# CSSS 512: Lab 4

## Cointegration Analysis

2018-4-27

# Cointegration Analysis

So far, we have been examining a single time series conditional upon some covariates (e.g. traffic accidents and seat belt law). We have been assuming that there is no feedback between variables.

Yet, we may be interested in the relationship between two potentially nonstationary time series that influence each other.

Cointegration analysis allows us to examine the short-run and long-run relationships between two nonstationary time series.

Key intuition: there is some combination of the nonstationary time series that yields an error term that is stationary, so that shocks are not permanent and the system holds its equilibrium.

# Cointegration Analysis

Consider two time series $y_t$ and $x_t$:

$$x_t = x_{t-1} + \epsilon_t$$

$$y_t = y_{t-1} + 0.6x_t + \nu_t$$

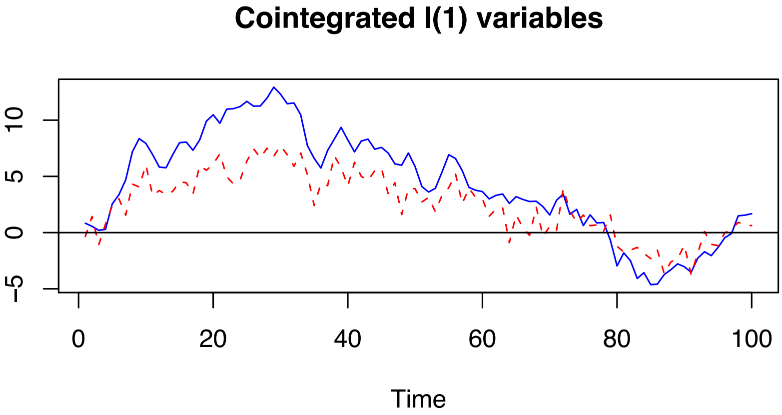
where $\epsilon_t$ and $\nu_t$ are white noise. $x_t$ and $y_t$ are both AR(1) processes, random walks, non-stationary, and I(1).

They are causally connected, and neither tends toward any particular level, but each tends toward the other. A large $\nu_t$ may move $y_t$ away from $x_t$ briefly, but eventually, $y_t$ will move back to $x_t$'s level. Think: drunkard and puppy.

The two time series may be in equilibrium in the long run but in the short run the two series deviate from that equilibrium. They will move together indefinitely. $x_t$ and $y_t$ are said to be cointegrated.

Shocks that persist over a single period are reequilibriated or adjusted by the cointegrating relationship.

# Cointegration Analysis



# Cointegration Analysis

Cointegration allows for us to acknowledge that two nonstationary time series may be related in multiple ways: in the short-run and in the long-run.

Examples of cointegrated time series:

1. crime rates and immigration rates
2. presidential approval and inflation
3. demand for money, interest rates, income, and prices

*Justification*:

Differencing nonstationary time series eliminates the possibility of capturing long run relationships. However, ignoring nonstationarity can uncover spurious relationships.

We want to examine nonstationary time series by allowing for the possibility of long-run relationships while also investigating whether short-run perturbations are related.

# Cointegration Analysis

Cointegration means that a specific combination of two nonstationary series may be stationary. The vector(s) that defines the stationary linear combination is called the cointegrating vector.

Specifically, two or more variables $y_t$, $x_t$ are cointegrated if

1. Each of the variables is I(d), d $\geq$ 1, usually I(1).
2. There is some cointegrating vector, $\boldsymbol{\alpha}$, such that

$$z_t = [y_t, x_t]'\boldsymbol{\alpha}$$

$$z_t \sim I(0)$$

In other words, there is some linear combination of the non-stationary variables that is stationary.

# Cointegration Analysis

**Steps:**

1. Determine whether the individual time series are stationary
2. If the series are non-stationary, then find out if they are cointegrated: if there is a linear combination of the series that is stationary
3. Fit an error correction model: Engle-Granger Two-Step or Johanson Estimator

# Engle-Granger Two-Step

Assume we have ascertained that the individual time series are non-stationary. Now, we can test for cointegration.

**Step 1:**

Estimate the following using linear regression (no constant):

$$y_t = x_t\beta + \epsilon_t$$

The residuals, $\hat{\epsilon_t}$, should be stationary if $x$ and $y$ are cointegrated.

We test to see if $\hat{\epsilon_t}$ is stationary. It takes the form of a unit root test:

$$\Delta\hat{\epsilon_t} = \alpha_1\hat{\epsilon}_{t-1} + z_t$$

We are interested in whether $\alpha_1 = 0$. Rejecting the null means that the residuals series is stationary.

# Engle-Granger Two-Step

**Step 2:**

Estimate the Error Correction Model

$$\Delta y_t = \psi_0 + \gamma_1 \hat{\epsilon}_{t-1} + \sum_{j=1}^{J} \psi_{1j} \Delta x_{t-j} + \sum_{k=1}^{K} \psi_{2k} \Delta y_{t-k} + u_t$$

$$\Delta x_t = \zeta_0 + \gamma_2 \hat{\epsilon}_{t-1} + \sum_{j=1}^{J} \zeta_{1j} \Delta y_{t-j} + \sum_{k=1}^{K} \zeta_{2k} \Delta x_{t-k} + v_t$$

The key terms to note at are the $\hat{\epsilon}_{t-1}$ from our previous step.

This gives us $\Delta y_t$ as a function of its lags, the lags of $\Delta x_t$, and the error of the long-run equilibrium, $\epsilon_{t-1}$.

A negative $\gamma$ shows us how quickly $y_t$ reverses back to $x_t$. Larger negative values of $\gamma$ mean fast adjustment back to equilibrium. It is called the *speed of adjustment parameter*.

# Engle-Granger Two Step

```
rm(list=ls())

#Load libraries
library(tseries)              # For unit root tests
library(forecast)                 # For decompose()
library(lmtest)               # For Breusch-Godfrey LM test of serial correlation
library(urca)                 # For estimating cointegration models
library(simcf)                # For counterfactual simulation via ldvsimev()
library(MASS)                 # For mvrnorm()
library(RColorBrewer)         # For nice colors
library(Zelig)                    # For approval data
library(quantmod)                 # For creating lags

data(approval)
attach(approval)

phony <- rnorm(length(approve))
for (i in 2:length(phony)){
    phony[i] <- phony[i-1] + rnorm(1)
}
```

# Engle-Granger Two Step

```
set.seed(123456)

# Generate cointegrated data
e1 <- rnorm(100)
e2 <- rnorm(100)
x <- cumsum(e1)
y <- 0.6*x + e2

#Run step 1 of the Engle-Granger two step
coint.reg <- lm(y ~ x -1)
#Estimate the cointegration vector by least squares with no constant
coint.err <- residuals(coint.reg)
#This gives us the cotingeration vector

#Check for stationarity of the cointegration vector
punitroot(adf.test(coint.err)$statistic, trend="nc")
```

```
## Dickey-Fuller
##  6.551997e-05
```

```
#Make the lag of the cointegration error term
coint.err.lag <- coint.err[1:(length(coint.err)-2)]
```

# Engle-Granger Two Step

```r
#Make the difference of y and x
dy <- diff(y)
dx <- diff(x)

#And their lags
dy.lag <- dy[1:(length(dy)-1)]
dx.lag <- dx[1:(length(dx)-1)]

#Delete the first dy, because we are missing lags for this obs
dy <- dy[2:length(dy)]
```

# Engle-Granger Two Step

```r
#Estimate an Error Correction Model with LS
ecm1 <- lm(dy ~ coint.err.lag + dy.lag + dx.lag)
summary(ecm1)
```

```
##
## Call:
## lm(formula = dy ~ coint.err.lag + dy.lag + dx.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9553 -0.5375  0.1538  0.7042  2.3240
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.02267    0.10381   0.218    0.828
## coint.err.lag -0.96617    0.15864  -6.090 2.45e-08 ***
## dy.lag        -1.05776    0.10848  -9.751 6.21e-16 ***
## dx.lag         0.81035    0.11223   7.221 1.33e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.026 on 94 degrees of freedom
## Multiple R-squared:  0.5456, Adjusted R-squared:  0.5311
## F-statistic: 37.62 on 3 and 94 DF,  p-value: 4.624e-16
```

# Johansen Estimator

The second approach to cointegration analysis is the Johansen method estimated via ML.

The Johansen method does not rely on an arbitrary choice of specification for the cointegrating vector. It also identifies multiple cointegrating vectors in the case of three or more variables.

The Johansen method gives us test statistics on the possible number of cointegrating vectors.

As with Engle-Granger, lag lengths can be determined by information criteria or the shortest lag length that results in serially uncorrelated residuals.

# Johansen Estimator

```
#Alternatively, we can use the Johansen estimator
#Create a matrix of the cointegrated variables
cointvars <- cbind(y,x)
# Perform cointegration tests
coint.test1 <- ca.jo(cointvars,
                     ecdet = "const",
                     type="eigen",
                     K=2,
                     spec="longrun")
```

# Johansen Estimator

```
summary(coint.test1)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in cointegrat
##
## Eigenvalues (lambda):
## [1] 3.105216e-01 2.077094e-02 3.335727e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 |  2.06  7.52  9.24 12.97
## r = 0  | 36.44 13.75 15.67 20.20
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                 y.l2      x.l2   constant
## y.l2      1.00000000   1.00000   1.000000
## x.l2     -0.58297186  10.12695  -1.215134
## constant -0.02960597 -50.23990 -38.501184
##
## Weights W:
## (This is the loading matrix)
##
##             y.l2         x.l2     constant
## y.d -0.967714950 -0.001015446 1.784095e-17
## x.d  0.002461222 -0.002817005 1.142521e-18
```

# Johansen Estimator

```
ecm.res1 <- cajorls(coint.test1,
                    r = 1,           # Cointegration rank
                    reg.number = 1)  # which variable(s) to put on LHS
#(column indexes of cointvars)
summary(ecm.res1)
```

```
##      Length Class  Mode
## rlm  12     lm     list
## beta  3     -none- numeric
```

# Johansen Estimator

```
#For the presidential approval example, use an ECM equivalent to the
#ARIMA(1,0,1) model that we chose earlier

cointvars <- cbind(approve,avg.price)
ecm.test1 <- ca.jo(cointvars,
                   ecdet = "const",
                   type="eigen",
                   K=2,
                   spec="longrun",
                   dumvar=cbind(sept.oct.2001,iraq.war)
                   )
```

# Johansen Estimator

```
summary(ecm.test1)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in cointegrat
##
## Eigenvalues (lambda):
## [1] 2.391542e-01 1.367744e-01 1.387779e-16
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 |  9.27  7.52  9.24 12.97
## r = 0  | 17.22 13.75 15.67 20.20
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##               approve.l2 avg.price.l2 constant
## approve.l2     1.0000000    1.0000000  1.00000
## avg.price.l2   0.1535049    0.3382829 -1.05616
## constant     -76.0019182 -120.7778161 90.24200
##
## Weights W:
## (This is the loading matrix)
##
##              approve.l2 avg.price.l2     constant
## approve.d   -0.12619985   0.02231967 5.407866e-17
## avg.price.d -0.02220281  -0.58771490 3.727850e-16
```

# Johansen Estimator

```
ecm.res1 <- cajorls(ecm.test1,
                    r = 1,
                    reg.number = 1)
summary(ecm.res1$rlm)
```

```
##
## Call:
## lm(formula = substitute(form1), data = data.mat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.1403 -1.6753 -0.2261  1.6430  5.9537
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## ect1          -0.12620    0.03006  -4.198 9.37e-05 ***
## sept.oct.2001 19.55846    2.11737   9.237 5.40e-13 ***
## iraq.war       5.01870    1.62432   3.090  0.00307 **
## approve.dl1   -0.31757    0.09448  -3.361  0.00138 **
## avg.price.dl1 -0.05055    0.02593  -1.949  0.05613 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.668 on 58 degrees of freedom
## Multiple R-squared:  0.6301, Adjusted R-squared:  0.5983
## F-statistic: 19.76 on 5 and 58 DF,  p-value: 1.915e-11
```

# Johansen Estimator

```
ecm.res1 <- cajorls(ecm.test1,
                    r = 1,
                    reg.number = 1)
summary(ecm.res1$rlm)
```

```
##
## Call:
## lm(formula = substitute(form1), data = data.mat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.1403 -1.6753 -0.2261  1.6430  5.9537
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## ect1          -0.12620    0.03006  -4.198 9.37e-05 ***
## sept.oct.2001 19.55846    2.11737   9.237 5.40e-13 ***
## iraq.war       5.01870    1.62432   3.090  0.00307 **
## approve.dl1   -0.31757    0.09448  -3.361  0.00138 **
## avg.price.dl1 -0.05055    0.02593  -1.949  0.05613 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.668 on 58 degrees of freedom
## Multiple R-squared:  0.6301, Adjusted R-squared:  0.5983
## F-statistic: 19.76 on 5 and 58 DF,  p-value: 1.915e-11
```

# Johansen Estimator

```r
#Cointegration analysis with a spurious regressor
cointvars <- cbind(approve,avg.price,phony)

ecm.test1 <- ca.jo(cointvars,
                   ecdet = "const",
                   type="eigen",
                   K=2,
                   spec="longrun",
                   dumvar=cbind(sept.oct.2001,iraq.war)
                   )
```

# Johansen Estimator

```
summary(ecm.test1)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in cointegrat
##
## Eigenvalues (lambda):
## [1] 3.941333e-01 1.787459e-01 1.137340e-02 1.304403e-16
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 2 |  0.72  7.52  9.24 12.97
## r <= 1 | 12.41 13.75 15.67 20.20
## r = 0  | 31.57 19.77 22.00 26.81
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                 approve.l2 avg.price.l2     phony.l2    constant
## approve.l2     1.00000000    1.0000000   1.00000000   1.0000000
## avg.price.l2  -0.06406128    0.5240426   0.09639494  -0.2174818
## phony.l2      -1.65161575    1.6333383  -2.23457701  -1.3289803
## constant     -50.29338662 -137.0074869 -95.74169292 -42.3748820
##
## Weights W:
## (This is the loading matrix)
##
##               approve.l2 avg.price.l2     phony.l2      constant
## approve.d   -0.14914360 -0.016564384  0.014448982  2.258302e-16
## avg.price.d  0.14745059 -0.478284212 -0.024008295 -1.481206e-15
```

# Johansen Estimator

```
ecm.res1 <- cajorls(ecm.test1,
                    r = 1,
                    reg.number = 1)
summary(ecm.res1$rlm)
```

```
##
## Call:
## lm(formula = substitute(form1), data = data.mat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5089 -1.6557  0.1365  1.2910  6.8158
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## ect1          -0.14914    0.02918  -5.111 3.89e-06 ***
## sept.oct.2001 18.41634    1.99527   9.230 6.52e-13 ***
## iraq.war       5.46937    1.55621   3.515 0.000871 ***
## approve.dl1   -0.36276    0.09159  -3.961 0.000210 ***
## avg.price.dl1 -0.02808    0.02467  -1.138 0.259779
## phony.dl1     -0.75693    0.35835  -2.112 0.039059 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.542 on 57 degrees of freedom
## Multiple R-squared:  0.6702, Adjusted R-squared:  0.6355
## F-statistic:  19.3 on 6 and 57 DF,  p-value: 3.893e-12
```