# Class 1: Including covariates: ARIMAX models

Andrew Parnell
andrew.parnell@ucd.ie

# Learning outcomes

- Be able to add on components to ARIMA models
- Understand the issues with fitting ARIMAX and other extensions to ARIMA models
- Understand how to create forecasts and accuracy measures with `forecast`
- More on model choice
- Know the basics of forecast calibration and scoring rules

# Bolting together models

- As we have already seen we can combine bits of models together, such as RW, AR and MA, into ARIMA
- The `forecast` package in R can fit these models really fast
- Unfortunately we need to be able to fit these models by hand (e.g. in JAGS or Stan) to create really interesting and complicated models
- The one extra really useful part of `forecast` is the ability to be able to add in covariates

# The ARIMAX framework

- The ARIMAX framework (ARIMA with eXplanatory variables) is just another extension to the ARIMA framework
- The basic ARIMAX model for a possible differenced series $z_t$ is:

$$z_t \sim N(\alpha + \text{AR terms} + \text{MA terms} + \phi_1 x_{1t} + \ldots + \phi_r x_{rt}, \sigma^2)$$

  where we now include $r$ possible explanatory variables with coefficients $\phi_1, \ldots, \phi_r$

# Warnings about ARIMAX models

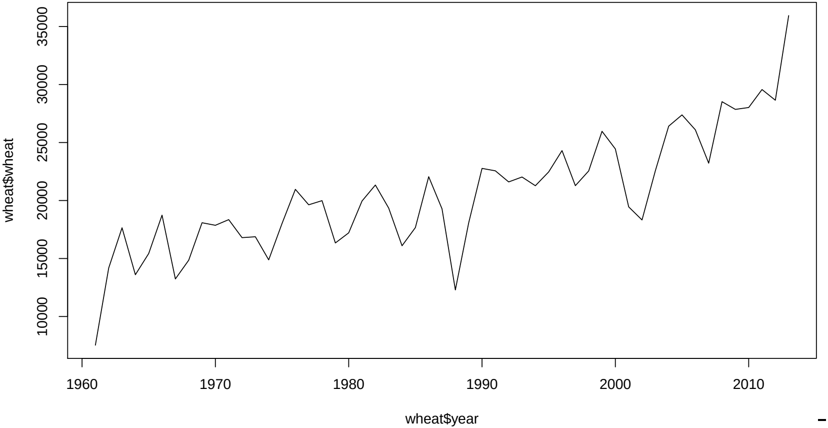
There are two key things to be wary of when using this type of ARIMAX model:

1. It's hard to interpret the $\phi$ values. It is not the case (as in normal regression) that an increase of 1 unit in $x$ will lead to an increase of $\phi$ in $y$ because of all the AR terms.
2. If you are differencing the data before running the model, you also need to difference the explanatory variables

If you're just interested in forecasting then the problem in 1 goes away, but if you are interested in the causation of $x$ on $y$ you can fit the regression model separately or try a dynamic regression model (see later in course)

# An ARIMAX model for the wheat data

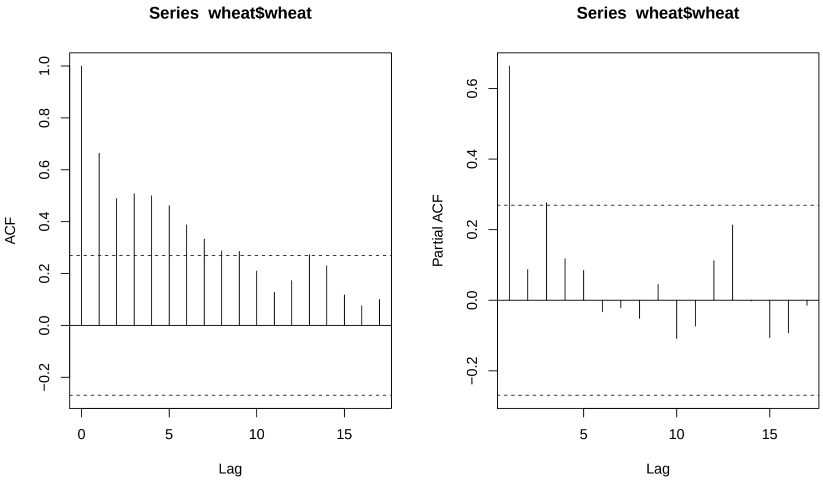
```
wheat = read.csv('../../data/wheat.csv')
plot(wheat$year, wheat$wheat, type = 'l')
```



Let's see if we can fit a time series model with year as the explanatory variable

# ACF/PACF plots

```
par(mfrow = c(1, 2))
acf(wheat$wheat)
pacf(wheat$wheat)
```



# A first ARIMAX model

- Try ARIMAX(1, 1, 1)

```
Arima(wheat$wheat, order = c(1, 1, 1), xreg = wheat$year)
```

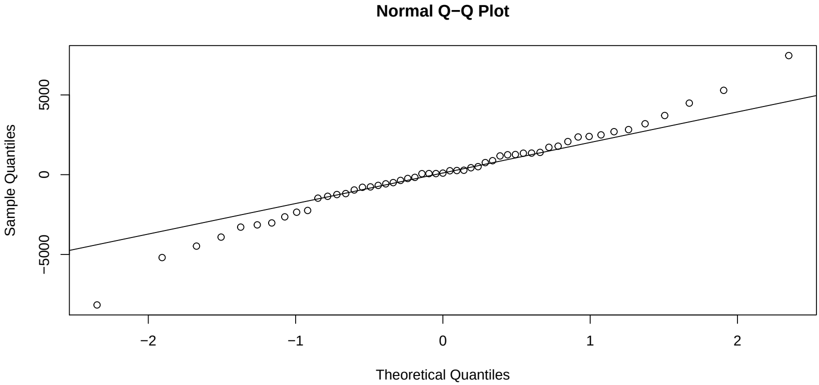
```
## Series: wheat$wheat
## ARIMA(1,1,1)
##
## Coefficients:
##          ar1      ma1  wheat$year
##       0.3832  -1.0000    296.6736
## s.e.  0.1564   0.0768     39.1763
##
## sigma^2 estimated as 7550790:  log likelihood=-485.58
## AIC=979.15   AICc=980   BIC=986.96
```

```
## Compare with:
Arima(wheat$wheat, order = c(1, 1, 1))$aic
```

```
## [1] 985.4555
```

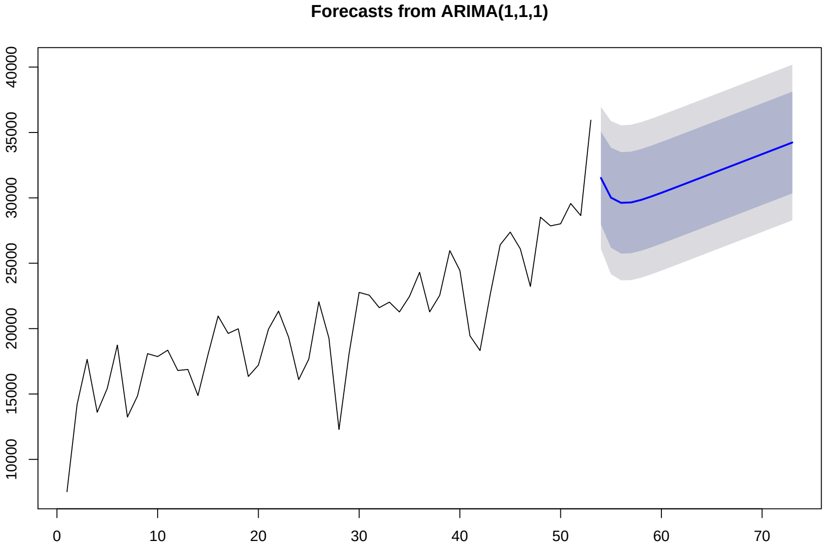
# Checking the residuals

```
my_model_ARIMAX111 = Arima(wheat$wheat,
                           order = c(1, 1, 1),
                           xreg = wheat$year)
qqnorm(my_model_ARIMAX111$residuals)
qqline(my_model_ARIMAX111$residuals)
```



# Predictions

```
plot(forecast(my_model_ARIMAX111, xreg = 2014:2033, h=20))
```



# A cheat way of skipping model choice

- The `forecast` package has a cheat function which will fit *all* of the possible ARIMA models for you and report the best one. It's called `auto.arima`

```
auto.arima(wheat$wheat, xreg = wheat$year)
```

```
## Series: wheat$wheat
## ARIMA(3,0,0) with non-zero mean
##
## Coefficients:
##          ar1      ar2     ar3  intercept
##       0.5833  -0.4929  0.3077  -567234.4
## s.e.  0.1577   0.1564  0.1570    74936.5
##       wheat$year
##         295.8574
## s.e.     37.7267
##
## sigma^2 estimated as 6388181:  log likelihood=-488.18
```

# Model choice

# Choosing different models: AICc and BIC

- So far we have just been using AIC to choose between models
- AIC is defined for an ARIMA model as:

$$AIC = -2 \log L + 2(p + q + 1)$$

- The forecast package also reports the Bayesian Information Criterion (BIC) which is:

$$BIC = -2 \log L + (p + q + 1) \log n$$

  where $n$ is the number of data points (after differencing)
- It also reports the ‘corrected’ AIC (AICc) value which is a very slight variation on the standard AIC for use with smaller sample sizes

# Measuring model complexity

- These information criteria all work by adding on a function of the number of parameters to the deviance, designed to approximate some performance criterion
- AIC was invented to match leave-one-out cross validation error (more on this later). BIC to match the probability of the data given the model
- The version JAGS uses is known as the Deviance Information Criterion (DIC) and is built specifically to penalise the deviance by the *effective* number of parameters, which it calls $p_D$
- The version Stan uses is known as the Wanatabe Akaike Information Criterion (WAIC) and use a different method to estimate an effective number of parameters

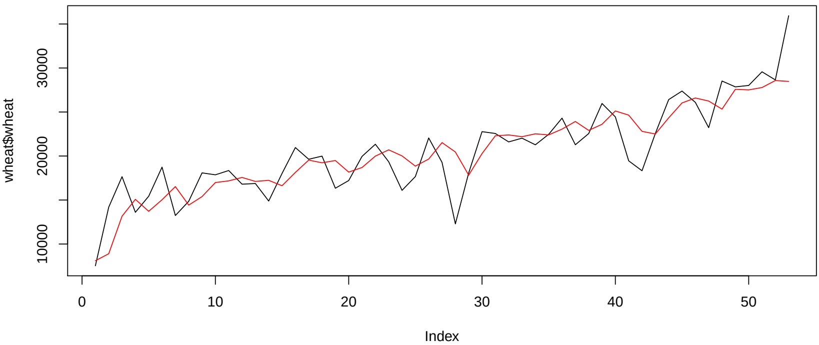
# An alternative; cross-validation

- Often the gold standard by which time series models are judged as how well they forecast future values of the series
- Without waiting for more data to become available, we can remove some of the data points at the end of the series, fit the model, and forecast into the future. This is leave one out cross validation or LOO-CV
- This is very computationally intensive as we have to re-fit the model and get new parameter estimates at every step

# Leave none out cross validation

- We can get a cheat version of LOO-CV by just using the fitted values from the ARIMA model fit.
- The `Arima` function stores the one step ahead forecasts in the object `fitted`:

```
plot(wheat$wheat, type = 'l')
lines(fitted(my_model_ARIMAX111), col = 'red')
```



# Accuracy measures from `forecast`

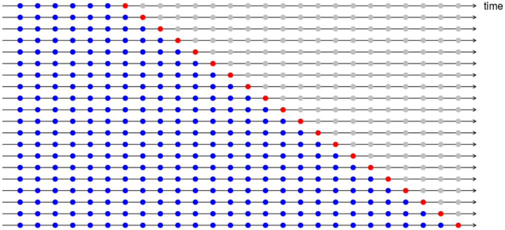
- For the fitted values you can also measures of accuracy such as root mean square error
- These are all calculated by comparing the fitted values with the forecasted values
- It actually provides way more:

```r
accuracy(my_model_ARIMAX111)
```

```
##                    ME     RMSE      MAE
## Training set 97.76983 2642.143 1937.182
##                     MPE     MAPE      MASE
## Training set -0.8540633 10.43452 0.7651624
##                   ACF1
## Training set 0.1271498
```

# Proper LOO-CV

- The `forecast` package has a function called `CVar` which implements leave one out cross validation. However it only works for AR models - not full ARIMA ones
- If you want to do ARIMA LOO-CV (sometimes called rolling-origin forecasting) you have to write it yourself.



(from `https://robjhyndman.com/hyndsight/tscv/`)

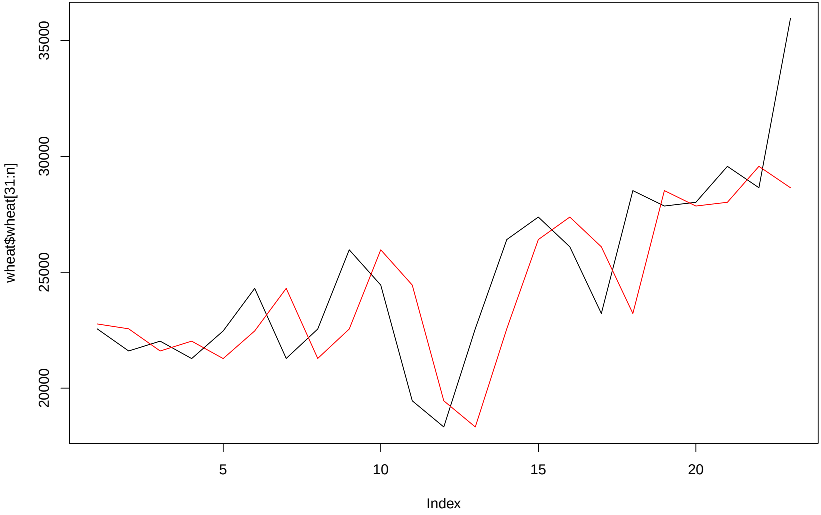
# Implementing loo-CV for an ARIMAX model

```r
n_min = 30 # minimum length for model
n = nrow(wheat)
mae = forecasts = rep(NA, n - n_min)
# Loop through time series
for(i in 1:(n-n_min)) {
  # Fit to the training set
  curr_model = auto.arima(wheat$wheat[1:(i + n_min - 1)])
  # Create 1 step ahead forecasts
  forecasts[i] = forecast(curr_model, h = 1)[['mean']]
  # Get mean absolute error
  mae[i] = abs(forecasts[i] - wheat$wheat[i + n_min])
}
mean(mae)
```

```
## [1] 2166.893
```

# Plotting the forecasts

```
plot(wheat$wheat[31:n], type = 'l')
lines(forecasts, col = 'red')
```



# Forecasting and scoring rules

- A common mantra in time series forecasting is to aim for *sharpness under calibration*
- Sharpness refers to the variance of the forecast. A *sharp* forecast is one with a low variance
- However, for a forecast to be useful, it needs to be *calibrated*. This means that if you predict a 20% chance of rain, it should rain on 20% of those days. A sharp forecast is only useful if it is calibrated
- Often forecasters use *scoring rules* to evaluate whether a forecast is calibrated or not. This is a very broad issue and beyond the remit of this course

# Summary

- We now know how to incorporate explanatory variables in ARIMA models (and we also know the pitfalls of doing so)
- We know how to compare models using AIC, AICc, BIC and cross validation
- We learnt how to create forecast accuracy
- We've learnt a little bit about forecast calibration