Unit Roots, Cointegration, and Error-Correction Models

Finance 6470: Derivatives Markets
Tyler J Brough
Last Update: February 20, 2020

Unit Roots and Stationarity

A simple starting model for efficient log-prices of assets is the ***Random Walk with Drift*** model:

$$y_t = \mu + y_{t-1} + \epsilon_t, \qquad \epsilon \sim N(0, \sigma_\epsilon^2)$$

with something like $y_t = \ln(p_t)$ with p_t a transaction price observed in some market. The expected value of this process is:

$$E(y_t) = \mu + y_{t-1}$$

To get the variance it is helpful to solve recursively as follows, assuming $y_0 = 0$ for simplicity:

$$y_t = t\mu + \sum_{i=0}^{t} \epsilon_{t-i}$$

We can now state the variance of the process as:

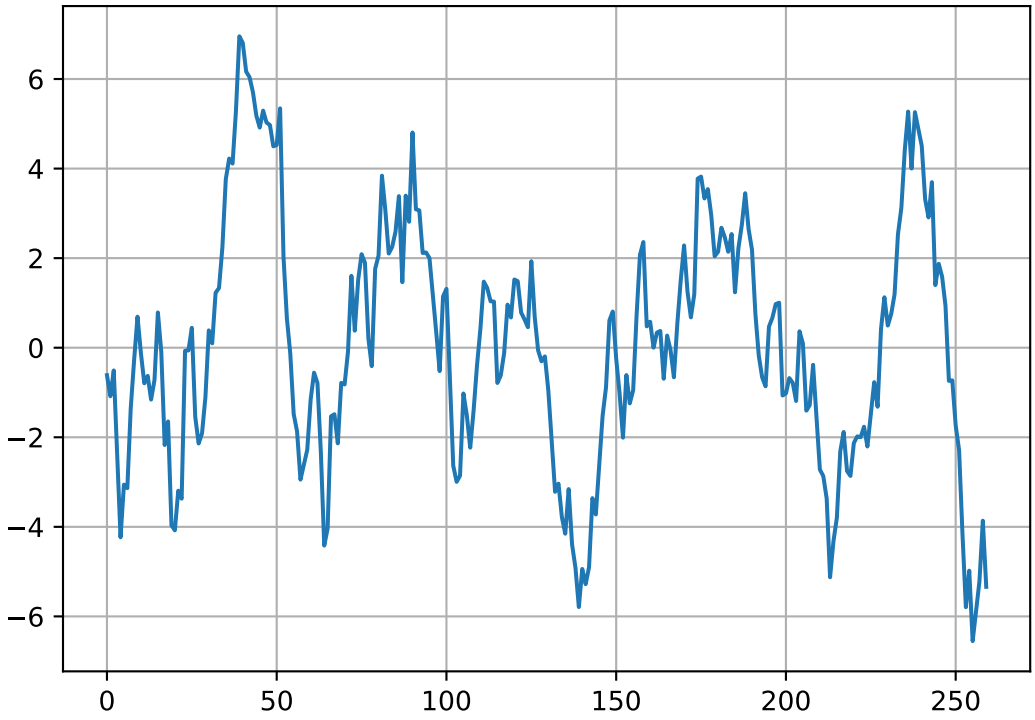
$$Var(y_t) = \sum_{i=0}^{t} Var(\epsilon_{t-i}) = Var(\epsilon_t) + Var(\epsilon_{t-1}) + \cdots + Var(\epsilon_0) = t\sigma_\epsilon^2$$

From this it is easy to see that this is a diffusive process as the variance is proportional to time. This translates to the value of the process at any time t being unpredictable based on the information known up to that time (I_{t-1}). This is a good starting place for a model of informationally efficient prices as the definition of such is one that incorporates all available information up to that point in time.

NB: Samuelson's paper: Proof That Properly Anticipated Prices Fluctuate Randomly

We can simulate this process as follows (setting $\mu = 0$ for convenience):

```
y = pd.Series(np.cumsum(np.random.normal(size=52*5)))
plt.plot(y)
plt.grid(True)
plt.show()
```



Weak Stationarity

In the time series literature a process is known as weakly stationary if the mean and autocovariance are not time varying.

A few notes:

- Clearly the random walk process is **NOT** weakly stationary.
- A weakly stationary time series process will exhibit mean reversion
- Mean reversion in a time series can be such that there is some predictibility in the process

It makes sense that informationally efficient prices should behave as a random walk model (with possible other extensions).

The random walk model is a special case of the AR(1) model:

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

It won't be shown here, but a technical requirement for the AR(1) model to be weakly stationary is $|\phi| < 1$. For the random walk model $\phi = 1$, thus the alternative name ***unit root***.

Some Notation

A random walk model is also known as a unit-root non-stationary process. In the literature this is often denoted as $y_t \sim I(1)$. We state this as: "*the process* y_t *is* ***integrated of order one***."

We can tranform and $I(1)$ process to a stationary process by ***first differencing*** the process like so:

$$\begin{aligned} y_t - y_{t-1} &= y_{t-1} - y_{t-1} + \epsilon_t \\ y_t - y_{t-1} &= \epsilon_t \\ \Delta y_t &= \epsilon_t \end{aligned}$$

In this case we can denote that Δy_t is now weakly stationary with the notation $\Delta y_t \sim I(0)$, i.e. "Δy_t *is* ***integrated of order zero.***"

Spurious Regression

It is important to understand the properties of unit root processes, because they can be problematic to work with in applying econometrics to finance.

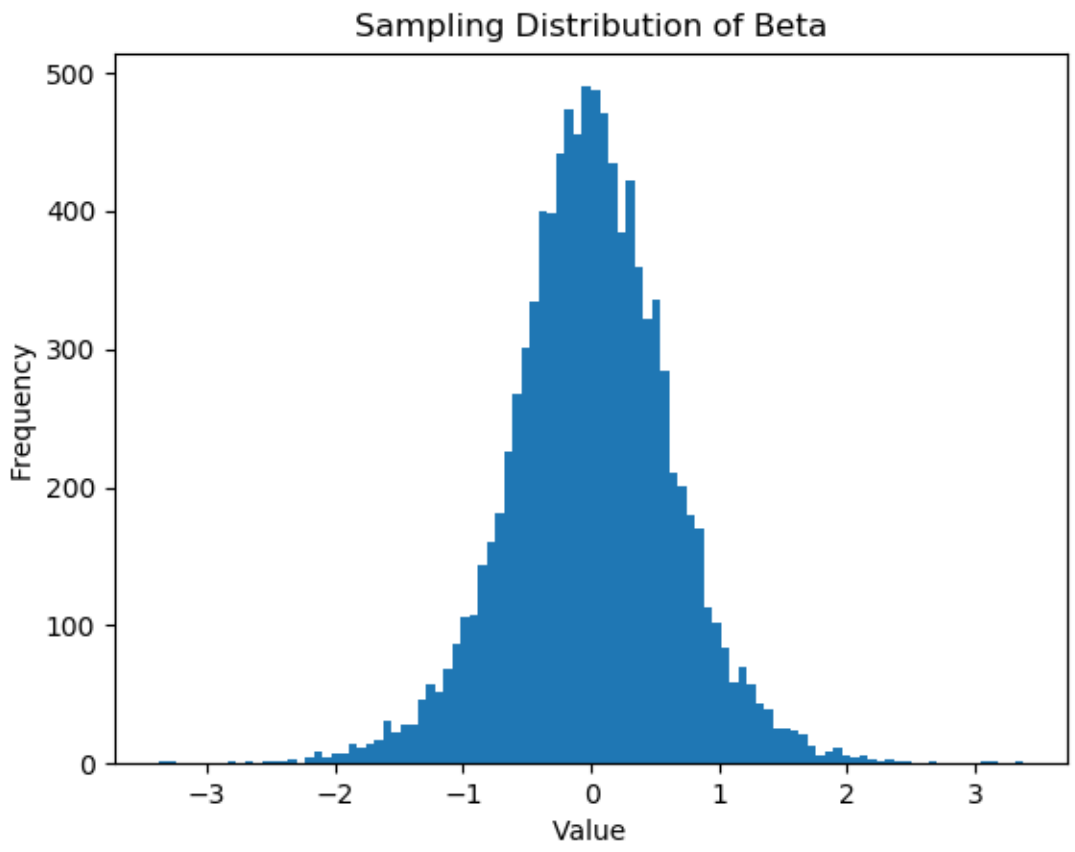
For example, there is a well-known problem of ***spurious regression*** when one unit root process is regressed on an independent unit root process:

$$y_t = \alpha + \beta x_t + u_t, \quad u_t \sim N(0, \sigma_u^2)$$

This regression is not valid because the homoscedasticity assumption of the error term is violated (recall that $Var(y_t) = t\sigma_\epsilon^2$)

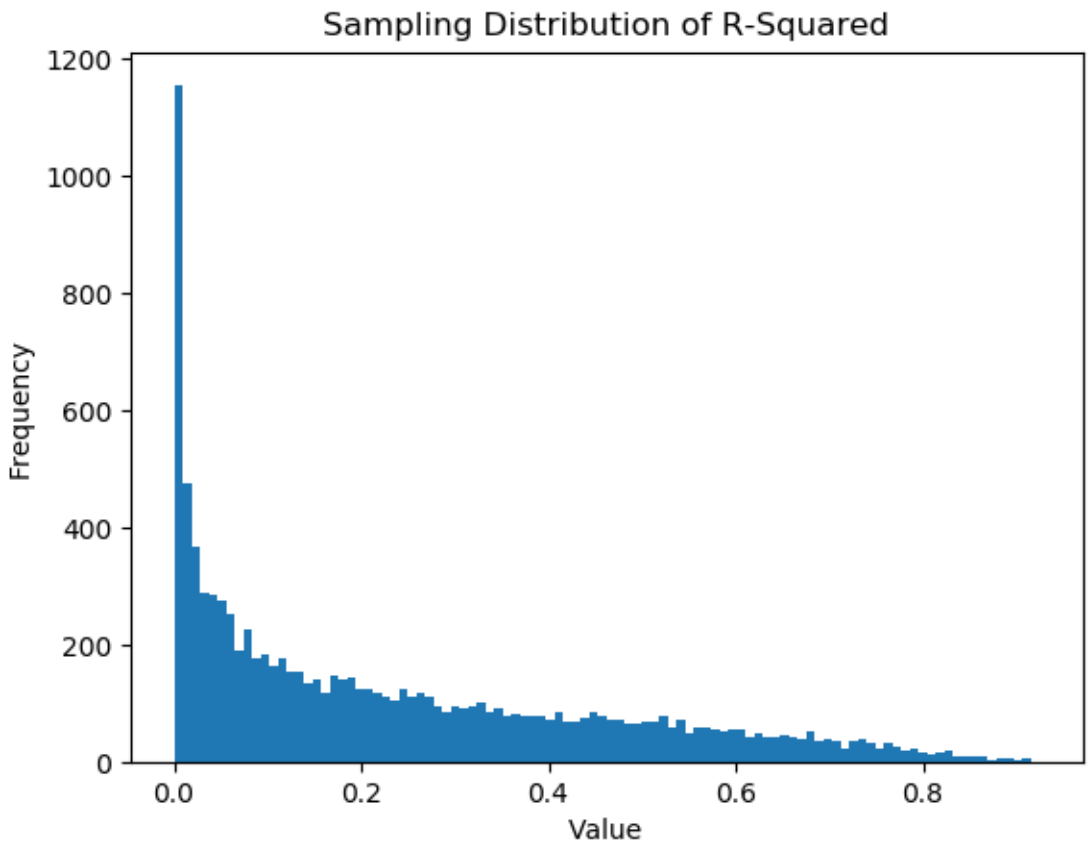
This can be easily demonstrated by a simple Monte Carlo study as follows:

```
M = 10000
N = 52 * 5
betaHat = np.empty(M)
Rsqrd = np.empty(M)

for i in range(M):
    y = np.cumsum(np.random.normal(size=N))
    x = np.cumsum(np.random.normal(size=N))
    reg = stats.linregress(x,y)
    betaHat[i] = reg.slope
    Rsqrd[i] = reg.rvalue ** 2

plt.hist(betaHat, bins=100);
plt.show()
```



While the central tendency of sampling distribution appears to be zero, the distribution is extremely wide, so while we will fail to reject the null hypothesis on average we will fail to do so far too often.

```
plt.hist(Rsqrd, bins=100);
plt.show()
```



We can also see from histogram of the R^2 that there are some extremely high values even though we know that the processes are independent!

The Dickey-Fuller Test for Unit Roots

We can test for unit roots in a time series process with the so-called ***Dickey-Fuller Test***, named for the statisticians who invented it.

It would seem natural to test the following hypothesis:

$$H_0 : \phi = 1$$
$$H_a : \phi \neq 1$$

for the regression: $y_t = \phi y_{t-1} + \epsilon_t$, but because the model under the null hypothesis leads to spurious regression we cannot conduct this direct test.

D-F had the bright idea to transform the model to render it amenable to such testing. We start by subtracting y_{t-1} from both sides:

$$\begin{aligned} y_t - y_{t-1} &= \phi y_{t-1} - y_{t-1} + \epsilon_t \\ \Delta y_t &= (\phi - 1) y_{t-1} + \epsilon_t \\ \Delta y_t &= \theta y_{t-1} + \epsilon_t \end{aligned}$$

where $\theta = \phi - 1$. With this transformation we can now conduct the test:

$$\begin{aligned} H_0 &: \theta = 0 \\ H_a &: \theta \neq 0 \end{aligned}$$

Because $\Delta y_t \sim I(0)$ and when $\phi = 1$ it means that $\theta = 0$ this model is now valid under the null hypothesis. We can form the standard t-ratio as our test statistic, but D-F showed that the asymptotic sampling distribution of t is no longer the Standard Normal distribution. Instead they provide critical values via a Monte Carlo method.

The Augmented Dickey-Fuller Test

D-F added one extension to the test to account for possible serial correlation in Δy_t. The model under the null hypothesis now becomes:

$$\Delta y_t = \theta y_{t-1} + \sum_{i=1}^{p} \gamma_i \Delta y_{t-i} + \epsilon_t$$

This tends to make the test more robust to short-term serial correlations in the process.

We can use the Statsmodels model to conduct the ADF test as follows. See Statsmodels documentation on adfuller for more details.

```
import statsmodels.api as sm

N = 52 * 5
y = np.cumsum(np.random.normal(size=N))
results = sm.tsa.stattools.adfuller(y)
msg = f"The value of the ADF statistic is {results[0]:0.4f}, "
msg += f"with a p-value of: {results[1]: 0.4f}"
print(msg)
results
```

Here we can see that we fail to reject the null hypothesis of the presence of a unit-root in y_t, which is not surprising since we simulated it as a random walk.

Cointegration

Fortunately, there is an upside to unit-root non-stationarity for financial modeling. It turns out that there is a relationship that is even stronger when pairs of asset prices are $I(1)$, but move together in a way. This concept is ***cointegration***.

If there is a linear combination of two processes that are separately $I(1)$ that is itself $I(0)$, we say that the two processes are ***cointegrated***. Cointegration is a stronger concept than mere correlation. It has a causal explanation. I often like to say that (at least in financial applications) *cointegration is the statistical footprint of an arbitrage relationship.*

We can test for cointegration using the ADF test developed above. To begin, set up and run the following regression:

$$y_t = \alpha + \beta x_t + u_t$$

If the variables are cointegrated this is not a spurious regression. In fact, it has the property of superconsistency (a kind of uber statistical efficiency).

Once the model is estimated, we can form the fitted residuals:

$$\hat{u_t} = y_t - \hat{\alpha} - \hat{\beta} x_t$$
$$\hat{u_t} = y_t - \hat{y_t}$$

We can now submit these fitted residuals to the ADF test as above.

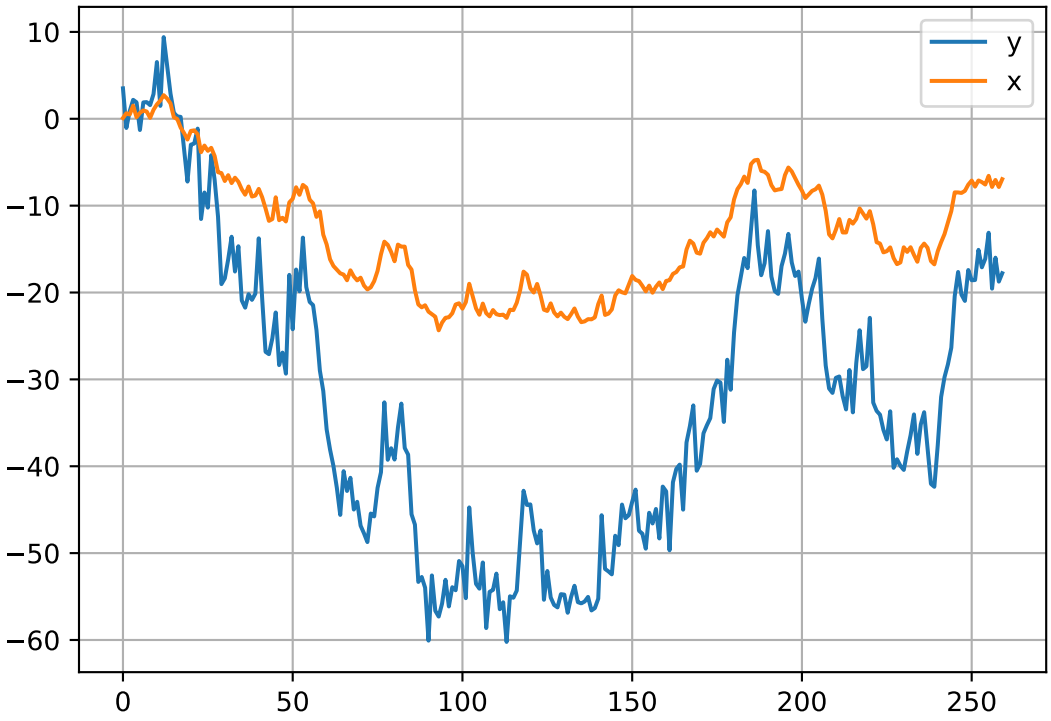
NB: the null hypothesis of the ADF test is that there ***is*** a unit-root. Cointegration exists between y_t and x_t if there ***is not*** a unit-root in $\hat{u_t}$. So we conclude that there is cointegration if reject the null hypothesis of the ADF test.

We can simulate this as follows.

```
N = 52 * 5
x = np.cumsum(np.random.normal(size=N))
u = np.random.normal(size=N, loc=0.0, scale=2.0)
y = 0.22 + 2.45 * x + u

df = pd.DataFrame(dict(y=pd.Series(y), x=pd.Series(x)))
df.head(10)
```

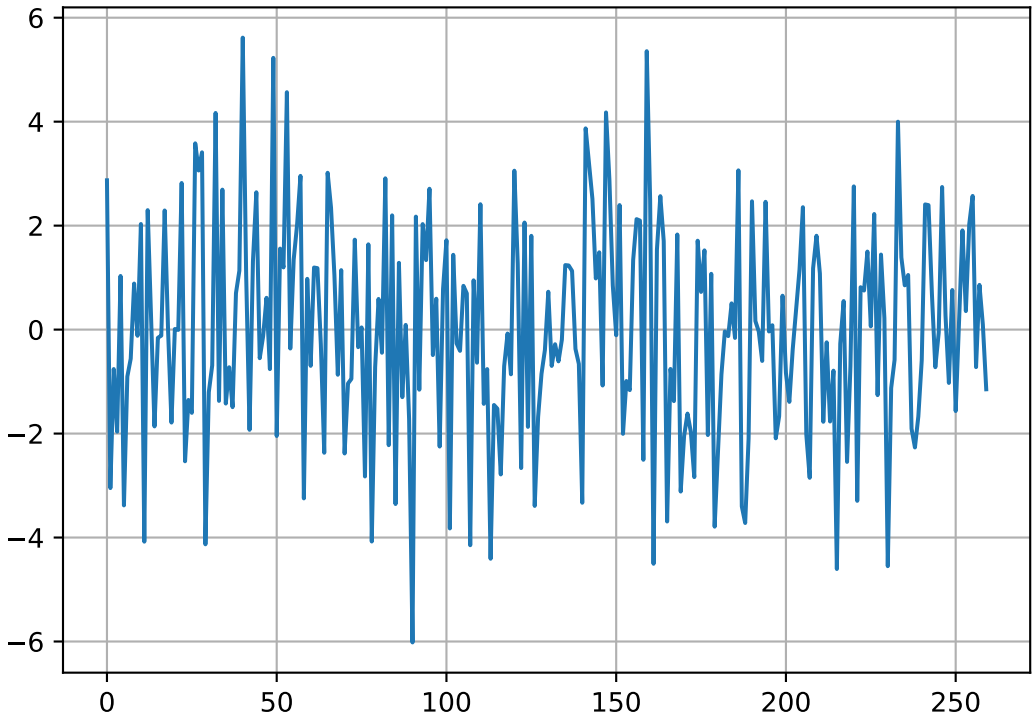
```
##           y         x
## 0  3.504627  0.083630
## 1 -1.059084  0.636684
## 2  0.884541  0.495363
## 3  2.182111  1.515442
## 4  1.895194  0.177574
## 5 -1.294561  0.676265
## 6  1.885821  0.962431
## 7  1.922757  0.837098
## 8  1.571496  0.105164
## 9  2.847457  1.034663
```

```
df.plot(grid=True)
```



Now let's run the regression and form the residuals:

```
reg = stats.linregress(x,y)
uhat = y - reg.intercept - reg.slope * x
resids = pd.Series(uhat)
resids.plot(grid=True)
```



The residuals clearly look to be mean-reverting and stationary. Let's see the results of the ADF test:

```
import statsmodels.api as sm
results = sm.tsa.stattools.adfuller(uhat)
results
msg = f"The ADF test statistic is: {results[0]: 0.4f}, "
msg += f"with a p-value of: {results[1] : 0.4f}"
print(msg)
```

That's an awefully small p-value, so we reject the null hypothesis of the ADF test of a unit-root and conclude that y_t and x_t are cointegrated. Again, we're not surprised since we engineered it.

Error-Correction Models

Whenever two (or more) asset prices are cointegrated, we can also write down an error-correction model. That is, cointegration implies an error-correction form.

We state the simplest form of the error-correction model as follows:

$$\Delta y_t = \lambda(y_{t-1} - \alpha - \beta x_{t-1}) + \nu_t$$
$$\Delta y_t = \lambda(z_{t-1}) + \nu_t$$

with $z_t = \hat{u}_t$.

This model form relates the changes in y_t, that is Δy_t to the ***spread*** between y_{t-1} and x_{t-1} in ***levels***. This is a valid time series regression, given that $\Delta y_t \sim I(0)$ via first differencing, and $z_{t-1} \sim I(0)$ via cointegration.

Let's see if we can develop some intuition for this model. Let's start by interpreting the coefficient λ, which we call the ***error-correction coefficient***. Its value will be such that when there is a large past deviation between y_{t-1} and x_{t-1} (i.e. a large error) it will cause an ***error correction*** in the change in y_t, or Δy_t. In other words, Δy_t will adjust based on a lagged error in the spread. There is now a stationary relationship (i.e. mean-reverting) that exists, and can even be predicted. It's easy to see now why we call this an error-correction model, and also its relationship with cointegration.

- **Q:** What causes the error-correction in Δy_t?
- **A:** in financial markets between related asset prices that are cointegrated, the answer is ***arbitrage***!

We can now think about the error-correction model in terms of some kind of equilibrium concept. When dynamic market forces are such that related asset prices are temporarily driven apart, an arbitrage relationship between the asset prices acts to restore the spread between the two to a long-run equilibrium level.

- **Q:** what kind of equilibrium concept fits this description?
- **Q:** is it a static neo-classical equilibrium?
- **Q:** is it more like the neo-Austrian type of equilibrium that has been mentioned in this class?

A More General Error-Correction Model

We can also account for possible short-run variation in Δy_t by adding lagged terms on the right-hand side of the model as follows (as well as a drift term):

$$\Delta y_t = \mu + \sum_{j=1}^{p} \delta_j \Delta y_{t-j} + \lambda z_{t-1} + \nu_t$$

A Vector Error-Correction Model

Now we can think in terms of systems of equations, and think about a multivariate relationship between y_t and x_t called a ***vector error-correction model*** (vecm).

Here is a VECM(1) model in y_t and x_t:

$$\Delta y_t = \mu + \delta_1 \Delta y_{t-1} + \gamma_1 \Delta x_{t-1} + \lambda_1 z_{t-1} + \nu_{1,t}$$
$$\Delta x_t = \mu + \delta_2 \Delta y_{t-1} + \gamma_2 \Delta x_{t-1} + \lambda_2 z_{t-1} + \nu_{2,t}$$