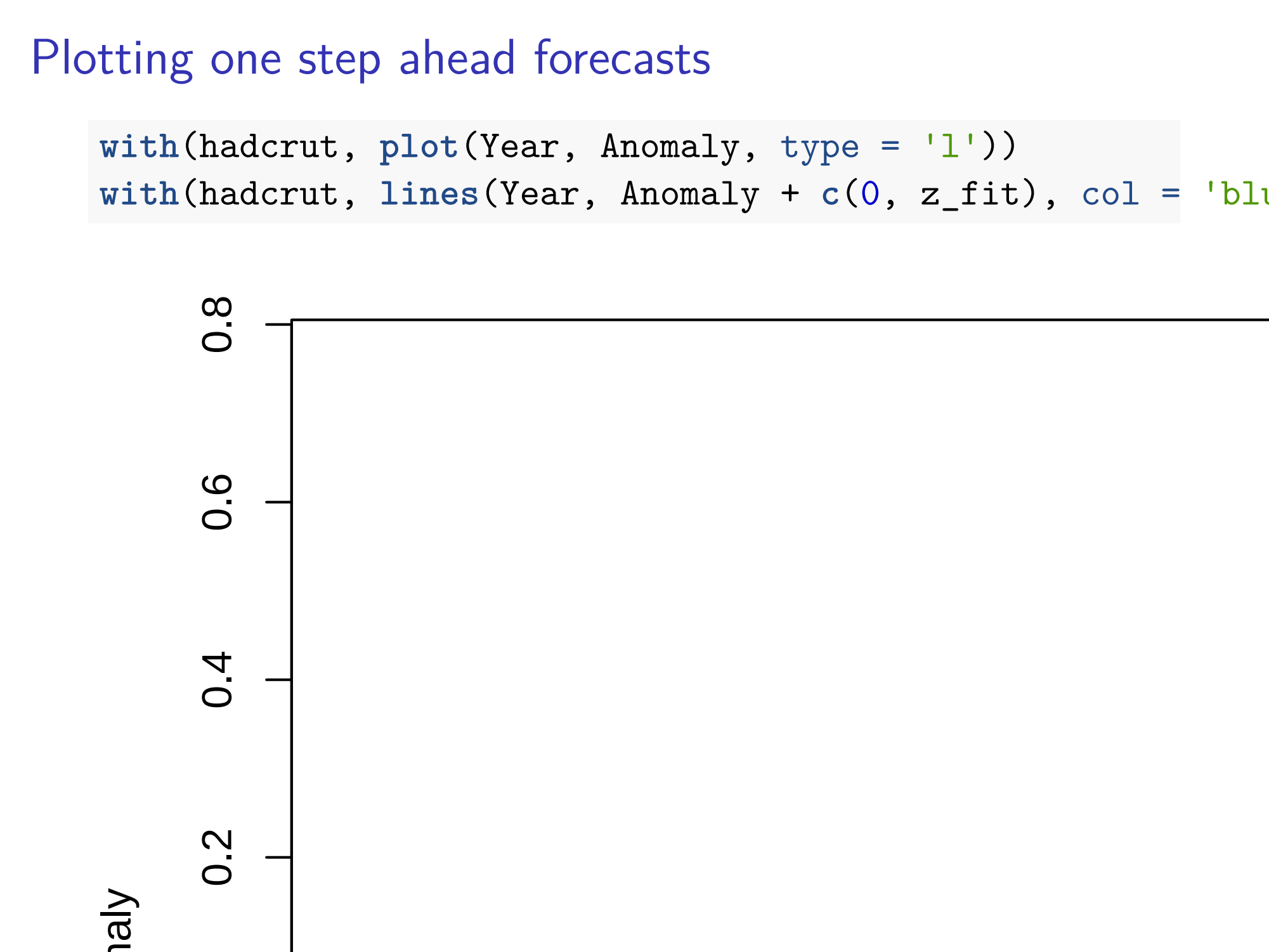
Predicting the future

We can use the structure of the model to create *one step ahead* forecasts:

```
# Get posterior means
post = real_data_run$BUGSoutput$sims.list
alpha_mean = mean(post$alpha)
theta_mean = apply(post$theta,2,'mean')
phi_mean = apply(post$phi,2,'mean')

# Create forecasts
z = diff(hadcrut$Anomaly, differences = d)
eps_fit = z_fit = rep(NA,real_data$T)
eps_fit[1:real_data$q] = z[1:real_data$q] - alpha_mean
z_fit[1:real_data$q] = alpha_mean
for (t in (real_data$q+1):real_data$T) {
  ar_mean = sum( phi_mean * z[(t-real_data$p):(t-1)] )
  ma_mean = sum( theta_mean * eps_fit[(t-real_data$q):(t-1)
  eps_fit[t] = z[t] - alpha_mean - ma_mean - ar_mean
  z_fit[t] = alpha_mean + ma_mean + ar_mean
```

Plotting one step ahead forecasts

```
with(hadcrut, plot(Year, Anomaly, type = 'l'))
with(hadcrut, lines(Year, Anomaly + c(0, z_fit), col = 'blu
```



Summary

- MA(q) models are used to create future forecasts based on the error in the previous forecasts
- ARMA models combine AR and MA ideas together
- ARIMA models further add in differencing
- All of the above can be fitted in JAGS
- It turns out that it is much simpler and easier to create all the forecasts in JAGS too. More on this later