Bivariate Time Series Analysis

Jan Rovny

3/31/2020

Introduction

In the previous class we learned about processes in a single time-series. Most importantly, we learned about *stationarity* – whether a distribution of time-series segments are comparable across the time-series, and about *dependence* – whether the series is independent of previous observations. We also learned about four key types of time-series processes – white noise, moving average, autoregression, and integration. All these lessons will be useful to us as we move on to consider bivariate time-series in this class.

In this lesson, we will learn how to assess associations between time-series. This will bring us to consider substantively interesting questions about causal flow between time-series variables, studying whether and how one series x_t leads to y_t.

To assess the causal flow $x_t \rightarrow y_t$ we can consider three possible approaches:

- Prewhitening
- Granger causality
- Error correction models of cointegration

Prewhitening

If we have two time-series that are white noise processes, we can see their association (causation) by looking at how the two series are correlated at different lags and leads (forward steps in the time-series). It is important to note that causation is by definition asymmetrical. If $x \rightarrow y$, then it is not true that $y \rightarrow x$. Time-series offers a particular advantage in the study of causality, in that it allows us to see the order in which series respond to, or are associated with, one another. Causal order implies that the predictor 'moves' first, and the dependent variable 'responds' afterwards. That implies that lagged values of x (that is x_{t-1}) are be correlated with y (that is y_t), but not the reverse.

As we saw in the last lesson, real world time-series, however, have various error aggregation processes, such as autoregression, moving average, or integration (ARIMA). In order to observe the causal dynamics between time-series, we need to remove the 'error filter', that is, we need to first model the ARIMA processes in our series, in order to observe x and y as white noise. This process is called *prewhitening*. Then, we can assess any causal association between the series.

Let's take a look in R at some real time-series. First, let's call up the necessary libraries, and load the data from the course website:

```
library(rio) #for importing data
library(tseries) #for stationarity test
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
D<-import("https://jan-rovny.squarespace.com/s/varexample.dta")
head(D)
```

```
##   date fedfunds inflation   unrate
## 1    0 3.933333 1.6881123 5.133333
## 2    1 3.696667 1.2506870 5.233333
## 3    2 2.936667 1.4828017 5.533333
## 4    3 2.296667 1.1138498 6.266666
## 5    4 2.003333 0.9187145 6.800000
## 6    5 1.733333 0.8953280 7.000000
```

This dataset provides time-series information about inflation rate `D$inflrate` and unemployment rate `D$unrate`. Let's test whether there is a relationship between them, and what the causal direction is. First, let's declare these variables as time-series in R:

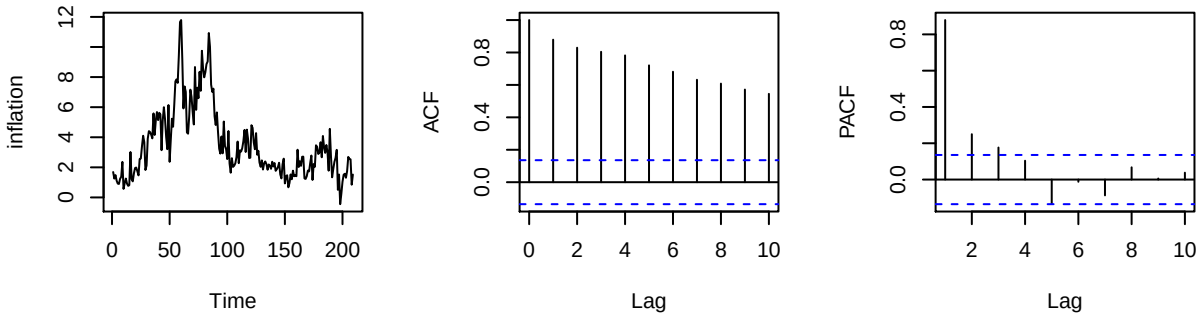
```
ts(D$inflation) #declare time series
ts(D$unrate) #declare time series
```

Identification

Next, we need to fit a correct ARIMA model to both series. Referring to what we learned in the introduction to time-series, to do this, we must first identify, and then estimate the ARIMA structure in both series. Let's start with identification:

```
#1 Identification
par(mfrow = c(2, 3)) #get 2 by 3 graph window
plot.ts(D$inflation, ylab = "inflation", main="")
acf(D$inflation, lag.max = 10, ylab="ACF", main="")
pacf(D$inflation, lag.max = 10, ylab = "PACF", main="") #AC declines slowly, looks might have a trend,
adf.test(D$inflation) #fail to reject null, suggests unit root and integration
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  D$inflation
## Dickey-Fuller = -2.6941, Lag order = 5, p-value = 0.2851
## alternative hypothesis: stationary
```

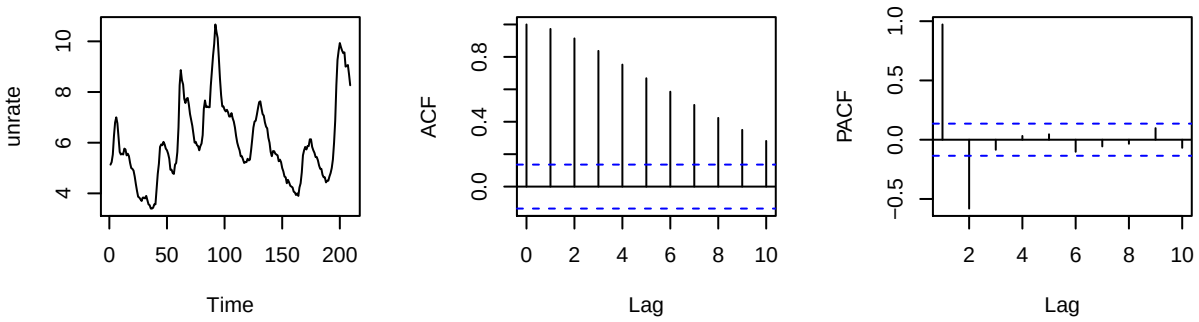


The AC function above declines slowly, which suggests that we might have a trend. This is supported by the Dickey-Fuller test, where we fail to reject the null of stationarity. The PAC function suggests an AR1 process. We should thus model an ARIMA(1,1,0) for the inflation series. Let's look at unemployment rate:

```
#1 Identification
par(mfrow = c(2, 3)) #get 2 by 3 graph window
plot.ts(D$unrate, ylab = "unrate", main="")
acf(D$unrate, lag.max = 10, ylab="ACF", main="")
```

```
pacf(D$unrate, lag.max = 10, ylab = "PACF", main="")
adf.test(D$inflation) #fail to reject null, suggests unit root and integration
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  D$inflation
## Dickey-Fuller = -2.6941, Lag order = 5, p-value = 0.2851
## alternative hypothesis: stationary
```



Unemployment seems to be following a similar pattern as inflation, suggesting an AR(1) with integration. We should thus model its error aggregation process as ARIMA(1,1,0).

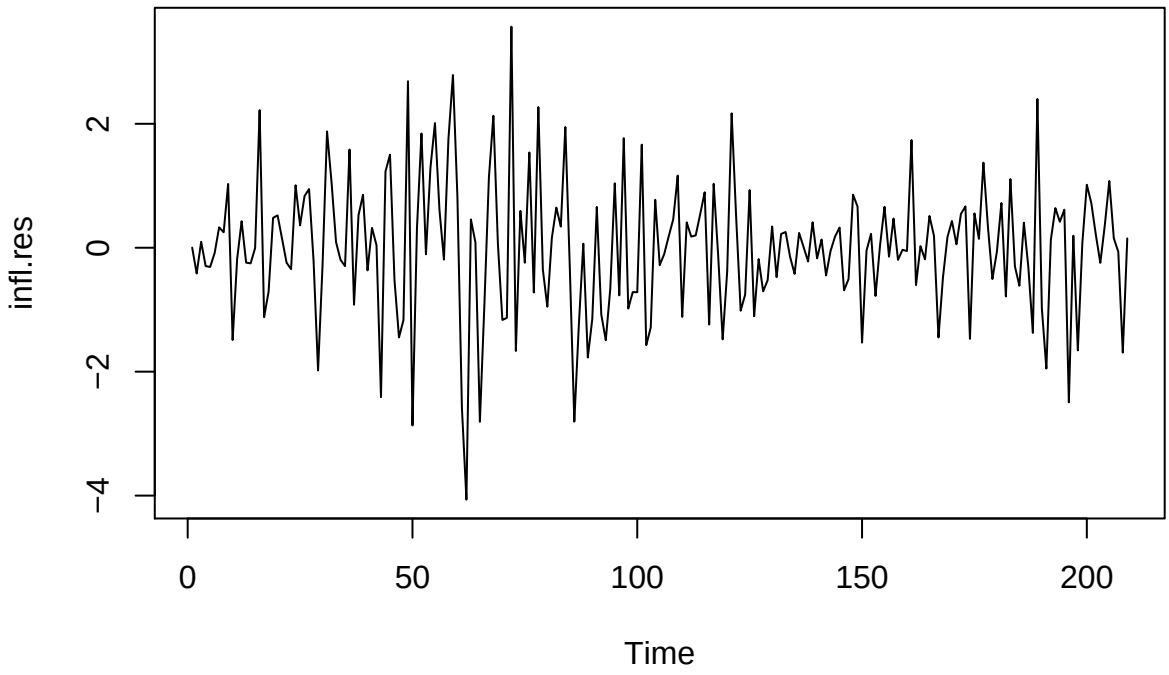
Estimation

Having identified both time-series processes, let's now estimate the appropriate ARIMA models, and capture the residuals left behind – these should now be white noise.

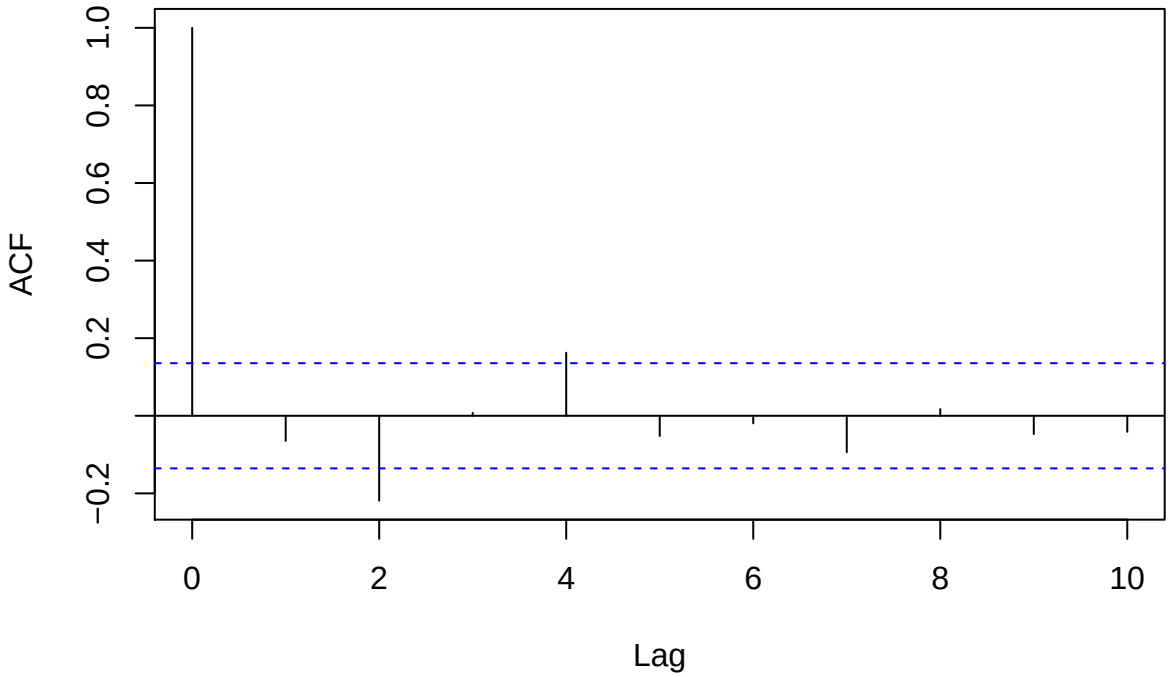
```
#a) Inflation
model.inflation<-arima(D$inflation, order=c(1,1,0)) #model error aggregation filter
model.inflation
```

```
##
## Call:
## arima(x = D$inflation, order = c(1, 1, 0))
##
## Coefficients:
##           ar1
##       -0.3100
## s.e.   0.0658
##
## sigma^2 estimated as 1.197:  log likelihood = -313.86,  aic = 631.71
```

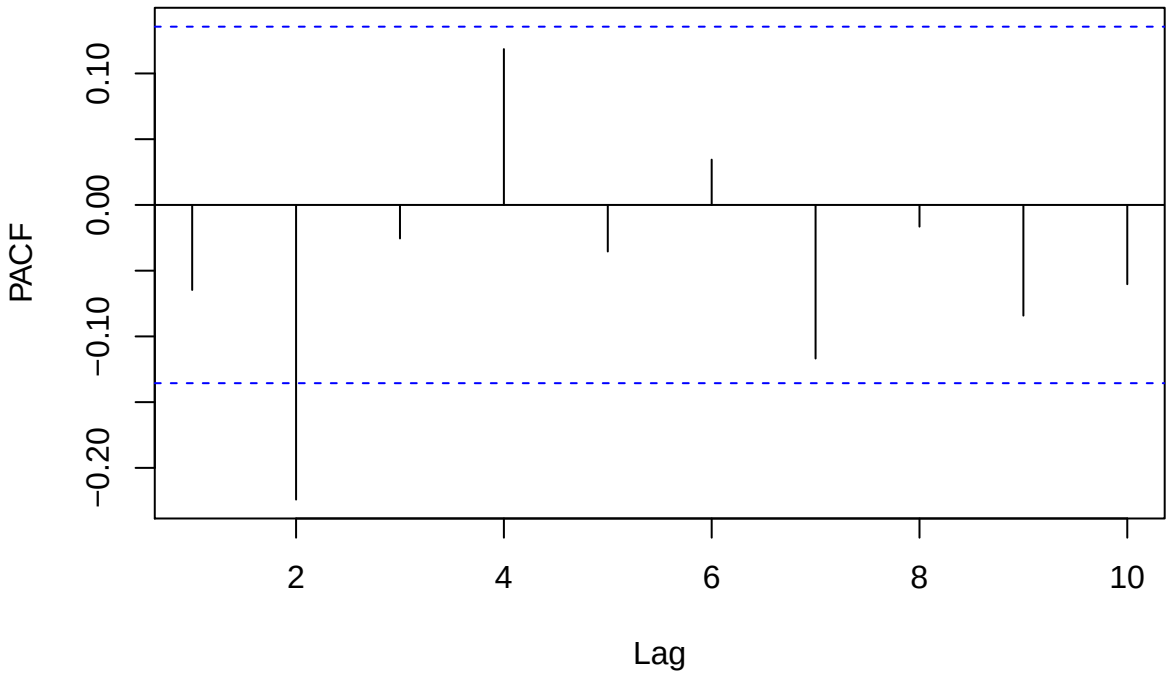
```
infl.res<-model.inflation$residuals #take residuals from  model, should be white noise
#check residulas
ts.plot(infl.res)
```




```
acf(infl.res, lag.max = 10, ylab="ACF", main="")
```




```
pacf(infl.res, lag.max = 10, ylab = "PACF", main="")
```




```
adf.test(infl.res) #residuals are stationary, and graphs look like white noise !!!
```

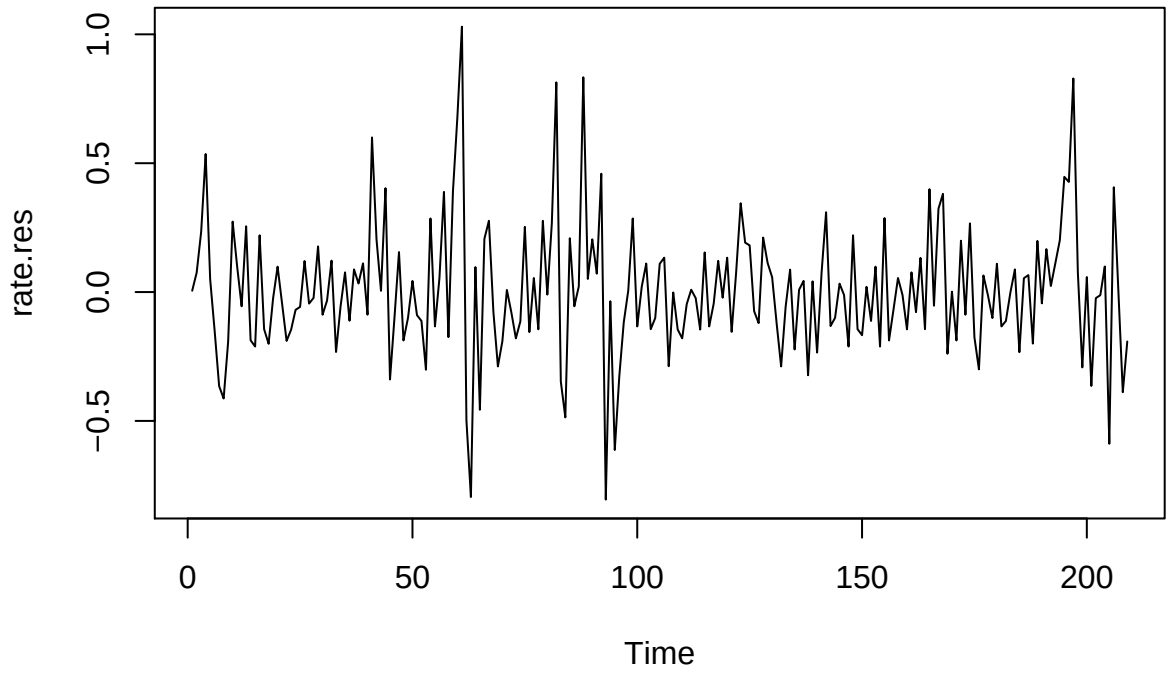
```
## Warning in adf.test(infl.res): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  infl.res
## Dickey-Fuller = -5.7638, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

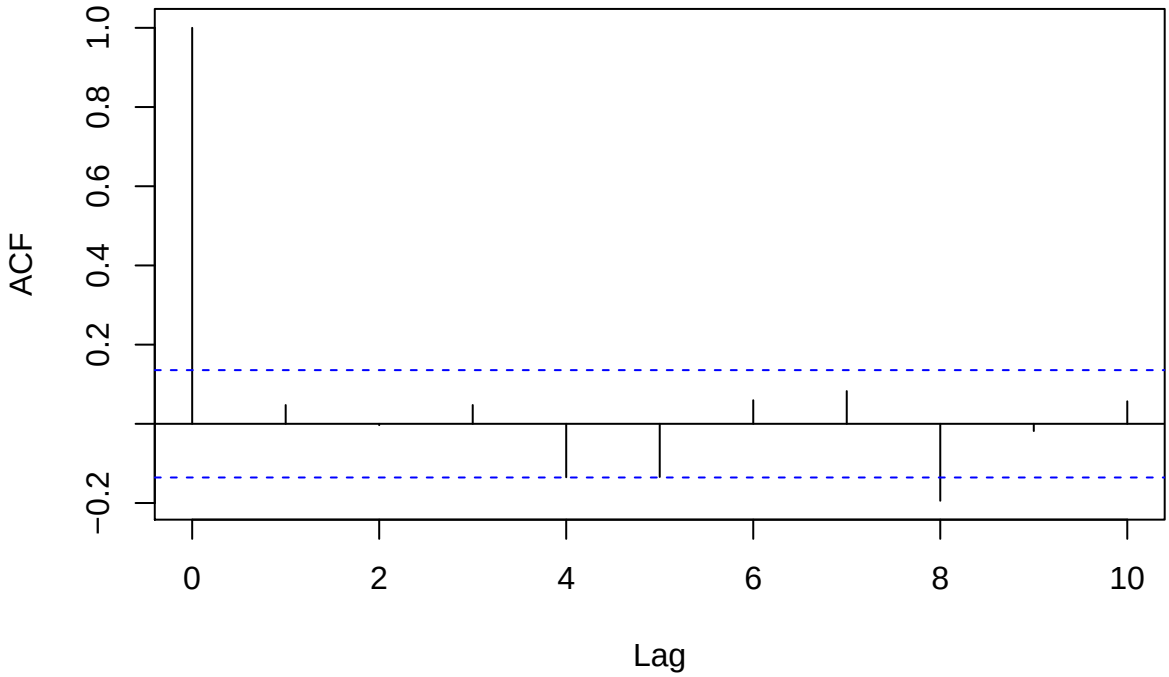
```
#b) unemployment rate
model.unrate<-arima(D$unrate, order=c(1,1,0)) #model error aggregation filter
model.unrate
```

```
##
## Call:
## arima(x = D$unrate, order = c(1, 1, 0))
##
## Coefficients:
##          ar1
##       0.6589
## s.e.  0.0520
##
## sigma^2 estimated as 0.06525:  log likelihood = -11.56,  aic = 27.11
```

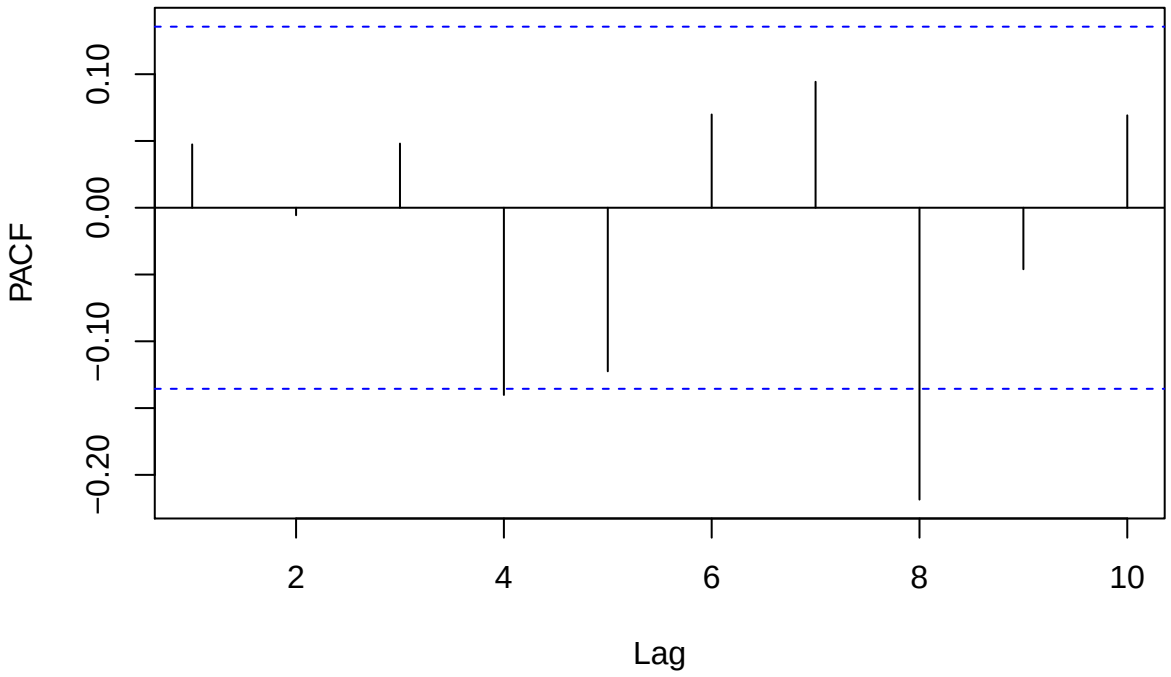
```
rate.res<-model.unrate$residuals #take residuals from model, should be white noise
#check residulas
ts.plot(rate.res)
```




```
acf(rate.res, lag.max = 10, ylab="ACF", main="")
```




```
pacf(rate.res, lag.max = 10, ylab = "PACF", main="")
```




```
adf.test(rate.res) #residuals of the model are stationary, and graphs look like white noise !!!
```

```
## Warning in adf.test(rate.res): p-value smaller than printed p-value
```

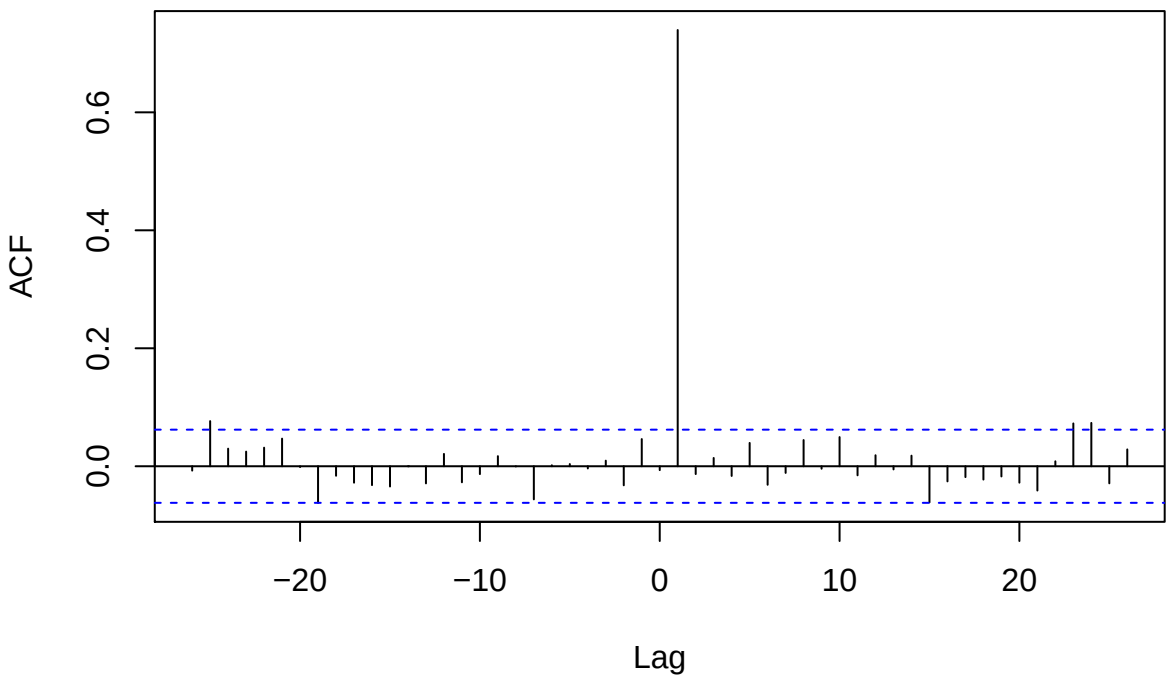
```
##
##  Augmented Dickey-Fuller Test
##
## data:  rate.res
## Dickey-Fuller = -6.2029, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

Wonderful, our results above suggest that we have managed to estimate the error aggregation processes in inflation and unemployment rate correctly. The residuals of these are white noise. We have thus removed the error aggregation filter from these series.

Cross-correlation

We can now check the causal association between our two series. We will do this by cross-correlating them. Before we do that, let's look at the logic of cross-correlation. We define a random variable x and then a variable y, which is a function of the first lag of x plus some random error, mathematically: $y_t = x_{t-1} + e_t$. Then, let's look at the cross-correlation function between x and y:

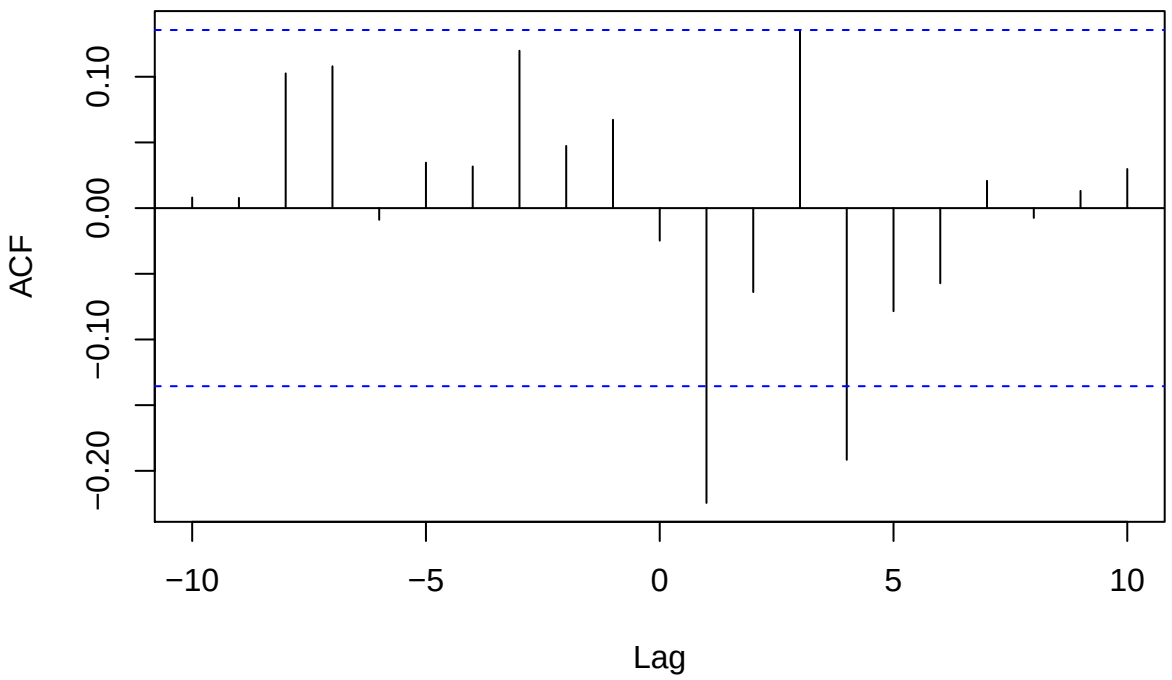
```
x<-rnorm(1000,0,1) #random x
y<-lag(x)+rnorm(1000,0,1) #y as function of lag of x plus error
ccf(x,y) #cross-correlation function
```



The cross-correlation function above shows a positive effect of x on y in the first lag (that is at $t-1$). This means that the as x increases, y will increase one period later. (Note that negative lags – on the left side of the graph above – are leads, or steps forward in the series.)

Now, knowing the logic of cross-correlation, let's finally look at the relationship between inflation and unemployment:

```
par(mfrow = c(1, 1)) #make plot be 1 by 1
ccf(infl.res, rate.res, lag.max = 10) #CCF = Cross-Correlation Function
```



The cross-correlation function above shows negative 1st and 4th lag effect. This means that unemployment lags inflation in the 1st and the 4th periods. This means that higher values of inflation cause lower values of unemployment 1 and 4 periods later. Note that if the lags were negative (=leads), it would suggest the reversed causal flow. We can thus conclude that inflation has a negative causal effect on unemployment. As inflation increases, unemployment decreases.

One final word of caution: the prewhitening process is data driven. This means that we are letting data speak without theory... You should always view this with suspicion!

Granger Causality

Granger causality is another approach to studying relationships between two time-series. Granger causality rests on the fundamental dual logic: First, series y will be a function of its own past. Second, if x causes y it must be the lags of x, that is x_{t-1} and so on, cause y now, that is y_t. Logically then, adding past information of x to predict y must improve our model.

Mathematically, this dual logic is expressed in the following way:

1) y_t is a function if its own past: $y_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + ... + \beta_k y_{t-k} + e_t$
2) If x causes y, then the past of x must improve our prediction of y_t: $y_t = \beta_0 + \beta_1 y_{t-1} + ... + \beta_k y_{t-k} + \gamma_1 x_{t-1} + ... + \gamma_k x_{t-k} + e_t$

In order to conclude that the causal flow goes indeed from x to y, and not the other way around, the Granger causality approch then reverses the model, seeing if the past of y predicts the present of x.

In R, we can use the `grangertest()` function in the `lmtest` package to fit Granger models. Let's see if we can assess the causal flow in the arms race between India and Pakistan. The following code loads in data on the military spending in these two countries over time.

```
A<-as.data.frame(matrix(c(1988  ,   7941    ,   2500    ,
          1989  ,   8161    ,   2499    ,
          1990  ,   8051    ,   2636    ,
          1991  ,   7532    ,   2823    ,
          1992  ,   7209    ,   2997    ,
          1993  ,   8137    ,   2993    ,
          1994  ,   8109    ,   2917    ,
          1995  ,   8340    ,   2965    ,
          1996  ,   8565    ,   2961    ,
          1997  ,   9307    ,   2837    ,
          1998  ,   9387    ,   2833    ,
          1999  ,   10482   ,   2858    ,
          2000  ,   10900   ,   2867    ,
          2001  ,   11397   ,   3071    ,
          2002  ,   11426   ,   3304    ,
          2003  ,   12394   ,   3350), ncol=3, byrow=T))
#change variable names in dataframe
library(data.table)
setnames(A, "V1", "year")
setnames(A, "V2", "India")
setnames(A, "V3", "Pakistan")
```

Now, let us assess which country's military spending leads to the other using Granger causality. We first model India's military spending as a function of Pakistan's.

```
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
grangertest(India~Pakistan,order=4, data=A)
```

```
## Granger causality test
##
## Model 1: India ~ Lags(India, 1:4) + Lags(Pakistan, 1:4)
## Model 2: India ~ Lags(India, 1:4)
##   Res.Df Df      F Pr(>F)
## 1      3
## 2      7 -4 2.1548 0.2772
```

The results above effectively compare two models. In the first we are predicting India's spending with 'order' lags of India's spending and lags of Pakistan's spending. The `order` function sets how many lags we should consider. In the second model, we are predicting India's spending only with the lags of India's spending. The final F-test shows us whether the additional information about Pakistan's military spending in model one improves our prediction of India's spending. If the p-value is significant, it does, if it is insignificant, it does not. Here, the p-value is insignificant, and so we conclude that Pakistan's military spending does not seem to cause India's. Now, let's turn the test around:

```
grangertest(Pakistan~India,order=4, data=A)
```

```
## Granger causality test
```

```
##
## Model 1: Pakistan ~ Lags(Pakistan, 1:4) + Lags(India, 1:4)
## Model 2: Pakistan ~ Lags(Pakistan, 1:4)
##   Res.Df Df      F  Pr(>F)
## 1      3
## 2      7 -4 23.976 0.01297 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now we are testing whether Pakistan's spending is caused by India's. Seeing a significant p-value on the F-test, we conclude that yes, it does. Given the information from the two tests, we can conclude that the causal order in this arms race is India→Pakistan!

Multiple Granger Causality

The logic of Granger causality is very applealing. We can use models based on this logic to let data atheoretically speak for themselves, testing relationships between many variables. This can be done using so-called *Vector Autoregressive* (VAR) modelling. It effectively means that we put all variables on both sides of the equation... which is an estimation travesty...

Let's go back to our D data frame with information in inflation, unemployment, and federal funds. We can estimate the effect of all these variables on all variables using vector autoregression applied in the `VAR()` function in the `vars` library in R.

```
#create a frame with variables of interest
dat<-cbind(D$fedfunds,D$inflation,D$unrate)

#estimate the effects of all variables on all variables
library(vars)
```

```
## Loading required package: MASS
## Loading required package: strucchange
## Loading required package: sandwich
## Loading required package: urca
```

```
est <- VAR(dat, p = 4, type = "const", season = NULL, exog = NULL) #p sets number of lags
```

```
## Warning in VAR(dat, p = 4, type = "const", season = NULL, exog = NULL): No column names supplied in y
```

```
summary(est)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: y1, y2, y3
## Deterministic variables: const
## Sample size: 205
## Log Likelihood: -512.453
## Roots of the characteristic polynomial:
## 0.9671 0.9671 0.8455 0.8455 0.6634 0.6634 0.5359 0.4968 0.4968 0.4521 0.3271 0.3271
## Call:
## VAR(y = dat, p = 4, type = "const", exogen = NULL)
##
##
## Estimation results for equation y1:
```

```
## ====================================
## y1 = y1.l1 + y2.l1 + y3.l1 + y1.l2 + y2.l2 + y3.l2 + y1.l3 + y2.l3 + y3.l3 + y1.l4 + y2.l4 + y3.l4 +
##
##       Estimate Std. Error t value Pr(>|t|)
## y1.l1  1.06019    0.07750  13.680  < 2e-16 ***
## y2.l1  0.08573    0.05769   1.486 0.138937
## y3.l1 -1.12743    0.26007  -4.335 2.35e-05 ***
## y1.l2 -0.40844    0.10959  -3.727 0.000255 ***
## y2.l2  0.14483    0.06411   2.259 0.024995 *
## y3.l2  1.43954    0.47438   3.035 0.002742 **
## y1.l3  0.34213    0.10881   3.144 0.001929 **
## y2.l3 -0.05812    0.06483  -0.896 0.371131
## y3.l3 -0.70204    0.47863  -1.467 0.144072
## y1.l4 -0.07427    0.07581  -0.980 0.328481
## y2.l4 -0.04398    0.06005  -0.732 0.464899
## y3.l4  0.35432    0.26123   1.356 0.176572
## const  0.22240    0.26021   0.855 0.393782
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.8318 on 192 degrees of freedom
## Multiple R-Squared: 0.9467,  Adjusted R-squared: 0.9434
## F-statistic: 284.3 on 12 and 192 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation y2:
## ====================================
## y2 = y1.l1 + y2.l1 + y3.l1 + y1.l2 + y2.l2 + y3.l2 + y1.l3 + y2.l3 + y3.l3 + y1.l4 + y2.l4 + y3.l4 +
##
##        Estimate Std. Error t value Pr(>|t|)
## y1.l1  0.153106   0.095926   1.596  0.11211
## y2.l1  0.496651   0.071412   6.955  5.4e-11 ***
## y3.l1 -0.864569   0.321911  -2.686  0.00787 **
## y1.l2 -0.056262   0.135653  -0.415  0.67879
## y2.l2  0.145957   0.079348   1.839  0.06739 .
## y3.l2  1.217548   0.587175   2.074  0.03945 *
## y1.l3  0.003771   0.134678   0.028  0.97769
## y2.l3  0.119329   0.080247   1.487  0.13865
## y3.l3 -0.309208   0.592437  -0.522  0.60232
## y1.l4 -0.113976   0.093842  -1.215  0.22603
## y2.l4  0.198238   0.074333   2.667  0.00831 **
## y3.l4 -0.143695   0.323339  -0.444  0.65725
## const  0.832060   0.322083   2.583  0.01053 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 1.03 on 192 degrees of freedom
## Multiple R-Squared: 0.823,   Adjusted R-squared: 0.8119
## F-statistic: 74.39 on 12 and 192 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation y3:
```

```
## ===================================
## y3 = y1.l1 + y2.l1 + y3.l1 + y1.l2 + y2.l2 + y3.l2 + y1.l3 + y2.l3 + y3.l3 + y1.l4 + y2.l4 + y3.l4 +
##
##        Estimate Std. Error t value Pr(>|t|)
## y1.l1  0.004900   0.022778   0.215    0.830
## y2.l1  0.013957   0.016957   0.823    0.411
## y3.l1  1.599911   0.076441  20.930  < 2e-16 ***
## y1.l2  0.045633   0.032212   1.417    0.158
## y2.l2 -0.002500   0.018842  -0.133    0.895
## y3.l2 -0.576848   0.139430  -4.137 5.26e-05 ***
## y1.l3 -0.036193   0.031981  -1.132    0.259
## y2.l3  0.014368   0.019055   0.754    0.452
## y3.l3 -0.022801   0.140679  -0.162    0.871
## y1.l4  0.001066   0.022284   0.048    0.962
## y2.l4 -0.024645   0.017651  -1.396    0.164
## y3.l4 -0.035087   0.076780  -0.457    0.648
## const  0.117705   0.076481   1.539    0.125
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.2445 on 192 degrees of freedom
## Multiple R-Squared: 0.9786,  Adjusted R-squared: 0.9773
## F-statistic: 732.7 on 12 and 192 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##          y1       y2       y3
## y1  0.69187  0.13554 -0.07901
## y2  0.13554  1.06001 -0.00749
## y3 -0.07901 -0.00749  0.05977
##
## Correlation matrix of residuals:
##         y1       y2       y3
## y1  1.0000  0.15827 -0.38852
## y2  0.1583  1.00000 -0.02976
## y3 -0.3885 -0.02976  1.00000
```

The results above suggest that $y1$ is caused by lags 1 and 2 of $y3$. $y2$ is caused by lags 1 and 2 of $y3$. Finally, $y3$ is only caused by its own past. This is kind of nice, and kind of atheoretically scary... It is like regressing everyting on everything and picking the best model... Conclusion: don't do this at home! To read more about VAR, see here.

Cointegration and Error Correction Models

It is important to note that cross-correlation and Granger causality models work only for stationary data – series that are not integrated. But what if our x and y are not stationary, what if they are integrated together – that is *co-integrated*? The logic of cointegration is that series x sets a target level to which series y responds. If the series are truly causally related, then a mismatch between the series must be subsequently corrected.

The logic is that co-integrated series move in tandem. Regressing one series on the other thus estimates the movement, and leaves the residuals stationary. Finally, to ascertain causality $x \rightarrow y$, the stationary residuals

get corrected in the future values of y.

There are two ways to estimate co-integration. The first approach of Engler and Granger uses a two-step method, the second approach, called simply the 'Error Correction Model' (ECM) carries out both steps in one equation.

Engel and Granger Two-Step method

1) Here we first estimate the simple regression between the two series to get its residuals z:

$y_t = \beta_0 + \beta_1 x_t + u_t$

2) In the second setp, we estimate the error correction in z

$\Delta y_t = \alpha \Delta x_t - \pi z_{t-1} + \upsilon_t$

Here note that Δ stands for 'change' or 'difference', $\Delta y_t = y_t - y_{t-1}$. It is implemented with the `diff()` function in R. Importantly, a negative coefficient π suggests an error correction process.

Error Correction Model

The Error Correction Model combines the two steps above into one equation:

$\Delta y_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 x_{t-1} + \beta_3 \Delta x_t + e_t$

Note that here y_{t-1} is the so-called *lagged dependent variable* (LDV). It just the past of y. The coefficient β_3 (on Δx) suggests Granger causality whereby change in x causes change in y. A negative coefficient β_1 (on the LDV) suggests error correction.

Let's consider an example of co-integration, using data on interest rates.

```
D<-import("https://jan-rovny.squarespace.com/s/RATES.DTA")
```

Here `D$tbond` is the U.S. tresury long-term interest rate on government debt, and `D$prime` is prime market (commercial banks) rate to lenders. The interest is to see whether the prime lending rate affect the U.S. tresury tbond rate.

Let's start with the Engel and Granger two-step method.

```
#1 Engel and Granger Two-Step Method

#Step 1)
m1<-lm(D$tbonds~D$prime)  #simple regression
res<-m1$residuals #get the residuals (should be stationary, as the series correct for each other)
D<-cbind(D,res)

#check stationarity
adf.test(D$tbonds) #integrated (big p-value)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  D$tbonds
## Dickey-Fuller = -1.3112, Lag order = 8, p-value = 0.8699
## alternative hypothesis: stationary
```

```
adf.test(D$prime) #integrated (big p-value)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  D$prime
## Dickey-Fuller = -2.7289, Lag order = 8, p-value = 0.2698
## alternative hypothesis: stationary
```

```
adf.test(D$res) #stationary!  (small p-value)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  D$res
## Dickey-Fuller = -3.963, Lag order = 8, p-value = 0.01085
## alternative hypothesis: stationary
```

```
#prepare data
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
D<-mutate(D, l.res = lag(res)) #create a lagged variable of res "l.res"
C<-D[-1,] #remove first row from D data (because we have a missing value on the lagged var)
C$d.tbonds<-diff(D$tbonds) #create differenced variable
C$d.prime<-diff(D$prime) #create differenced variable

#Step 2)
m2<-lm(d.tbonds~d.prime+l.res, data=C)
#here we want to see that the stationary errors (res) get corrected in the future values of y
summary(m2)
```

```
##
## Call:
## lm(formula = d.tbonds ~ d.prime + l.res, data = C)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.35077 -0.08889 -0.00272  0.08061  1.40567
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  0.004184   0.009200   0.455   0.6494
## d.prime      0.175137   0.020487   8.549   <2e-16 ***
## l.res       -0.021660   0.008551  -2.533   0.0116 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2234 on 587 degrees of freedom
## Multiple R-squared:  0.1142, Adjusted R-squared:  0.1112
## F-statistic: 37.85 on 2 and 587 DF,  p-value: 3.468e-16
```

In the model summary the coefficient on Δx (d.prime) indicates causality of Granger type: $\Delta x \rightarrow \Delta y$ The negative coefficient on the lagged residuals (l.res) indicates error correction.

Next, let's repeat the analysis with the ECM model.

```
#####
#2. One-Step ECM

#prepare data
l.tbonds<-transmute(D, l.tbonds = lag(tbonds)) #create lag of tbonds
l.tbonds<-l.tbonds[-1,]  #remove first obs of lagged variable (as it is empty)
l.prime<-transmute(D, l.prime = lag(prime)) #create lag of prime
l.prime<-l.prime[-1,] #remove first obs of lagged variable (as it is empty)
C<-cbind(C,l.tbonds,l.prime) #combine with C frame

m3<-lm(d.tbonds~l.tbonds+d.prime+l.prime, data=C) #estimate model
summary(m3)
```

```
##
## Call:
## lm(formula = d.tbonds ~ l.tbonds + d.prime + l.prime, data = C)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.33888 -0.09159 -0.00545  0.07978  1.41389
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.037099   0.022712   1.633   0.1029
## l.tbonds    -0.021631   0.008558  -2.528   0.0117 *
## d.prime      0.174536   0.020560   8.489   <2e-16 ***
## l.prime      0.014403   0.006613   2.178   0.0298 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2236 on 586 degrees of freedom
## Multiple R-squared:  0.1144, Adjusted R-squared:  0.1099
## F-statistic: 25.24 on 3 and 586 DF,  p-value: 2.265e-15
```

In the summary above, note that `l.tbonds` is the lagged dependent variable (LDV). The LDV's negative effect indicates error correction. The significant coefficient on `d.prime` indicates Granger type causality: $\Delta x \rightarrow \Delta y$.

We can interpret the results of this model in the following way. Movements in the prime rate cause movements in long tbond rates in two ways: First, changes in the prime flow through to changes in long (tbond) rates. Second, the prime rates have an equilibrium relationship with long (tbond) rates which is corrected whenever one or the other strays from the target levels.