

**WORLDQUANT**  
**UNIVERSITY**<sup>TM</sup>

# Econometrics

## Notes

**WORLDQUANT**  
**UNIVERSITY**<sup>TM</sup>

## Table of Contents

A Word About Python.....	8
Types of Data.....	8
Sources of Financial Data - Economic Data .....	10
Financial Data from Yahoo!.....	15
Problems with Data .....	16
Review of Rules of Stats.....	17
Rules for Expected Value .....	21
Types of Distributions .....	22
Plotting Data to Look for Patterns to Suggest Functional Form of Regression ...	24
Interpretation of Variables in Equations .....	27
Finance Models.....	30
What Is Econometrics and Financial Econometrics? .....	34
Finance Topics Amenable to Econometrics .....	35
Yield Curve (Term Structure of Interest Rates) .....	41
Early Warnings Systems .....	42
Applied Hypothesis in Finance .....	44
Predicting House Prices with Regression.....	50
Predicting House Prices with Support Vector Machines for Regression .....	54
Two-Variable Regression .....	58
Things to Bear in Mind .....	59
Exchange Rates and OLS .....	70
Stationarity and Non-stationarity .....	78
Trends.....	79
Spurious Regression.....	81
Mean Reversion.....	82

<b>Types of Non-stationarity .....</b>	<b>83</b>
<b>Augmented Dickey-Fuller Test .....</b>	<b>85</b>
<b>Critical Values for the Dickey-Fuller Unit Root t-Test Statistics.....</b>	<b>86</b>
<b>Augmented Dickey-Fuller Unit Root Test .....</b>	<b>88</b>
<b>Cooking Raw Data.....</b>	<b>97</b>
<b>Autocorrelation .....</b>	<b>104</b>
<b>Durbin Watson Test .....</b>	<b>104</b>
<b>Heteroscedasticity .....</b>	<b>110</b>
<b>Multicollinearity .....</b>	<b>136</b>
<b>Detecting Multicollinearity .....</b>	<b>138</b>
<b>Detecting MC.....</b>	<b>139</b>
<b>Condition Number (Reported by Python).....</b>	<b>141</b>
<b>Structural Breaks .....</b>	<b>142</b>
<b>Outliers.....</b>	<b>149</b>
<b>Outliers and Quantitative Finance .....</b>	<b>149</b>
<b>Cook's Distance.....</b>	<b>150</b>
<b>Treatment of Outliers .....</b>	<b>151</b>
<b>Dummy Variable Technique .....</b>	<b>153</b>
<b>Residuals and Predicted Values.....</b>	<b>159</b>
<b>Python Code: .....</b>	<b>160</b>
<b>The AR(1) Process .....</b>	<b>161</b>
<b>Forecasting an AR(1) .....</b>	<b>166</b>
<b>Thiel's U.....</b>	<b>168</b>
<b>OLS Regression Results .....</b>	<b>172</b>
<b>MA Process .....</b>	<b>173</b>
<b>ACF and PACF Graphs for a MA(1) Process .....</b>	<b>174</b>

<b>Time Series Analysis.....</b>	<b>175</b>
<b>AR(1) Process.....</b>	<b>176</b>
<b>ACF and PACF in AR(1) .....</b>	<b>176</b>
<b>Moving Average MA(1) Process .....</b>	<b>178</b>
<b>Stationarity.....</b>	<b>181</b>
<b>Autocorrelations and Partial Autocorrelations.....</b>	<b>184</b>
<b>Identifying an AR process.....</b>	<b>185</b>
<b>Introduction.....</b>	<b>192</b>
<b>Box-Jenkins Method .....</b>	<b>192</b>
<b>Steps in Box-Jenkins methodology.....</b>	<b>192</b>
<b>Description .....</b>	<b>192</b>
<b>Theoretical Patterns of ACF and PACF .....</b>	<b>193</b>
<b>Type of model.....</b>	<b>193</b>
<b>Typical pattern of ACF.....</b>	<b>193</b>
<b>Typical pattern of PACF.....</b>	<b>193</b>
<b>ACF and PACF Examples .....</b>	<b>194</b>
<b>Modeling House Prices in The Largest Cities of Australia.....</b>	<b>200</b>
<b>Wold's Decomposition.....</b>	<b>204</b>
<b>VAR.....</b>	<b>206</b>
<b>Model Fitting .....</b>	<b>208</b>
<b>Lag Order Selection.....</b>	<b>211</b>
<b>Information Criteria-Based Order Selection .....</b>	<b>211</b>
<b>Forecasting .....</b>	<b>213</b>
<b>Forecast Error Variance Decomposition (FEVD).....</b>	<b>217</b>
<b>Granger Causality.....</b>	<b>220</b>
<b>Conclusion .....</b>	<b>220</b>
<b>Vector Error Correction Models .....</b>	<b>221</b>

<b>General Case.....</b>	<b>224</b>
<b>Food for Thought .....</b>	<b>226</b>
<b>Filtering and Smoothing.....</b>	<b>227</b>
<b>Kalman Filter .....</b>	<b>229</b>
<b>ARCH(p) .....</b>	<b>233</b>
<b>Simulating an ARCH(1) Process .....</b>	<b>233</b>
<b>Dynamic Correlation Models.....</b>	<b>235</b>
<b>Definitions of Correlation.....</b>	<b>238</b>
<b>Relationship Between Correlation and Volatility.....</b>	<b>245</b>
<b>GARCH Model .....</b>	<b>248</b>
<b>I-GARCH.....</b>	<b>250</b>
<b>A-GARCH: Asymmetric GARCH.....</b>	<b>250</b>
<b>E-GARCH: Exponential GARCH.....</b>	<b>251</b>
<b>N-GARCH: Non-linear GARCH.....</b>	<b>251</b>
<b>High-Frequency GARCH Models .....</b>	<b>252</b>
<b>Multivariate GARCH.....</b>	<b>253</b>
<b>Multivariate GARCH Prediction.....</b>	<b>254</b>
<b>EWMA and Quantitative Finance .....</b>	<b>261</b>
<b>Trading Rules and Concluding Thoughts.....</b>	<b>263</b>
<b>Parametric (Analytical) Value-at-Risk.....</b>	<b>265</b>
<b>Relationship of Correlation and VaR .....</b>	<b>267</b>
<b>Credit Value Adjustments .....</b>	<b>268</b>
<b>The Linear Probability Model .....</b>	<b>276</b>
<b>Logit.....</b>	<b>280</b>
<b>Logistic Regression.....</b>	<b>281</b>
<b>Probit Model.....</b>	<b>283</b>

Probit Example .....	284
Sample Selection Bias, Tobit, Censored Regression .....	286
Tobit or Censored Regression .....	287
Structure.....	289
Estimation .....	290
Notes .....	293
Principal Components Analysis .....	295
Methods .....	302
fit(X[, y]) .....	302
Fit the model with X.....	302
Scree Plot in Python.....	303
To Get Eigenvectors (Evec Below) and Eigenvalues (Eval) from Randomly Generated Data .....	304
Weighted Least Squares.....	305
WLS Regression Results .....	308
WLS Regression Results .....	309
Kernel Density Estimation (KDE) .....	310
Kernel Density Estimation with SciPy .....	311
Nonparametric Tests in Python.....	313
Nonparametric Regression.....	314
Instrumental Variables and Two-Stage Least Squares Estimation.....	315
Ridge Regression.....	324
Theory of MLE.....	329
Binomial MLE.....	329
Introduction.....	338
Nonlinear Model Regression .....	344
Methods Used for Fitting the Data.....	344

<b>Non-linear Growth Models .....</b>	<b>344</b>
<b>Introduction to HJM Framework .....</b>	<b>354</b>
<b>Foreign Exchange.....</b>	<b>358</b>
<b>Determining the Equilibrium Exchange Rate .....</b>	<b>358</b>

# Types of Data

## A Word About Python

Before you can do all the things we do in the course make sure the following are downloaded and installed on your machine.

1. Python
2. Anaconda
3. Spyder
4. NumPy
5. Pandas
6. SciPy
7. Pysal

If you miss one you can install it later just make sure it eventually gets done! Your life will be much better.

## Types of Data

**Time series data:** a variable takes on different values depending on the time that it is generated. For example, closing share prices occur on all days a bourse is open. Monthly data is available from the US Department of Labor indicating the rate of unemployment at a point in time.

**Cross section data:** are collected over many different people, companies etc at a given point in time. The 2010 Census represents many individuals with corresponding characteristics at a point in time.

**Panel Data (pooled cross section time series data):** Represents observation of cross sections of the same individuals/companies at different points in time.

Example of panel data for various companies:

Company	Year	Earnings(mils)	Credit rating	CEO
A	2011	5.4	Aaa	Smith
A	2012	7.5	Aaa	Smith
A	2013	3.2	Aaa	Jones
B	2011	12.6	Ba2	Williams
B	2012	12.1	Ba2	Williams

Examples of government-collected panel data include: The US Current Population Survey (CPS), the German Social Economic Panel, the Swedish study of household market and nonmarket activities and the Intomart Dutch panel of households.

## Sources of Financial Data - Economic Data

### World

[OECD.StatExtracts](#) includes data and metadata for OECD countries and selected nonmember economies.

### United Kingdom

<http://www.statistics.gov.uk/>

### United States

[Federal Reserve Economic Data - FRED](#) (includes URL-based API)

<http://www.census.gov/> <http://www.bls.gov/>

<http://www.ssa.gov/> <http://www.treasury.gov/>

<http://www.sec.gov/> <http://www.economagic.com/>

<http://www.forecasts.org/>

## Foreign Exchange

OANDA Historical Exchange Rates

Dukascopy - Historical FX prices; XML and CSV

ForexForums Historical Data - Historical FX downloads via Amazon S3

GAIN Capital - Historical FX rates (in ZIP format)

## Equity and Equity Indices

<http://finance.yahoo.com/>

<http://www.iasg.com/managed-futures/market-quotes>

[http://kumo.swcp.com/stocks/ Kenneth French Data Library](http://kumo.swcp.com/stocks/)

<http://unicorn.us.com/advdec/>

## Fixed Income

LIBOR rates

FRB: H.15 Selected Interest Rates

Barclays Capital Live (institutional clients only)

## Options and Implied Volatility

<http://www.ivolatility.com/>

<http://www.optionmetrics.com/> <http://www.livevol.com/>

<http://www.historicaloptiondata.com/>

## Futures

<http://www.simiansavants.com/cmedata.shtml>

<http://www.cmegroup.com/market-data/index.html>

<http://www.tradingblox.com/tradingblox/free-historical-data.htm>

<http://www.quandl.com>

## Commodities

<https://globalderivatives.nyx.com/nl/commodities/nyse-liffe>

[LIFFE Commodity Derivatives](#) - 15 min delay; free registration

## Multiple Asset Classes and Miscellaneous

OHIO STATE UNIVERSITY DEPARTMENT OF FINANCE: The Financial Data Finder

Google Finance APIs and Tools

QuantNet Master List RBS Databank

<http://www.eoddata.com/>

Robert Shiller Online Data

## Specific Exchanges

Spanish Futures & Options (MEFF)

## Retrieving Data (into Python) from the Internet:

Getting data from the Federal Reserve Economic Data (FRED) base. The following gets the unemployment rate (UNRATE) from FRED for the time period Jan 1, 2006 to Dec 31 2012. The data is called data and will have 84 monthly observations:

```
>>> import pandas  
>>> import pandas.io.data as web  
>>> import datetime as dt  
>>> start, end = dt.datetime(2006, 1, 1), dt.datetime(2012,  
12,  
31)  
>>> data = web.DataReader('UNRATE', 'fred', start, end)
```

**Note:** for a partial list of FRED US economic data bases see

[http://en.wikipedia.org/wiki/Federal Reserve Economic Data](http://en.wikipedia.org/wiki/Federal_Reserve_Economic_Data)

## Financial Data from Yahoo!

To get financial data for a company such as GE (use the ticker symbol) input the same commands as above but put in the following to access Yahoo:

```
| >>> data = web.DataReader('GE', 'yahoo', start, end)|
```

Try wsj for Wall St. J. and google for practice

Discussion Question:

I have mentioned several data sources above. Find a source of financial data from the internet that you find will export to Excel or Pandas in Python. Post your findings on the discussion board. Please chime in to indicate whether you think someone has found a great source. Remember this is a globalized environment. Bonus points for sources in developing countries.

## Problems with Data

1. Data sometime contain errors for many reasons. Something as simple as transposing 19 as 91 can be a real problem.
2. Selectivity bias: a survey may contain only data completed by willing participants. Those not interested in completing the survey are excluded from analysis thereby all participants are not equally represented potentially skewing the data in a particular direction.
3. May not measure what you want. Example: income of a company may be reported for global operations and you want to study earnings from within the US. This occurs a lot with highly aggregated macroeconomic data.

# Descriptive Stats

## Review of Rules of Stats

### Rule 1

The variance of a constant is zero.

$$\sigma^2_c = \text{VAR}(c) = E[(c - E(c))^2] = 0$$

### Rule 2

Adding a constant value,  $c$ , to a random variable does not change the variance, because the expectation (mean) increases by the same amount.

$$\sigma^2_{x+c} = \text{VAR}(X + c) = E[((X_1 + c) - E(X + c))^2] = \text{VAR}(X)$$

### Rule 3

Multiplying a random variable by a constant increases the variance by the square of the constant.

$$\sigma^2_{cx} = \text{VAR}(cX) = c^2 \text{VAR}(X)$$

## Rule 4

The variance of the sum of two or more random variables is equal to the sum of each of their variances only when the random variables are independent.

$$\text{VAR}(X + Y) = \text{VAR}(X) + 2\text{COV}(X,Y) + \text{VAR}(Y)$$

and in terms of the sigma notation

$$\sigma_{x+y}^2 = \sigma_x^2 + 2\sigma_{xy} + \sigma_y^2$$

When two random variables are independent,  $\sigma_{xy} = 0$  so that

$$\sigma_{x+y}^2 = \sigma_x^2 + \sigma_y^2$$

## Rules for The Covariance

### Rule 1

The covariance of two constants,  $c$  and  $k$ , is zero.

$$\text{COV}(c, k) = E[(c - E(c))(k - E(k))] = E[(0)(0)] = 0$$

### Rule 2

The covariance of two independent random variables is zero.

$$\text{COV}(X, Y) = 0$$

### Rule 3

The covariance is a combinative as is obvious from the definition.

$$\text{COV}(X, Y) = \text{COV}(Y, X)$$

### Rule 4

The covariance of a random variable with a constant is zero.

$$\text{COV}(X, c) = 0$$

## Rule 5

Adding a constant to either or both random variables does not change their covariances.

$$\text{COV}(X + c, Y + k) = \text{COV}(X, Y)$$

## Rule 6

Multiplying a random variable by a constant multiplies the covariance by that constant.

$$\text{COV}(cX + kY) = ck * \text{COV}(X, Y)$$

## Rule 7

The additive law of covariance holds that the covariance of a random variable with a sum of random variables is just the sum of the covariances with each of the random variables.

$$\text{COV}(X + Y, Z) = \text{COV}(X, Z) + \text{COV}(Y, Z)$$

## Rule 8

The covariance of a variable with itself is the variance of the random variable. By definition:

$$\text{COV}(X, Y) = E[(X - E(X))(X - E(X))] = E[(X - E(X))^2] = \text{VAR}(X)$$

## Rules for Expected Value

### Rule 1

The expectation of a constant,  $c$ , is the constant.

$$E(c) = c$$

### Rule 2

Adding a constant value,  $c$ , to each term increases the mean, or expected value, by the constant.

$$E(X+c) = E(X)+c$$

## Rule 3

Multiplying a random variable by a constant value,  $c$ , multiplies the expected value or mean by that constant.

$$E(cX) = cE(X)$$

## Rule 4

The expected value or mean of the sum of two random variables is the sum of the means. This is also known as the additive law of expectation.

$$E(X+Y) = E(X)+E(Y)$$

## Types of Distributions

*Normal distribution:* bell shaped continuous probability density function (pdf) defined by parameters mean and standard deviation.

*Log normal distribution* (usually assumed to be the case for returns on many assets): A continuous distribution in which the log (e.g., returns) of a variable has a normal distribution.

*Uniform distribution:* a pdf in which all possible events (e.g., returns) are equally likely to happen.

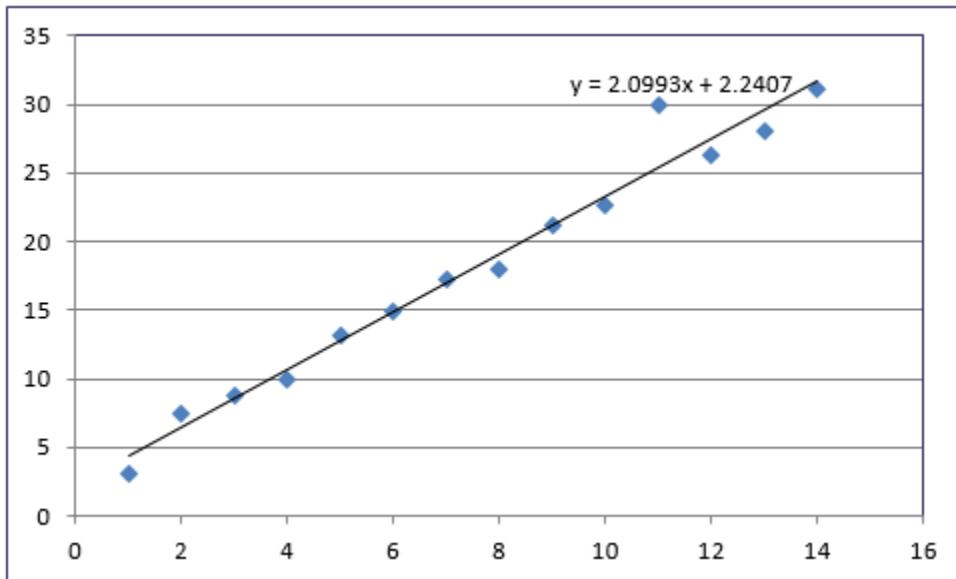
The following three distributions are especially important because they are used in hypothesis testing for a number of econometric procedures:

*t distribution* is symmetrical, bell-shaped, and similar to the standard normal curve. The higher the degrees of freedom the closer the t will resemble the standard normal distribution with mean zero and standard deviation of one.

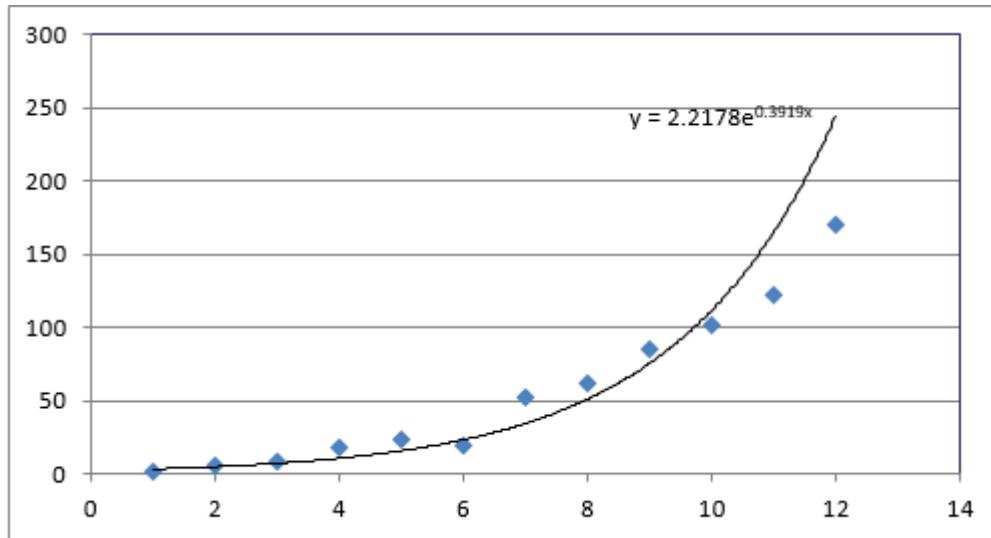
# Types of Econometric Models

## Plotting Data to Look for Patterns to Suggest Functional Form of Regression

Linear Relationship: The following plot of data (with a linear trend line/equation from Excel) suggests a linear model between x and y is appropriate.



In contrast, the following graph (again with a trend line) suggests a nonlinear relationship between x and y (exponential relationship between x and y)



This can be transformed into a linear relationship as follows

$$\ln(y) = a + b(x)$$

Some other possible transformations are given in the table below:

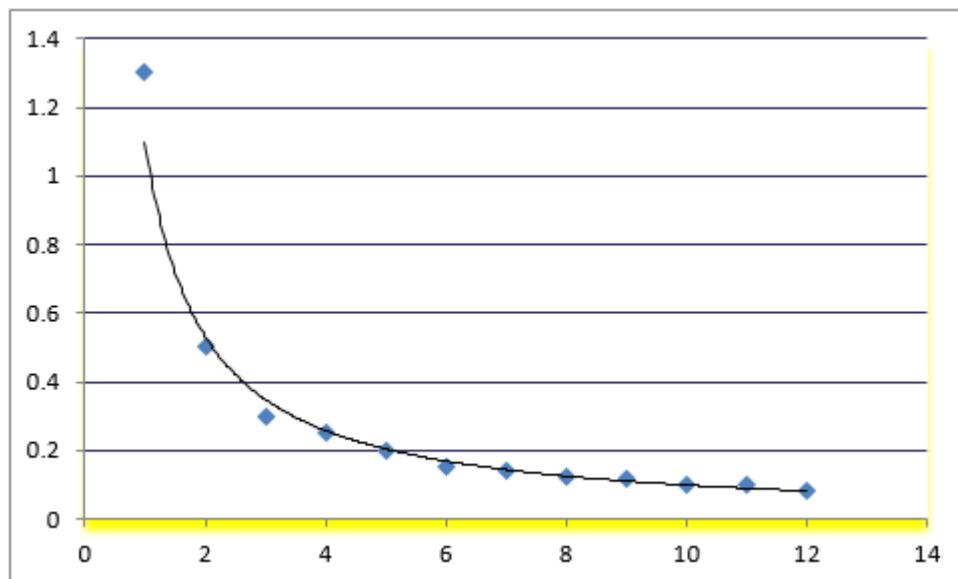
Table Of Transformations	
Method	Linearized estimating equation
Reciprocal	$1/y = a + b x$
Log model	$y = a + b \ln(x)$
Power model	$\ln(y) = a + b \ln(x)$
Quadratic	$\sqrt{y} = a + b x$

**Exercises:**

1. Can the following equation be expressed linearly? If so how?

$$y = e^{\beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + \epsilon}$$

2. For the data below, what kind of functional relationship exists between x and y and how this might be transformed using the transformations discussed in this section.



## Interpretation of Variables in Equations

### Dummy Variables

A dummy variable (binary independent variable) assumes a value of 1 if some criterion is true, zero otherwise.

For Example

D=0 if respondent is male

D=1 if respondent is female

CEO earnings (in thousands of USD)

$$Y = 90 + 50x - 25 D \quad [x \text{ is the number of years of experience}]$$

If D=1 (female) the CEO can expect to make \$25,000 less each year

### Categorical Variables

Related to dummies are categorical variables. These non-continuous variables can take on several values some of which may be meaningful others not.

Variable C takes on several values

## CEO's favorite color

1. Yellow
2. Blue
3. Green
4. Red

Preferring yellow to blue does not demonstrate any intensity in the variable being measured. That is, the green is not 3 times as intense as yellow. Simply running this variable in a regression doesn't mean anything. This type of variable can be transformed into meaning by creating a dummy variable.

For Example:

$D3 = 1$  if yellow is favorite color

$D3 = 0$  otherwise.

The regression coefficient for  $D3$  might be

$\text{CEO salary} = \dots + 40 D3 + \dots$

In this case a CEO whose favorite color is yellow earns \$40,000 more per year than one whose favorite color is something else. A categorical variable may express intensity and might include in a regression as a categorical variable without resort to the use of a dummy.

For example, C is average household income

\$0 to \$20,000

\$20,001 to \$40,000

\$40,001 to \$60,000

\$60,001 to \$80,000

More than \$80,000

In this case, intensity is represented by the data. The data represent a proxy for an unobservable variable, continuous income. For example, in a regression of

## Double Log Equation

$$\ln(y) = a + b \ln(x)$$

b is the percentage change in y resulting from a one percent change in x. For example, b=5 then a one percent increase in x results in a 5% increase in y.

## Exercises:

The following equation evaluating business majors starting salary contains two dummy variables D1 and D2

D1=1 if college major was accounting, zero otherwise.

D2=1 if college major was finance, otherwise zero.

Salary = 34 + 3.76D1 + 8.93 D2 [starting salary of business majors]

What is the starting salary of a business grad majoring in accounting?

What is the starting salary of a business grad majoring in finance?

What is the starting salary of other business majors?

## Finance Models

We are largely considering linear relationships in this course. As a consequence, we wish to have an independent variable, x, made a linear function.

## Linearizing Models

1. It maybe you are trying to model a nonlinear relationship between y and independent variable

$y: Y = aX^b$  (where a and b are constant parameters)

This can be linearized by taking the natural log of both sides:

$$\ln(Y) = \ln(a) + b \ln(X) \quad [\text{log-log relationship}]$$

This type of model can be estimated using linear regression which we shall talk about later. It is important to note that if we change x the change in y is as follows:

$dY = b(Y/X) dX$  [it depends on the values chosen for X and Y, e.g., mean values]

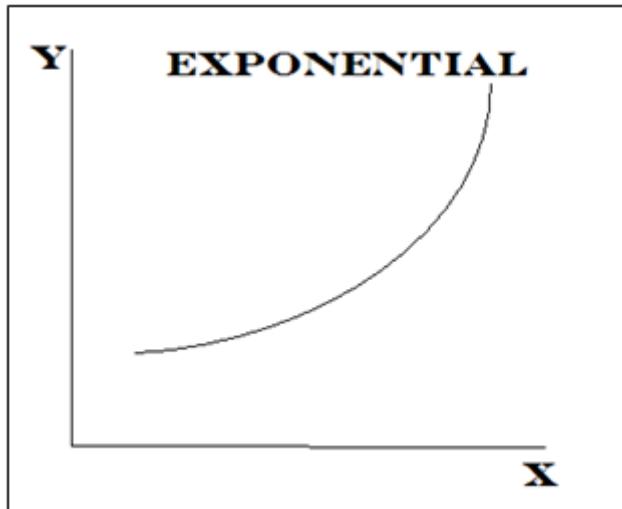
2. Suppose instead the relationship between x and y is

$$Y = e^{a+bx}$$

This nonlinear relationship can be expressed as:

$$\ln Y = \ln[e^{a+bx}] = a + bx$$

Bear in mind the relationship is not linear in X and Y but instead is illustrated by the graph below:



3. Another semi-log relationship is:

$$Y = a + b \ln(X)$$

4. Suppose that X is the independent variable and Y the dependent variable but a strict linear relationship does not exist Instead:

$$Y = a + bX + c X^2 \quad [\text{polynomial}]$$

This can still be estimated using the linear regression techniques of this course but we treat  $X^2$  as a new variable  $Z = X^2$ .

**Exercise:**

Make the following into a linear relationship:

$$Y = [a + bX]^{-1}$$

Hint: What is the definition for the new dependent variable?

# Econometrics, Finance Theory and Variables to Consider

## What Is Econometrics and Financial Econometrics?

Econometrics is the sometimes theoretical but usually empirical application of statistical methods in the study of economic data. The primary tool used in analysis is computer software often called canned software. Financial econometrics is identical but focuses on financial data and financial models such as the CAPM,

Financial econometrics looks at functional relationships between variables in the financial environment such as returns on financial assets or trends in variables such as bonds yields. As a consequence, we estimate equations that are linear or nonlinear in nature using the techniques developed in this course.

## Finance Topics Amenable to Econometrics

### Capital Asset Pricing Model (CAPM)

CAPM shows a linear relationship between risk and expected return and that is used in the pricing of risky securities. The security market line of the CAPM is defined as

$$\text{SML: } R_{i,t} - R_{f,t} = \alpha_i + \beta_i (R_{M,t} - R_{f,t}) + \varepsilon_{i,t}$$

1. The alpha coefficient ( $\alpha_i$ ) is the constant (intercept) in the security market line of the CAPM.
2. The alpha coefficient indicates how an investment has performed after accounting for the risk it involved.
3. If markets are *efficient* then  $\alpha=0$ . If  $\alpha<0$  the security has earned too little for its risk and if  $\alpha>0$  the investment has a risk in excess of the return for the given risk.

## French-Fama Model

The FF model is a factor model that expands on the capital asset pricing model (CAPM) by adding size and value factors along with the market risk factor in CAPM. They maintained that value and small cap stocks outperform markets on a consistent basis.

Fama and French started with the observation that two classes of stocks have tended to do better than the market as a whole: (i) small caps and (ii) stocks with a high book-value-to-price ratio (customarily called "value" stocks; their opposites are called "growth" stocks). Where book value equals assets minus liabilities and preferred shares.

They then added two factors to CAPM to reflect a portfolio's exposure to these two classes:

$$r - R_f = \beta_3 \times (K_m - R_f) + b_s \times SMB + b_v \times HML + \alpha$$

where  $r$  is the portfolio's return rate,

$R_f$  is the risk-free return rate (US T bills),

SMB (Small Minus Big) is the “average return on the three small portfolios minus the average return on the three big portfolios”,

SMB = $\frac{1}{3}$  (Small Value + Small Neutral + Small Growth)-  $\frac{1}{3}$  (Big Value + Big Neutral + Big Growth).

HML (High Minus Low) is the average return on the two value portfolios minus the average return on the two growth portfolios

HML = $\frac{1}{2}$  (Small Value + Big Value) -  $\frac{1}{2}$  (Small Growth + Big Growth)

and  $K_m$  is the return of the whole stock market. Momentum is sometimes used as another factor. The daily and weekly French-Fama factors can be found at the following:

[http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data\\_library.html](http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html)

## Portfolio Analysis

Like CAPM, the Fama and French model is used to explain the performance of portfolios via linear regression; only now the two extra factors give you two additional axes, so instead of a simple line the regression is a big flat thing that lives in the fourth dimension.

Even though you can't visualize this regression, you can still solve for its coefficients in a spreadsheet. The result is typically a better fit to the data points than you get with CAPM, with an r-squared in the mid-ninety percent range instead of the mid-eighties.

## Investing for the Future

Analyzing the past is a job for academics; most people are more interested in investing intelligently for the future. Here the approach is to use software tools and/or professional advice to find the exposure to the three factors that's appropriate for you, and then to invest in special index funds that are designed to deliver that level of the factors. You can try this tool on another website. When you try it, you should note that it in fact collapses all risk to the single factor of volatility, which is typical, and which brings up the earlier question of whether the additional factors really are measures of risk. In this case, how would you ever help somebody figure out what their "value tolerance" was? As a matter of fact, what would that question even *mean*?

## Arbitrage Pricing Theory (APT)

ABT states that the return of a financial asset is a linear function of a number of factors such as (a) macroeconomic factors (e.g., the market rate of interest, economic growth, inflation) and (b) market indices such as:

- short term interest rates;
- the difference in long-term and short-term interest rates;
- a diversified stock index such as the S&P 500 or NYSE Composite;
- oil prices
- gold and other precious metal prices
- Currency exchange rates

The APT produces a rate of return that can be checked against actual return to see if the asset is correctly priced.

Assumptions:

- security returns are a function of a number of factors
- sufficient securities so that firms-specific (idiosyncratic) risk can be diversified away
- well-functioning security markets do not allow for persistent arbitrage opportunities

Risky asset returns are said to follow a *factor intensity structure* if they can be expressed as:

$$r_i = a_i + b_{i1} F_1 + b_{i2} F_2 + \dots + b_{in} F_n + \varepsilon_i$$

Where

- $a_i$  is a constant for asset  $i$
- $F_i$  is a systematic factor where  $n$  such factors exist in this model
- $b$  is called the *factor loading* or the change in return resulting from a unit change in the factor
- $\varepsilon_i$  is called the idiosyncratic random shock with zero mean

Idiosyncratic shocks are assumed to be uncorrelated across assets and uncorrelated with the factors.

The APT states that if asset returns follow a factor structure then the following relation exists between expected returns and the factor sensitivities:

$$E(r_j) = r_f + b_{j1}RP_1 + b_{j2}RP_2 + \dots + b_{jn}RP_n$$

where

- $RP_k$  is the risk premium of the factor
- $r_f$  is the risk-free rate

## Yield Curve (Term Structure of Interest Rates)

Liquidity Preference Theory suggests market participants demand a premium for long-term investments in bonds (such as Treasuries, gilts or French OATs) compared to short term loans to the government. Even in a situation where no interest rate change is expected, the curve will slightly slope upwards because of the risk/liquidity premium investors demand from buying long term. This is the usual case (normal interest curve) but inverse or inverted curves are possible.

The logic behind the econometrics associated with the yield curve can be understood by looking at investor expectations about what future yields will be. Investors expect (expected value) that the return on a one year bond maturing in two years to be  $E_t(r_{1,t+1})$  or short-term rate one year from today; a one year bond maturing in three years has expected yield of  $E_t(r_{1,t+2})$ , etc. The yield on a n-period bond is equal to the average of the short-term bonds plus a risk/liquidity premium,  $P$  or

$$r_{n,t} = P + [r_{1,t} + E_t(r_{1,t+1}) + E_t(r_{1,t+2}) + \dots + E_t(r_{1,t+n-1})]/n$$

The expected one-year returns are all at time t. Assume the investor bases his/her expectation on the current one-year bond,  $r_{1,t}$ . The estimating equation for the n –year bond (long-term) is

$$r_{n,t} = a + b r_{1,t} \quad [\text{estimating equation for yield curve}]$$

Note that the theory (if it panned out) suggests that  $b=1$ . This can be tested using the t-test from OLS.

## Early Warnings Systems

It is known that financial imbalances commonly lead to widespread financial stress which may cause collapse of a bank or other company, financial crises and recessions.

It is therefore a crucial importance to detect and to predict potential financial stress. There are a host of early warning systems regarding financial or credit crises. One is the MIMIC that consists of two sets of equations

$$Y = bW + v$$

And

$$W=cX + u$$

Where Y is a crisis indicator. X is an observable potential crisis cause. W is a latent variable thought to represent the severity of the crisis. Disturbance terms u and v are explained later. Macroeconomic variables that can be used as crisis indicators include:

- Credit growth
- Property price growth
- Credit to GDP gap
- Equity price growth
- And so on

# Applied Hypothesis Testing Finance

## Applied Hypothesis in Finance

According to the monetary theory, the evolution of money supply has a significant influence on GDP. An increase of the money supply stimulates real GDP growth while a contraction of the money supply determines output losses. We want to study this linear relation between U.S. GDP and M2 monetary aggregate using a two-variable regression as following: GDP the dependent variable and M2 monetary aggregate independent variable.

We download GDP and M2 data from FRED database.

Period: 1970-2013 Frequency: annual data

gdp - Real Gross Domestic Product, 3 Decimal, Index 2007=100, Annual,  
Seasonally Adjusted Annual Rate m2 - M2 Money Stock, Index 2007=100, Annual,  
Seasonally Adjusted

We introduce the data in Python.

```
>>> m2=[8.4, 9.5, 10.8, 11.5, 12.1, 13.6, 15.5, 17.1, 18.3, 19.8, 21.5, 23.6, 25.6,  
28.5, 31.0, 33.5, 36.7, 38.0,  
  
40.2, 42.4, 44.0, 45.3, 46.0, 46.6, 46.8, 48.7, 51.1, 54.0, 58.5, 62.0, 65.9, 72.6,  
77.1, 81.1, 85.8, 89.3, 94.6, 100.0, 109.7, 113.8, 117.8, 129.3, 139.9, 147.4]  
  
>>> gdp=[7.4, 8.1, 9.1, 10.1, 10.9, 12.0, 13.2, 14.8, 16.9, 18.6, 20.4, 22.4, 23.2,  
25.8, 28.2, 30.3, 31.8, 34.2,  
  
36.9, 39.2, 41.0, 42.8, 45.6, 47.9, 50.9, 53.1, 56.4, 59.8, 63.5, 67.6, 71.3, 72.9,  
75.6, 80.5, 85.5, 91.1, 95.8, 100.0, 99.1, 99.2, 103.7, 107.5, 111.2, 116.3]
```

We import the Python modules we need in our analysis.

```
>>> import numpy as np  
>>> import statsmodels.api as sm  
>>> from matplotlib.finance import quotes_historical_yahoo
```

We apply the logarithm function to GDP and M2 variables and we implement the regression using statsmodels module from Python.

```
>>> m=np.log(m2)  
>>> gdp2=np.log(gdp)  
>>> m=sm.add_constant(m)  
>>> model=sm.OLS(gdp2, m)  
>>> results=model.fit()  
>>>
```

We obtained the following results:

The estimated linear relation:

```
>>> print results.summary()
                OLS Regression Results
-----
Dep. Variable:                      y      R-squared:                 0.988
Model:                             OLS      Adj. R-squared:            0.988
Method:                            Least Squares      F-statistic:             3605.
Date:                    Fri, 28 Nov 2014      Prob (F-statistic):       2.38e-42
Time:                           13:59:43      Log-Likelihood:          45.086
No. Observations:                  44      AIC:                   -86.17
Df Residuals:                      42      BIC:                   -82.60
Df Model:                           1
-----
              coef    std err        t      P>|t|      [95.0% Conf. Int.]
const     -0.1603     0.065   -2.466      0.018     -0.291     -0.029
x1         1.0267     0.017   60.042      0.000      0.992     1.061
-----
Omnibus:                     0.360      Durbin-Watson:           0.150
Prob(Omnibus):                 0.835      Jarque-Bera (JB):        0.135
Skew:                         -0.136      Prob(JB):               0.935
Kurtosis:                      2.995      Cond. No.                 19.7
-----
>>>
```

$$GDP = 1.0267M2 - 0.1603$$

According to the model a 1% increase of the M2 monetary aggregate determines an increase of U.S. GDP by 1.0267 %.

Dependent variable: y (in our case GDP)

Independent variable: x1 (in our case M2 monetary aggregate)

Method of estimation: Ordinary Least Squares (OLS) Method

Degrees of freedom of the model: 1

Degrees of freedom of the residuals: 42

Akaike Information Criterion (AIC): -86.17

Bayesian Information Criterion (BIC): -82.60

An information criterion should be as low as possible.

Number of observations: 44

Coefficient of determination (R-squared): 0.988

Adjusted coefficient of determination (Adj. R-squared): 0.988

According to the coefficient of determination, M2 explains 98.8 % of GDP evolution in U.S.

t-test

Null Hypothesis: The estimated coefficient is zero

Alternative hypothesis: The estimated coefficient is not zero in our regression:

For the constant: t is -2.466 and the p-value=0.018

For the M2 coefficient: t is 60.042 and the p-value=0.000

The low p-values indicate us that the constant as well as M2 coefficient are statistically different from zero.

Skewness: -0.136

Kurtosis: 2.995

An assumption of the linear regression is that the residuals are normally distributed. Skewness should be close to zero while kurtosis should be close to 3 when the residuals are normally distributed. In our case the registered values are close to the theoretic ones which indicate us that the residuals come from a normal distribution.

Durbin-Watson statistic: 0.150

Durbin-Watson statistic shows if there is autocorrelation of residuals. A DW statistic close to 2 indicates that there is no autocorrelation. In our case, the value is far from two which indicates that we face autocorrelation. Introducing new variables in the regression eliminates autocorrelation. In fact GDP is influenced by dozens of indicators (inflation, industrial production, interest rates, capacity utilization, foreign demand, foreign direct investments, investors' confidence etc) and not just by the evolution of the money supply.

Jarque-Bera (JB): 0.135

JB test indicates us if the residuals are normally distributed or not. A JB close to zero indicates that residuals come from a normal distribution.

Prob (JB): 0.935

# Model Specification

## Predicting House Prices with Regression

Big data changed our world. We receive a huge quantity of information each hour, minute or second. We are able to see the interest and photos of different individuals using social networks. We can see the weather forecast as well as real-time temperature in our city. We have access to real-time information about financial markets evolution. People have always been interested to know the future. Can we use science to know the future? Can we use big data to forecast future evolution? The answer is yes, we can.

We want to predict house prices evolution in Boston using a database of the city which includes 506 instances and 14 features. For more details, you can visit the following site: <http://archive.ics.uci.edu/ml/datasets/Housing>.

Machine learning will be used in order to make the desired projections. Machine learning is a branch of Artificial Intelligence that is focused on the construction and study of algorithms that can learn from the data. The following three concepts are found at a machine learning problem:

- Learning to solve a task (develop an algorithm)
- Need some experience (database)
- Measure of performance (how well we are solving our task).



*Scikit-learn is an open source Python library for implementing machine learning algorithms.*

Machine learning can be used to solve different problems like: classification, image recognition and forecasting. In this lab, we will use machine learning for making projections. A detailed description of machine learning in scikit-learn module is provided by Raul Garreta and Guillermo Moncecchi – “Learning scikit-learn: Machine learning in Python”.

The first step is to import Boston database from sklearn.datasets. Then we load the database in Python. We can see the dimensions of our database using print function. The function tells us that the Boston database has 506 lines and 14 columns. We use boston.feature\_names function if we want to see the 14 features considered in the database.

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> from sklearn.datasets import load_boston
>>> boston = load_boston()
>>> print boston.data.shape
(506L, 13L)
>>> print boston.feature_names
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
>>> print np.max(boston.target), np.min(boston.target), np.mean(boston.target)
50.0 5.0 22.5328063241
>>> |
```

Source: Raul Garreta, Guillermo Moncecchi – “Learning Scikit Learn: Machine Learning in Python”

The function `boston.DESCR` provides a detailed description of each feature.

```
>>> boston.DESCR
'Boston House Prices dataset\n\nNotes\n-----\nData Set Characteristics:\n  :Number of Instances: 506\n  :Number of Attributes: 13 numeric/categorical predictive\n  :Median Value (attribute 14) is usually the target\n  :Attribute Information (in order):\n    - CRIM      per capita crime rate by town\n    - INDUS     proportion of non-retail business acres per town\n    - NOX       nitric oxides concentration (parts per 10 million)\n    - RM        average number of rooms per dwelling\n    - AGE       proportion of owner-occupied units built prior to 1940\n    - DIS       weighted distances to five Boston employment centres\n    - RAD       index of accessibility to radial highways\n    - TAX       full-value property-tax rate per $10,000\n    - PTRATIO   pupil-teacher ratio by town\n    - B         1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town\n    - LSTAT    % lower status of the population\n    - MEDV     Median value of owner-occupied homes in $1000's\n  :Missing Attribute Values: None\n\nC\nreator: Harrison, D. and Rubinfeld, D.L.\nThis is a copy of UCI ML housing dataset.\nhttp://archive.ics.uci.edu/ml/datasets/Housing\nThis dataset was taken from the Statlib library which is maintained at Carnegie Mellon University.\n\nThe Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic prices and the demand for clean air', J. Enviro n. Economics & Management, vol.5, 81-102, 1978. Used in Belsley, Kuh & Welsh, 'Regression diagnostics...'. Wiley, 1980. N.B. Various transformations are used in the table on pages 244-261 of the latter.\n\nThe Boston house-price data has been used in many machine learning papers that address regression problems.\n\n***References***\n - Bels ley, Kuh & Welch, 'Regression diagnostics: Identifying Influential Data and Sources of Collinearity', Wiley, 1980. 244-261.\n - Quinlan, R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the Tenth International Conference of Machine Learning, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.\n - many more! (see h ttp://archive.ics.uci.edu/ml/datasets/Housing)\n'
>>>
```

*Source: Raul Garreta, Guillermo Moncecchi – “Learning Scikit Learn: Machine Learning in Python”*

We slice the learning set into training and testing datasets.

```
>>> from sklearn.cross_validation import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(boston.data, boston.target, test_size=0.25, random_state=33)
>>> from sklearn.preprocessing import StandardScaler
>>> scalerX = StandardScaler().fit(X_train)
>>> scalery = StandardScaler().fit(y_train)
>>> X_train = scalerX.transform(X_train)
>>> y_train = scalery.transform(y_train)
>>> X_test = scalerX.transform(X_test)
>>> y_test = scalery.transform(y_test)
```

*Source: Raul Garreta, Guillermo Moncecchi – “Learning Scikit Learn: Machine Learning in Python”*

Forecasting is equivalent to solving a regression problem according to the machine learning theory. We implement a function that trains the model and evaluates its performance using:

- The coefficient of determination
- Cross validation iterator

We have selected the coefficient of determination to evaluate the performance of our results. The coefficient of determination measures the variation explained by the model and it should be close to one.

```
>>> from sklearn.cross_validation import *
>>> def train_and_evaluate(clf, X_train, y_train):
...     clf.fit(X_train, y_train)
...     print "Coefficient of determination on training set:",clf.score(X_train, y_train)
...     # create a k-fold cross validation iterator of k=5 folds
...     cv = KFold(X_train.shape[0], 5, shuffle=True, random_state=33)
...     scores = cross_val_score(clf, X_train, y_train, cv=cv)
...     print "Average coefficient of determination using 5-fold crossvalidation:",np.mean(scores)
... 
```

Source: Raul Garreta, Guillermo Moncecchi – “Learning Scikit Learn: Machine Learning in Python”

We implement a linear model called SGRegressor which tries to minimize the squared loss.

```
>>> from sklearn import linear_model
>>> clf_sgd = linear_model.SGDRegressor(loss='squared_loss', penalty=None, random_state=42)
>>> train_and_evaluate(clf_sgd,X_train,y_train)
Coefficient of determination on training set: 0.743303511411
Average coefficient of determination using 5-fold crossvalidation: 0.707860049111
```

Source: Raul Garreta, Guillermo Moncecchi – “Learning Scikit Learn: Machine Learning in Python”

We obtained a coefficient of determination of 0.743303511411. The calculated hyperplane coefficients are presented bellow:

```
>>> print clf_sgd.coef_
[-0.07641527  0.06963738 -0.05935062  0.10878438 -0.06356188  0.37260998
-0.02912886 -0.20180631  0.08463607 -0.05534634 -0.19521922  0.0653966
-0.36990842]
```

Source: Raul Garreta, Guillermo Moncecchi – “Learning Scikit Learn: Machine Learning in Python”

We selected penalty=None in the model presented above. The penalization parameter avoids overfitting by penalizing the hyperparameters that have the too large coefficients.

We choose penalty='l2' – l2 norm (the squared sum of coefficients) in the next example.

```
>>> clf_sgd1 = linear_model.SGDRegressor(loss='squared_loss', penalty='l2', random_state=42)
>>> train_and_evaluate(clf_sgd1, X_train, y_train)
Coefficient of determination on training set: 0.743300618503
Average coefficient of determination using 5-fold crossvalidation: 0.707861811562
>>>
```

Source: Raul Garreta, Guillermo Moncecchi – “Learning Scikit Learn: Machine Learning in Python”

We obtained a coefficient of determination of 0.743300618503. In conclusion, we did not obtain a significant improvement as compared to the previous example.

## Predicting House Prices with Support Vector Machines for Regression

Support Vector Machines (SVM) is a machine learning model that provides reliable results in many applications. SVM was proposed in the mid '90s and is based on statistical learning. If we consider the points in a multidimensional space, a hyperplane separates points (instances) of one class from the rest. SVM obtains the hyperplanes that pass through the widest possible gaps between instances of different classes using an optimization technique. The main advantages of SVM are the following:

- Efficient in high dimensional spaces (problems which have many features to learn from)
- Efficient when there is a high-dimensional space with few instances
- Efficient in terms of memory usage as we do not need all the points in the learning space
- Solves nonlinear problems

The importance of SVM in finance:

- Predicting futures, house, stock and commodity prices
- Forecasting FX evolution
- Trading commodities
- Mining stock market tendency Credit scoring

In this chapter we want to forecast house prices in Boston using SVM for regression.

$(x_i, y_i)$  – training data

$i = 1 \dots m$

$x_i$  – input data

$y_i$  – output data

The regression function is written as following:

$$y = (\mathbf{z}^* \mathbf{x}_i) + l$$

$\mathbf{z}^*$  - weight vector

$l$  – normal

Weight vector and normal are obtained from the following function:

$$\min_{z^*, b, \xi^*} \frac{1}{2} \|z^*\|^2 + F \sum_{i=1}^I (s_i + s_i^*)$$

$$\begin{cases} ((z^* x_i) + l) - y_i \leq \varepsilon + s_i \\ y_i - ((z^* x_i) + l) \leq \varepsilon + s_i^*, i = 1, \dots, q \\ s_i^* \geq 0, \quad i = 1, \dots, q \end{cases}$$

$\varepsilon$  – intensive loss function

$F$  – punishment coefficient

$s_i^*$  - slack variable

The nonlinear regression function is the following:

$$y = \sum_{i=1}^q (\alpha'_i - \alpha_i) K(x_i, x) + l'$$

$$\sum_{i=1}^q (\alpha'_i - \alpha_i) = 0$$

$$0 \leq \alpha'_i$$

$$\alpha_i \leq P$$

$$i=1, \dots, q$$

$$\alpha'' = (\alpha_1, \alpha'_1, \dots, \alpha_q, \alpha'_q)^T$$

We implement Support Vector Machines (SVM) for regression

```
>>> from sklearn import svm
>>> clf_svr = svm.SVR(kernel='linear')
>>> train_and_evaluate(clf_svr, X_train, y_train)
Coefficient of determination on training set: 0.71886923342
Average coefficient of determination using 5-fold crossvalidation: 0.707838419194
>>> clf_svr_poly = svm.SVR(kernel='poly')
>>> train_and_evaluate(clf_svr_poly, X_train, y_train)
Coefficient of determination on training set: 0.904109273301
Average coefficient of determination using 5-fold crossvalidation: 0.779288545488
>>> clf_svr_rbf = svm.SVR(kernel='rbf')
>>> train_and_evaluate(clf_svr_rbf, X_train, y_train)
Coefficient of determination on training set: 0.900132065979
Average coefficient of determination using 5-fold crossvalidation: 0.833662221567
>>>
```

Source: Raul Garreta, Guillermo Moncecchi – “Learning Scikit Learn: Machine Learning in Python”

We chose a linear kernel in the first example and we obtained a coefficient of determination of 0.71886923342. According to the results provided, we have not obtained an improvement as compared to the linear regression presented in the first example.

We chose a polynomial kernel in the second example. We obtained a coefficient of determination of 0.904109273301 which is a much better result as compared to the linear estimate.

In the third example, we implemented a Radial Basis Function (RBF) kernel and we obtained a coefficient of determination of 0.833662221567.

# Introduction to OLS

## Two-Variable Regression

Many models in econometrics assume a linear relationship between a dependent variable, Y and one (or more) independent variables, X: The independent variable has different names including explanatory variable, regressor, predictor, stimulus, exogenous, covariate or control variable. The Y can also go by many names including endogenous, predicted, response, outcome and controlled variable.

$$Y = \beta_1 + \beta_2 X + u$$

where  $u$ , known as the disturbance, or error, term, is a random (stochastic) variable that has well-defined probabilistic properties. The disturbance term  $u$  may well represent all those factors that affect consumption but are not taken into account explicitly.

A simple business example of a possible linear relationship might be sales,  $y$ , and advertising expenditures,  $x$  (measured as each additional \$10,000 spent on ads). Suppose the relationship is calculated and the following linear relationship is estimated:

$$Y = 200 + 2 x$$

For example, if advertising increases by \$10,000 (one unit of  $x$ ) then sales will increase by 2 times \$10,000 = \$20,000. A simple and perhaps unrealistic model but a model.

## Things to Bear in Mind

1. (Deterministic vs. Correlation) Econometrics usually entails specifying a functional (such as linear) relationship between variables. The independent variable is thought to determine the values of the dependent variable. In many instances the relationship may simply reflect correlation between X and Y. Some relationships are quite deterministic as when we measure the growth of a plant and the amount of sun and water it receives.
2. (Population vs. Sampling) We are rarely able to observe an entire population of data we want to model. Instead econometrics typically relies on sampling of data when we make estimates.

## Linear Relationship and Sampling

Suppose there exists a linear dependence of Y on values of X. For a given value of X, Y is not always the same value but the expected value of Y is given by:

$$E(Y|X) = a + bX \quad (\text{Y conditional upon X})$$

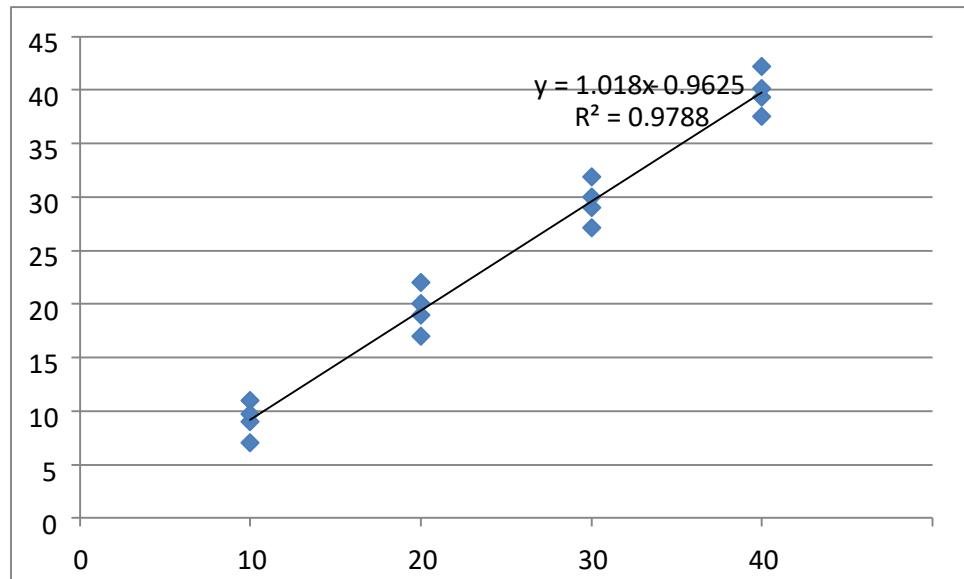
The expected value of Y conditional upon X reflects the population values (infinite sampling) for a and b.

**Example:**

Consider the Keynesian concept of the consumption function (from macroeconomics). This says that as the disposable income, X, increases by a dollar a household will increase its consumption by something less than a dollar (e.g., .9). We conduct a sample of income and household consumption:

X	Y	X	Y	X	Y	X	Y
10	7	20	17	30	27.1	40	37.5
10	9	20	19	30	29	40	39.3
10	11	20	20	30	30	40	40.1
10	9.7	20	22	30	31.9	40	42.2

For a given X they tend to regress towards the consumption function but there exists a dispersion about the true line as depicted in the graph below:



$R^2$  is the goodness of fit we shall define later.

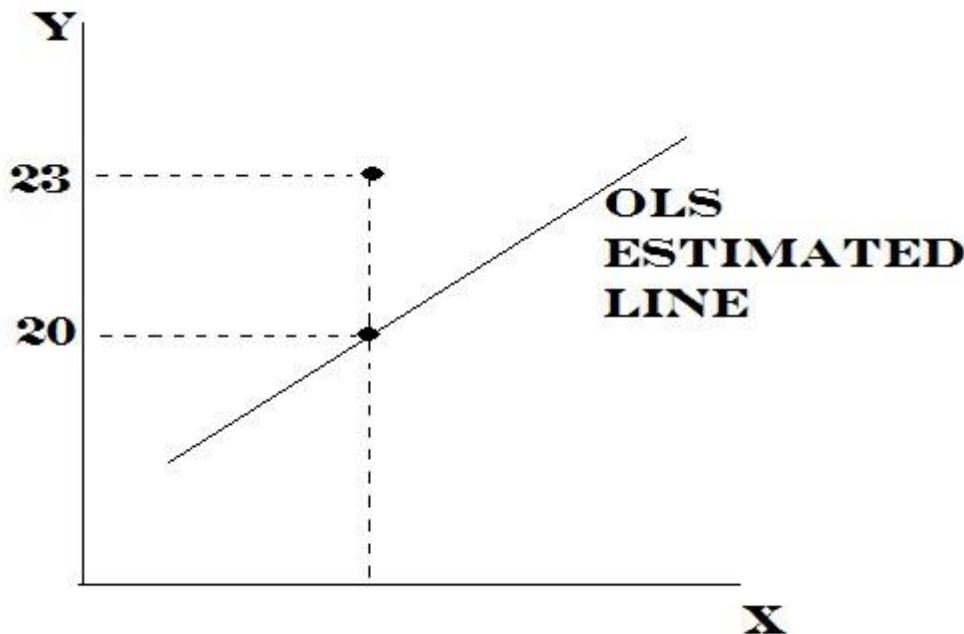
Note: this was produced in Excel which is handy for creating trend lines in its graphs.

## Derivation of Formulas Used in Regression of OLS

For a simple regression ( $Y_i = \beta_0 + \beta_1 X_i$ ) with only one independent variable, X. the derivation of the OLS parameter estimates is as follows. The error in a regression line is for observation i is

$$\text{Error: } u_i = Y_i - \hat{Y}_i = Y_i - (\beta_0 + \beta_1 X_i)$$

An error is represented in the graph below. The regression line predicts Y to be 20. The actual observed value is 23. The error is the difference 23-20. We square the errors because they would otherwise cancel out positive with negative.



OLS uses estimators that minimize the sum of the squared errors (SSE) or Min  
 $SSE = \sum_{i=1}^n u_i^2$  (n is the number of observations in the sample)  $\frac{\partial(SSE)}{\partial \beta_0} = 0$  (EQ 1) and  $\frac{\partial(SSE)}{\partial \beta_1} = 0$  (EQ2)

Eq1 and EQ 2 imply the following minimization conditions:

$$0 = n (\bar{y} - \beta_0 - \beta_1 \bar{x})^2 \quad (\text{from EQ1})$$

And

$$0 = \{ \beta_1 - \sum(X_i - \bar{x})(Y_i - \bar{y}) / \sum(X_i - \bar{x})^2 \}^2 \quad (\text{from EQ2})$$

Where the  $\bar{y}$  bar and  $\bar{x}$  bar are the data means. Solving simultaneously we get

Estimated value of  $\hat{\beta}_1 = \sum(X_i - \bar{x})(Y_i - \bar{y}) / \sum(X_i - \bar{x})^2$  Or  $\hat{\beta}_1 = \text{Cov}(X, Y) / \text{Var}(X)$  and

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

## Assumptions of OLS

To make statistical inference and perform hypothesis test we make assumptions about the true model error terms,  $\varepsilon$ .

The error terms are independently and identically distributed with a normal distribution:  $E \sim N(0, \sigma^2)$

This states:

- a.  $E(\varepsilon) = 0$
- b. The variance of the error terms does not change as values of the independent variables change

$E(\varepsilon | X) = 0$  (the errors are independent of the regressors) and

$\text{Var}(\varepsilon | X) = \text{constant} = \sigma^2$  (this is called the homoscedasticity assumption)

The error terms are independent of errors...

Equal variance of errors...

Normality of errors...

## OLS Using Excel

The idea of the ordinary least squares (OLS) principle is to choose parameter estimates that minimize the squared distance between the data and the model. In terms of the general, additive model,

$$Y_i = f(x_{i1}, \dots, x_{ik}; \beta_1, \dots, \beta_p) + \varepsilon_i$$

the OLS principle minimizes

$$\text{SSE} = \sum_{i=1}^n (y_i - f(x_{i1}, \dots, x_{ik}; \beta_1, \dots, \beta_p))^2$$

The least squares principle is sometimes called "nonparametric" in the sense that it does not require the distributional specification of the response or the error term, but it might be better termed "distributionally agnostic." In an additive-error model it is only required that the model errors have zero mean.

For example, the specification

$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

$$E[\varepsilon_i] = 0$$

is sufficient to derive ordinary least squares (OLS) estimators for  $\hat{\beta}_0$  and  $\hat{\beta}_1$  and to study a number of their properties. It is easy to show that the OLS estimators in this SLR model are

$$\hat{\beta}_1 = \left( \sum_{i=1}^n (Y_i - \bar{Y}) \sum_{i=1}^n (x_i - \bar{x}) \right) / \sum_{i=1}^n (x_i - \bar{x})^2$$

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{x}$$

Based on the assumption of a zero mean of the model errors, you can show that these estimators are

unbiased,  $E[\hat{\beta}_1] = \beta_1$ ,  $E[\hat{\beta}_0] = \beta_0$ . However, without further assumptions about the distribution, you cannot derive the variability of the least squares estimators or perform statistical inferences such as hypothesis tests or confidence intervals. In addition, depending on the distribution of the  $\varepsilon_i$ , other forms of least squares estimation can be more efficient than OLS estimation.

The conditions for which ordinary least squares estimation is efficient are zero mean, homoscedastic, uncorrelated model errors. Mathematically,

$$E[\varepsilon_i] = 0$$

$$\text{Var}[\varepsilon_i] = \sigma^2$$

$$\text{Cov}[\varepsilon_i, \varepsilon_j] = 0 \text{ if } i \neq j$$

The second and third assumption are met if the errors have an *iid* distribution—that is, if they are independent and identically distributed. Note, however, that the notion of stochastic independence is stronger than that of absence of correlation. Only if the data are normally distributed does the latter implies the former.

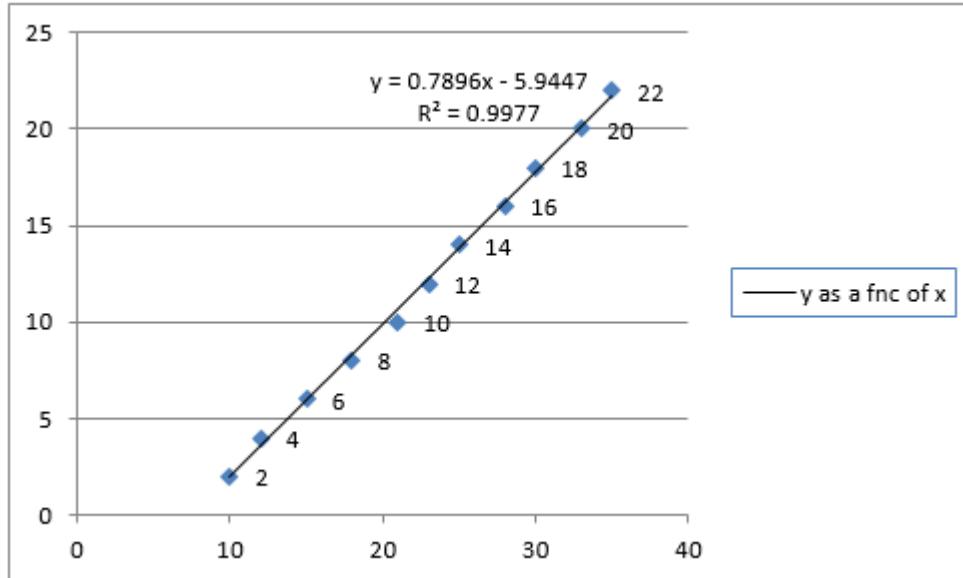
For the bivariate case that postulates a relationship between dependent variable,  $y$ , and independent variable,  $x$ , one can graph the supposed relationship. This

- Shows whether or not a linear relationship exists. It may also show some other graphical relationship such as exponential
- Allows one to quickly construct an ordinary least squares (OLS) equation and line available in Excel.

Example: Below are some data points in an Excel spreadsheet.

obs	....x	.....y
1	10	2
2	12	4
3	15	6
4	18	8
5	21	10
6	23	12
7	25	14
8	28	16
9	30	18
10	33	20
11	35	22

- Highlight this data in excel
- Click insert on the top menu bar
- Choose scatter graph (a bunch of dots)
- Right click on any point
- This gives the option for a trendline choose “add trendline”
- Under “options” choose “linear”
- Also choose at bottom “display equation on chart” and “R<sup>2</sup> “ then close. The graph should be like the one below.



This is a limited OLS routine with the intercept term equal to -5.944 and the slope of the line equal to .7896. The problem is that not a lot of information we would like is there. Because you have a graph you might want to explore alternative functional (nonlinear) forms.

Footnote notice the  $y$  values are on the right-hand side in the data. You see this on the graph as well.

## Exchange Rates and OLS

As an example, suppose that a forecaster for a Canadian company has been tasked with forecasting the USD/CAD exchange rate over the next year. He believes an econometric model would be a good method to use and has researched factors he thinks affect the exchange rate. From his research and analysis, he concludes the factors that are most influential are: the interest rate differential between the U.S. and Canada (INT), the difference in GDP growth rates (GDP), and income growth rate (IGR) differences between the two countries. The econometric model he comes up with is shown as:

$$\text{USD/CAD (1-year)} = z + a(\text{INT}) + b(\text{GDP}) + c(\text{IGR})$$

We won't go into the details of how the model is constructed, but after the model is made, the variables INT, GDP and IGR can be plugged into the model to generate a forecast. The coefficients a, b and c will determine how much a certain factor affects the exchange rate and direction of the effect (whether it is positive or negative). You can see that this method is probably the most complex and time-consuming approach of all the ones discussed so far. However, once the model is built, new data can be easily acquired and plugged into the model to generate quick forecasts.

## OLS in Python—Create Data in Python

How to fit a linear plot, for two arrays (or columns) of data:

- Python can produce a lot more information than the line trend in Excel
- The data used to run the program in python can be imported as a csv file from Excel or created in Python as an array [I don't recommend this if the data set is large]

These are the steps to follow when you are in Spyder when the data is set up as an array in Python:

```
# indicates a comment by me not part of the routine you should follow import pandas as pd import numpy as np import statsmodels.api as sm import matplotlib.pyplot as plt

#setup a array y for the dependent variable y=[2,4,6,8,10,12,14,16,18,20,22]

#similarly for the independent variable x:

x=[10,12,15,18,21,23,25,28,30,33,35]

#now create a constant term to be used in the OLS as follows

x = sm.add_constant(x)

#in order to see the ones are there enter the following; x

#enter the following and hit return est = sm.OLS(y, x) est = est.fit() est.summary()
```

Python returns the following regression output:

The screenshot shows the Spyder Python 2.7 IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, View, and Help. Below the menu is a toolbar with various icons. The left pane shows an 'Editor - C:\Users\gib\spyder2\py.statsmodels youtube.py' window containing the following code:

```

2 """
3 Spyder Editor
4
5 This is a temporary script file.
6 """

```

The right pane is the 'IPython console' window, which displays the following output from a script named 'youtube.py':

```

In [11]: est.summary()
C:\Users\gib\Anaconda\lib\site-packages\scipy\stats\stats.py:1205: UserWarning: kurtosistest only valid for n>=20 ...
continuing anyway, n=11
    int(n))
Out[11]:

```

OLS Regression Results

Dep. Variable:	y	R-squared:	0.998			
Model:	OLS	Adj. R-squared:	0.997			
Method:	Least Squares	F-statistic:	3952.			
Date:	Sat, 09 Aug 2014	Prob (F-statistic):	3.29e-13			
Time:	13:07:45	Log-Likelihood:	-2.4185			
No. Observations:	11	AIC:	8.837			
Df Residuals:	9	BIC:	9.633			
Df Model:	1					
coef	std err	t	P> t  [95.0% Conf. Int.]			
const	-5.944	70.303	-19.644	0.000	-6.629	-5.260
x1	0.7896	0.013	62.865	0.0000	0.761	0.818
Omnibus:	0.755	Durbin-Watson:	1.566			
Prob(Omnibus):	0.685	Jarque-Bera (JB):	0.424			
Skew:	-0.438	Prob(JB):	0.809			
Kurtosis:	2.600	Cond. No.	72.7			

In [12]:

At the bottom, there are tabs for Console, History log, and IPython console. Status information at the bottom includes Permissions: RW, End-of-lines: CRLF, Encoding: UTF-8, Line: 4, Column: 1, and Memory: 59 %.

- The estimates for the constant and slope terms are the same as in Excel
- There are a lot more statistics that we can talk about later

## OLS in Python—Importing Data from Excel

Most of the commands are the same but we import the data below:

```
import pandas as pd import numpy as np import statsmodels.api as sm  
import matplotlib.pyplot as plt  
  
#setup a array y for the dependent variable x = sm.add_constant(x)  
  
#in order to see the ones are there enter the following; x  
  
#enter the following and hit return est = sm.OLS(y, x) est = est.fit()  
est.summary()
```

## French-Fama Model Example

The FF model was discussed above. French and Fama added two factors to CAPM to reflect a portfolio's exposure to these two classes:

$$r - R_f = \beta_3 \times (K_m - R_f) + b_s \times SMB + b_v \times HML + \alpha$$

Document d70.csv below contains the weekly returns for GE for the year 1999 less the French-Fama riskfree return. The data set also contain French-Fama factors (courtesy of Dartmouth U and Ken French) that were discussed earlier

Mkt=  $K_m - R_f$  = the excess return of the market over the risk-free return

SMB (Small Minus Big) is the “average return on the three small portfolios minus the average return on the three big portfolios”.

SMB = 1/3 (Small Value + Small Neutral + Small Growth)

- 1/3 (Big Value + Big Neutral + Big Growth)

HML (High Minus Low) is the average return on the two value portfolios minus the average return on the two growth portfolios

HML = 1/2 (Small Value + Big Value) - 1/2 (Small Growth + Big Growth) and  $K_m$  is the return of the whole stock market.

```
import statsmodels.formula.api as smf url = 'c:/users/gib/documents/d70.csv'  
dat = pd.read_csv(url)  
results = smf.ols('ge ~ mkt + smb +hml', data=dat).fit()  
print results.summary()
```

## OLS Regression Results

=====									
Dep. Variable:	ge			R-squared:	0.772				
Model:	OLS			Adj. R-squared:	0.756				
Method:	Least Squares			F-statistic:	50.67				
Date:	Wed, 03 Sep 2014			Prob (F-statistic):	1.79e-14				
Time:	17:38:02			Log-Likelihood:	-59.842				
No. Observations:	49			AIC:	127.7				
Df Residuals:	45			BIC:	135.3				
Df Model:	3				=====				
coef	std err	t	P> t	[95.0% Conf. Int.]	=====				
Intercept	1.6725	0.130	12.888	0.000	1.411	1.934	mkt	0.4529	0.084
5.376	0.000	0.283	0.623	smb	-0.5878	0.096	-6.146	0.000	-0.780
0.395	hml	0.6651	0.114	5.818	0.000	0.435	0.895		
Omnibus:	52.336			Durbin-Watson:	2.092				
Prob(Omnibus):	0.000			Jarque-Bera (JB):	229.928				
Skew:	-2.906			Prob(JB):	1.18e-50				
Kurtosis:	11.879			Cond. No.	3.56				
=====									

## Suppressing the Constant Term in a Regression

Python code:

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import csv
url="C:\user\Gibs\documents\D5.csv"
dat=pd.read_csv(url)
results=smf.OLS('w~v+x-1', data=dat).fit()
results.summary()
```

# Stationarity

## Stationarity and Non-stationarity

Stationarity definition of a time series,  $y_t$ :

- a constant mean or  $E(y_t) = \mu$  (for all  $t$  / the mean does not vary with time)
- a constant variance or  $\text{Var}(y_t) = \sigma^2$
- no autocorrelation i.e.,  $\text{cov}(y_t, y_{t-k})$  is not a function of  $k$
- an autocovariance that does not depend on time i.e.,  $\text{cov}(y_t, y_{t-k})$  is not a function of time itself

If these hold the time series is called second-order stationarity or stationarity of order 2.

## Trends

1. A deterministic trend is a non-random function of time as in the following:

- Deterministic:  $y_t = \alpha + \beta t + u_t$  (where  $u$  is white noise)
- The expected value of this equation is  $\alpha + \beta t$ . If this were estimated by OLS or GLS we would get:  $y_t = \alpha^* + \beta^* t$  (where the stars denote estimates).
- The original equation can be detrended so that a new series is created that is stationary:

$$u^* = y_t - \alpha^* - \beta^* t$$

2. Stochastic trends are random and vary with time and look as follows:

$$y_t = \alpha + y_{t-1} + u_t$$

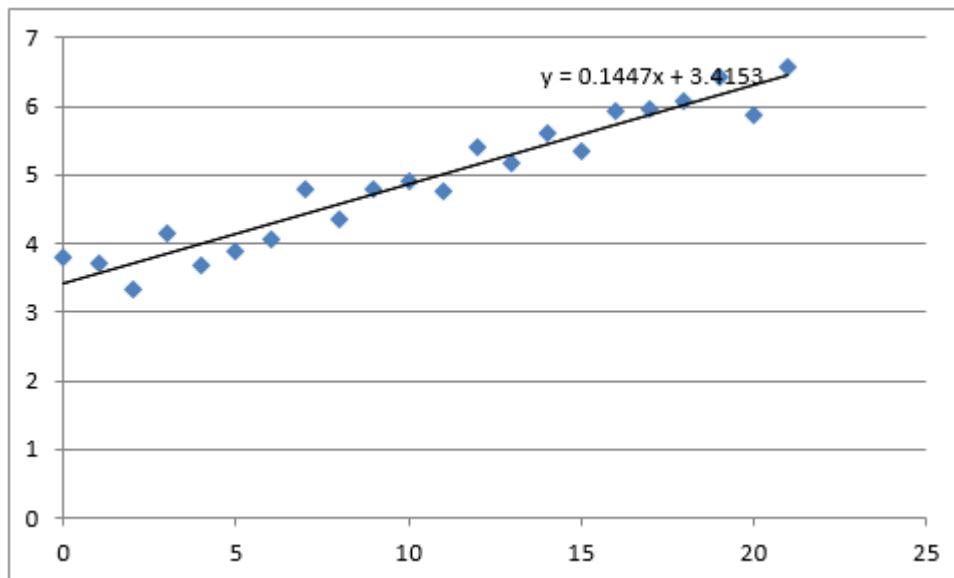
- This is called a random walk with drift or a local level model (where  $u$  is white noise).
- This can be made stationary by subtracting

$$\Delta y_t = y_t - y_{t-1} = \alpha + u_t$$

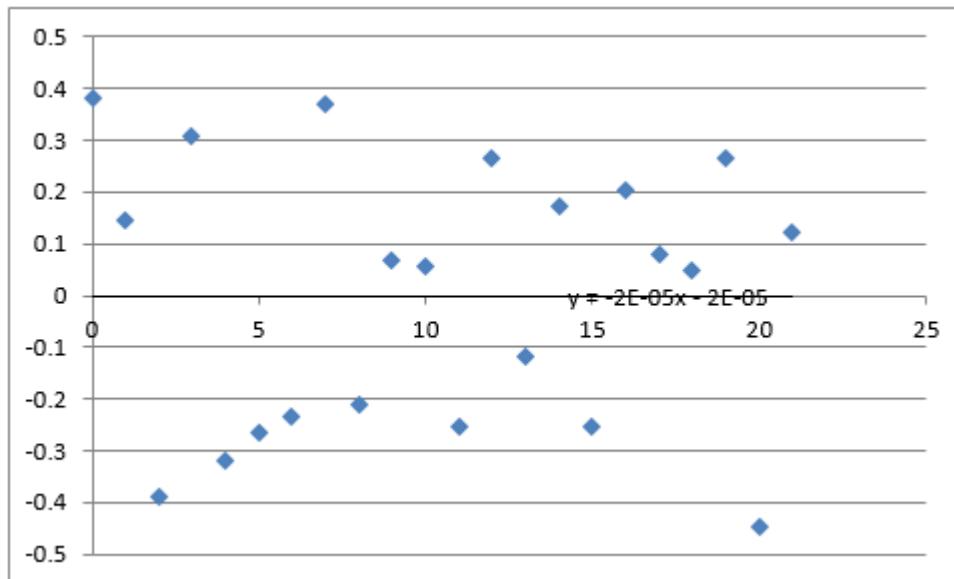
**Note:** When trying to make data stationary, it is important to recognize if a trend in time exists or a stochastic trend.

**Example:**

Look at the following data and see if a trend exists using Excel:



After detrending by subtracting the excel equation  $3.4153 + .1447x$  from the series the graph looks like white noise as follows:



## Spurious Regression

The importance of stationarity can be illustrated by the concept of spurious (false) regression. This involves regressing one nonstationary process against another. For example, if we regress the consumer price index (which usually increases over time) against an unrelated variable (e.g., sunspot activity) the correlation may be very high even though no theoretical connection is likely to exist between the two series.

## Mean Reversion

Mean *reversion* states that a stationary process will always tend toward the mean because of a constant variance which limits the range the process travels. In finance, mean reversion is profitably exploited by short selling when the asset price is above the mean and buying back when the price is below the mean.

In financial trading, when reversion does not occur it is referred to as *momentum*.

Ways to make a time series stationary:

- Differencing
- Variable transformation: taking logs
- Variable transformation: taking square roots

## Definitions of Non-stationarity

Processes that have a mean that varies over time or a variance that varies over time. There is also the issue of covariances that will be discussed later.

## Why Do We Need to Test for Non-Stationarity?

- The persistence of shocks will be infinite for nonstationary series
- If two variables are trending over time, a regression of one on the other could have a high R<sup>2</sup> even if the two are totally unrelated. This is called spurious regression.

## Types of Non-stationarity

1. Deterministic stochastic non-stationarity:

Model  $y_t = \mu + \alpha y_{t-1} + u_t$  ( $\mu$  is the mean  $u$  is random disturbance)

2. Random walk

$$y_t = \mu + \alpha y_{t-1} + u_t$$

if  $\alpha=1$  we have a “unit root problem” and the series is non-stationary. If  $|\alpha|<1$  then the series is stationary. If  $|\alpha|>1$  then the series is explosive and useless.

**Note:** consider following. The differenced data is stationary since

$$y_t = y_{t-1} + u_t$$

$$\Delta y_t = y_t - y_{t-1} = u_t$$

And the expected value of  $u$  is zero.

### 3. Random walk with a drift

$$y_t = \alpha + y_{t-1} + u_t$$

As mentioned this is nonstationary but take differences:

$$\Delta y_t = y_t - y_{t-1} = \alpha + y_{t-1} - y_{t-1} + u_t$$

$$\Delta y_t = \alpha + u_t$$

$$E(\Delta y_t) = E(\alpha + u_t) = \alpha$$

$$\text{Var}(\Delta y_t) = \text{Var}(u_t) = \sigma^2 \text{ a constant over time.}$$

#### Example:

An example of a random walk with a drift occurs in the bond market. The relationship between the six and three-month T bill yields can be expressed as follows: Yield on six-month T bill =  $\alpha$  + yield on the three-month T bill Where  $\alpha$  represents a risk/liquidity premium.

### 4. The deterministic time process that is a function of time t:

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

### 5. Random walk with drift and a deterministic trend:

$$y_t = \alpha + y_{t-1} + \beta t + u_t$$

# Augmented Dickey-Fuller Test

## Augmented Dickey-Fuller Test

The testing procedure for the ADF test is the same as for the Dickey–Fuller test but it is applied to the model

$$\Delta y_t = \alpha + \gamma y_{t-1} + \beta t + \delta_1 \Delta y_{t-1} + \dots + \delta_{p-1} \Delta y_{t-p+1} + u_t$$

Where  $\alpha$  is a constant,  $\beta$  the coefficient on a time trend and  $p$  the lag order of the autoregressive process.

The unit root test is then carried out under the null hypothesis  $\gamma=0$  against the alternative hypothesis of  $\gamma < 0$ . Once a value for the test statistic

$$DF_\tau = \frac{\hat{\gamma}}{SE(\hat{\gamma})}$$

is computed it can be compared to the relevant critical value for the Dickey–Fuller Test. If the test statistic is less (this test is non-symmetrical so we do not consider an absolute value) than the (larger negative) critical value, then the null hypothesis of  $\gamma=0$  is rejected and no unit root is present. There are different values for the critical DF values depending on the type of test that is being conducted. Some tests do not include a trend,  $t$ , for example. DF critical values are summarized in the table below:

## Critical Values for the Dickey-Fuller Unit Root t-Test Statistics

Probability to the Right of Critical Value

Model Statistic N 1% 2.5% 5% 10% 90% 95% 97.5% 99%

Model I (no constant, no trend)									
ADF <sub>tr</sub>	25	-2.66	-2.26	-1.95	-1.60	0.92	1.33	1.70	2.16
	50	-2.62	-2.25	-1.95	-1.61	0.91	1.31	1.66	2.08
	100	-2.60	-2.24	-1.95	-1.61	0.90	1.29	1.64	2.03
	250	-2.58	-2.23	-1.95	-1.61	0.89	1.29	1.63	2.01
	500	-2.58	-2.23	-1.95	-1.61	0.89	1.28	1.62	2.00
	>500	-2.58	-2.23	-1.95	-1.61	0.89	1.28	1.62	2.00

Model II (constant, no trend)									
ADF <sub>tr</sub>	25	-3.75	-3.33	-3.00	-2.62	-0.37	0.00	0.34	0.72
	50	-3.58	-3.22	-2.93	-2.60	-0.40	-0.03	0.29	0.66
	100	-3.51	-3.17	-2.89	-2.58	-0.42	-0.05	0.26	0.63
	250	-3.46	-3.14	-2.88	-2.57	-0.42	-0.06	0.24	0.62
	500	-3.44	-3.13	-2.87	-2.57	-0.43	-0.07	0.24	0.61
	>500	-3.43	-3.12	-2.86	-2.57	-0.44	-0.07	0.23	0.60

Model III (constant, trend)									
ADF <sub>tr</sub>	25	-4.38	-3.95	-3.60	-3.24	-1.14	-0.80	-0.50	-0.15
	50	-4.15	-3.80	-3.50	-3.18	-1.19	-0.87	-0.58	-0.24
	100	-4.04	-3.73	-3.45	-3.15	-1.22	-0.90	-0.62	-0.28
	250	-3.99	-3.69	-3.43	-3.13	-1.23	-0.92	-0.64	-0.31
	500	-3.98	-3.68	-3.42	-3.13	-1.24	-0.93	-0.65	-0.32
	>500	-3.96	-3.66	-3.41	-3.12	-1.25	-0.94	-0.66	-0.33

The Python options for the ADF test are given below:

```
statsmodels.tsa.stattools.adfuller  
statsmodels.tsa.stattools.adfuller(x, maxlag=None,  
regression='c', autolag='AIC', store=False,  
regresults=False)[source]
```

## Augmented Dickey-Fuller Unit Root Test

The Augmented Dickey-Fuller test can be used to test for a unit root in a univariate process in the presence of serial correlation.

```
x : array_like, 1d data series maxlag : int  
Maximum lag which is included in test, default  
12*(nobs/100)^{1/4} regression : str {'c','ct','ctt','nc'}
```

Constant and trend order to include in regression \* 'c': constant only \* 'ct': constant and trend \* 'ctt': constant, and linear and quadratic trend \* 'nc': no constant, no trend

**autolag:** {'AIC', 'BIC', 't-stat', None}

if None, then maxlag lags are used if 'AIC' or 'BIC', then the number of lags is chosen to minimize the corresponding information criterium

't-stat' based choice of maxlag. Starts with maxlag and drops a lag until the t-statistic on the last lag length is significant at the 95 % level.

**store:** bool

If True, then a result instance is returned additionally to the adf statistic

**regresults:** bool

If True, the full regression results are returned.

**Parameters:**

**adf:** float

Test statistic

**pvalue:**

float

MacKinnon's approximate p-value based on MacKinnon

(1994) **usedlag:** int

Number of lags used.

**nobs:** int

Number of observations used for the ADF regression and calculation  
of the critical values.

**Returns:**

**critical values:** dict

Critical values for the test statistic at the 1 %, 5 %, and 10 % levels. Based on MacKinnon (2010)

**lcbest:** float

The maximized information criterion if autolag is not None.

**regresults:** RegressionResults instance

**The resstore:** (optional) instance of ResultStore an instance of a dummy class with results attached as attributes

## Notes

- The null hypothesis of the Augmented Dickey-Fuller is that there is a unit root, with the alternative that there is no unit root. If the p-value is above a critical size, then we cannot reject that there is a unit root.
- The p-values are obtained through regression surface approximation from MacKinnon 1994, but using the updated 2010 tables. If the p-value is close to significant, then the critical values should be used to judge whether to accept or reject the null.

## In Python:

```
>>> import numpy as np
>>> import pandas as pd
n [8]: import statsmodels.tsa.stattools
#the following is a list of returns
>>> a=[1,2,-1,2,-3,0,.5,.8,-.21,2,-1,2,-3,0,.5,.8,-2,2,-1,2,-
3,0,.5,.8,-.21,2,-1,2,-3,0,.5,.8,-.2]
>>> statsmodels.tsa.stattools.adfuller(a, maxlag=None)
Out[10]:
(-1.7034844119211205, 0.42931810315407193, 7,
25, {'1%': -3.7238633119999998, '10%': -
2.632800399999998, '5%': -2.98648896},
49.908550173095492)
Testing for Mean Reversion A continuous mean-reverting time series can be represented by an Ornstein-Uhlenbeck stochastic differential equation:  $dxt = \theta(\mu - xt)dt + \sigma dW_t$ 
```

Where  $\theta$  is the rate of reversion to the mean,  $\mu$  is the mean value of the process,  $\sigma$  is the variance of the process and  $W_t$  is a Wiener Process or Brownian Motion. In a discrete setting the equation states that the change of the price series in the next time period is proportional to the difference between the mean price and the current price, with the addition of Gaussian noise. This property motivates the Augmented Dickey-Fuller Test, which we will describe below.

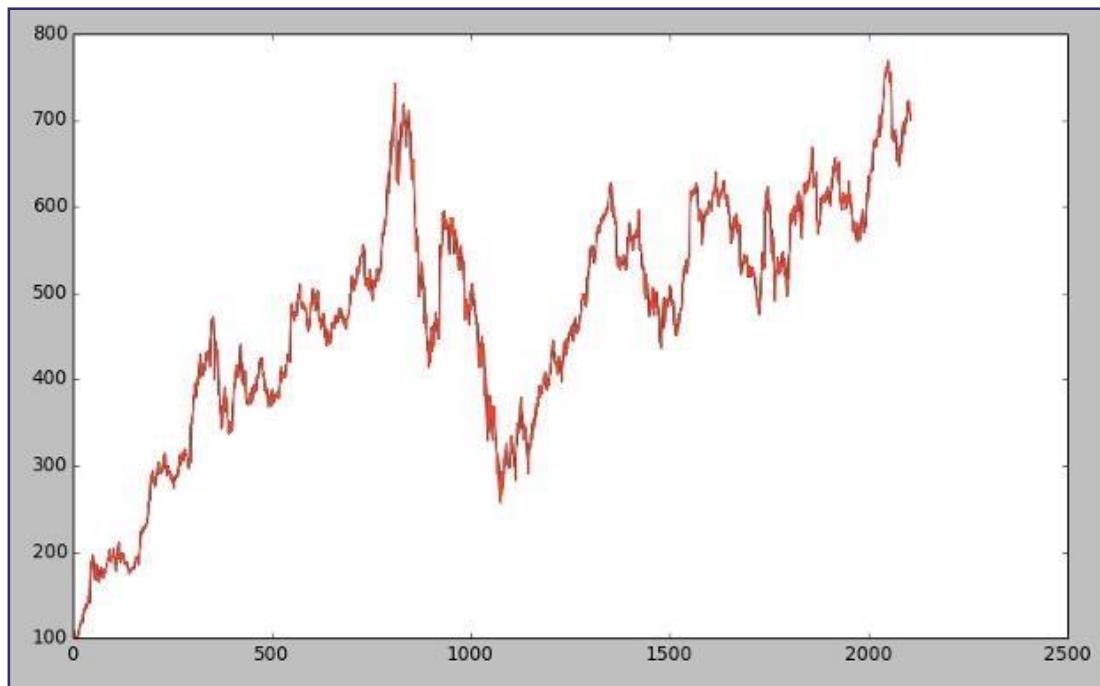
Augmented Dickey-Fuller (ADF) Test Mathematically, the ADF is based on the idea of testing for the presence of a unit root in an autoregressive time series sample. It makes use of the fact that if a price series possesses mean reversion, then the next price level will be proportional to the current price level. A linear lag model of order  $p$  is used for the time series:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \dots + \delta_{p-1} \Delta y_{t-p+1} + \epsilon_t$$

Where  $\alpha$  is a constant,  $\beta$  represents the coefficient of a temporal trend and  $\Delta y_t = y(t) - y(t-1)$ . The role of the ADF hypothesis test is to consider the null hypothesis that  $\gamma=0$ , which would indicate (with  $\alpha=\beta=0$ ) that the process is a random walk and thus non-mean reverting. If the hypothesis that  $\gamma=0$  can be rejected then the following movement of the price series is proportional to the current price and thus it is unlikely to be a random walk. So how is the ADF test carried out? The first task is to calculate the test statistic ( $DFT$ ), which is given by the sample proportionality constant  $\hat{\gamma}$  divided by the standard error of the sample proportionality constant:

$$DFT = \hat{\gamma} / SE(\hat{\gamma})$$

Dickey and Fuller have previously calculated the distribution of this test statistic, which allows us to determine the rejection of the hypothesis for any chosen percentage critical value. The test statistic is a negative number and thus in order to be significant beyond the critical values, the number must be more negative than these values, i.e. less than the critical values. A key practical issue for traders is that any constant long-term drift in a price is of a much smaller magnitude than any short-term fluctuations and so the drift is often assumed to be zero ( $\beta=0$ ) for the model. Since we are considering a lag model of order  $p$ , we need to actually set  $p$  to a particular value. It is usually sufficient, for trading research, to set  $p=1$  to allow us to reject the null hypothesis. To calculate the Augmented Dickey-Fuller test we can make use of the pandas and statsmodels libraries. The former provides us with a straightforward method of obtaining Open-High-Low-Close-Volume (OHLCV) data from Yahoo Finance, while the latter wraps the ADF test in an easy to call function. We will carry out the ADF test on a sample price series of Google stock, from 1st January 2000 to 1st January 2013.



Google price series from 2000-01-01 to 2013-01-01

Here is the Python code to carry out the test:

```
# Import the Time Series library import  
statsmodels.tsa.stattools as ts  
# Import Datetime and the Pandas DataReader from datetime  
import datetime from pandas.io.data import DataReader  
  
# Download the Google OHLCV data from 1/1/2000 to  
1/1/2013 goog = DataReader("GOOG", "yahoo",  
datetime(2000,1,1), datetime(2013,1,1))  
  
# Output the results of the Augmented Dickey-Fuller test for  
Google  
# with a lag order value of 1 ts.adfuller(goog['Adj Close'], 1)
```

Here is the output of the Augmented Dickey-Fuller test for Google over the period. The first value is the calculated test-statistic, while the second value is the p-value. The fourth is the number of data points in the sample. The fifth value, the dictionary, contains the critical values of the test-statistic at the 1, 5 and 10 percent values respectively.

```
(-2.1900105430326064,  
 0.20989101040060731,  
 0,  
 2106,  
 {'1%': -3.4334588739173006,  
 '10%': -2.5675011176676956,  
 '5%': -2.8629133710702983},  
 15436.871010333041)
```

Since the calculated value of the test statistic is larger than any of the critical values at the 1, 5 or 10 percent levels, we cannot reject the null hypothesis of  $\gamma=0$  and thus we are unlikely to have found a mean reverting time series. An alternative means of identifying a mean reverting time series is provided by the concept of stationarity, which we will now discuss.

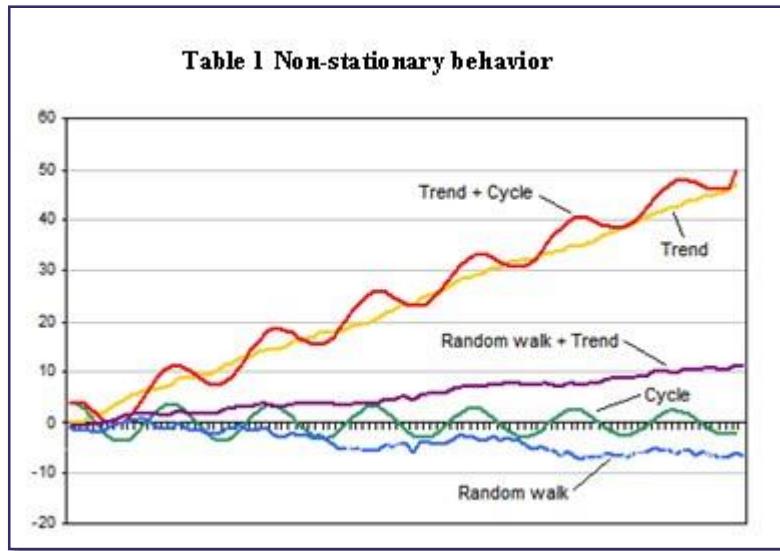
Financial institutions and corporations as well as individual investors and researchers often use financial time series data (such as asset prices, exchange rates, GDP, inflation and other macroeconomic indicators) in economic forecasts, stock market analysis or studies of the data itself.

But refining data is key to being able to apply it to your stock analysis. It is possible to isolate the data points that are relevant to your stock reports. Stationarity means here that the mean, variance and intertemporal correlation structure remains constant over time. Non-stationarities can either come from deterministic changes like trend or seasonal fluctuations, or the stochastic properties of the process, if for example the autoregressive process has a unit root, that is one of the roots of the lag polynomial is on the unit circle. In the first case, we can remove the deterministic component by de-trending or deseasonalization.

## Cooking Raw Data

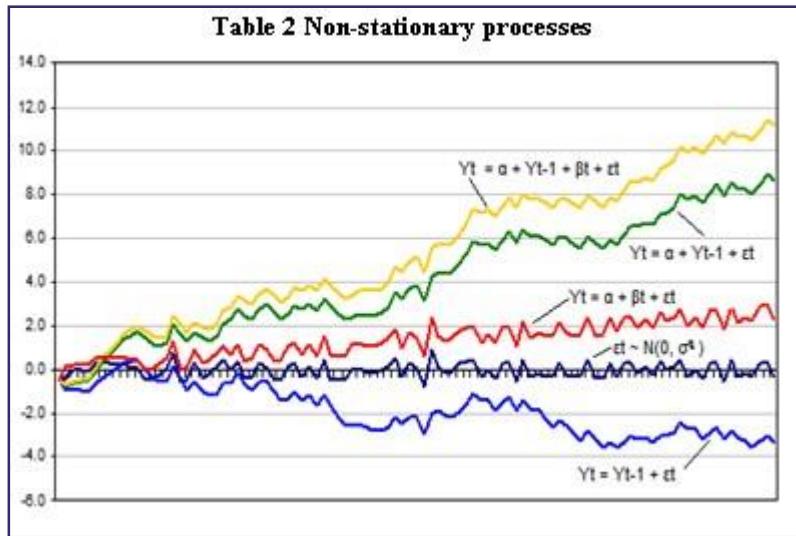
Data points are often non-stationary or have means, variances and covariances that change over time. Non-stationary behaviors can be trends, cycles, random walks or combinations of the three.

Non-stationary data, as a rule, are unpredictable and cannot be modeled or forecasted. The results obtained by using non-stationary time series may be spurious in that they may indicate a relationship between two variables where one does not exist. In order to receive consistent, reliable results, the nonstationary data needs to be transformed into stationary data. In contrast to the non-stationary process that has a variable variance and a mean that does not remain near, or returns to a long-run mean over time, the stationary process reverts around a constant long-term mean and has a constant variance independent of time.



#### Types of Non -Stationary Processes

Before we get to the point of transformation for the non-stationary financial time series data, we should distinguish between the different types of the non-stationary processes. This will provide us with a better understanding of the processes and allow us to apply the correct transformation. Examples of nonstationary processes are random walk with or without a drift (a slow steady change) and deterministic trends (trends that are constant, positive or negative, independent of time for the whole life of the series).



### Pure Random Walk ( $Y_t = Y_{t-1} + \varepsilon_t$ )

Random walk predicts that the value at time "t" will be equal to the last period value plus a stochastic (non-systematic) component that is a white noise, which means  $\varepsilon_t$  is independent and identically distributed with mean "0" and variance " $\sigma^2$ ". Random walk can also be named a process integrated of some order, a process with a unit root or a process with a stochastic trend. It is a non-mean reverting process that can move away from the mean either in a positive or negative direction. Another characteristic of a random walk is that the variance evolves over time and goes to infinity as time goes to infinity; therefore, a random walk cannot be predicted.

### **Random Walk with Drift ( $Y_t = \alpha + Y_{t-1} + \varepsilon_t$ )**

If the random walk model predicts that the value at time "t" will equal the last period's value plus a constant, or drift ( $\alpha$ ), and a white noise term ( $\varepsilon_t$ ), then the process is random walk with a drift. It also does not revert to a long-run mean and has variance dependent on time.

### **Deterministic Trend ( $Y_t = \alpha + \beta t + \varepsilon_t$ )**

Often a random walk with a drift is confused for a deterministic trend. Both include a drift and a white noise component, but the value at time "t" in the case of a random walk is regressed on the last period's value ( $Y_{t-1}$ ), while in the case of a deterministic trend it is regressed on a time trend ( $\beta t$ ). A non-stationary process with a deterministic trend has a mean that grows around a fixed trend, which is constant and independent of time.

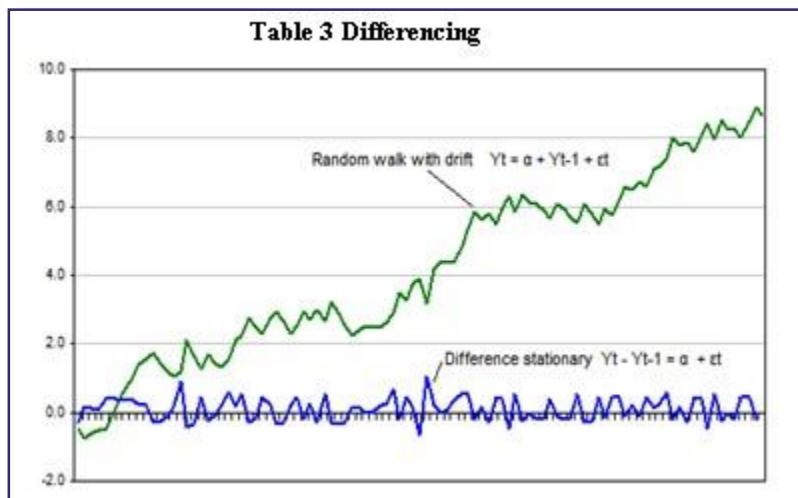
### **Random Walk with Drift and Deterministic Trend ( $Y_t = \alpha + Y_{t-1} + \beta t + \varepsilon_t$ )**

Another example is a non-stationary process that combines a random walk with a drift component ( $\alpha$ ) and a deterministic trend ( $\beta t$ ). It specifies the value at time "t" by the last period's value, a drift, a trend and a stochastic component. (To learn more about random walks and trends, see our *Financial Concepts* tutorial).

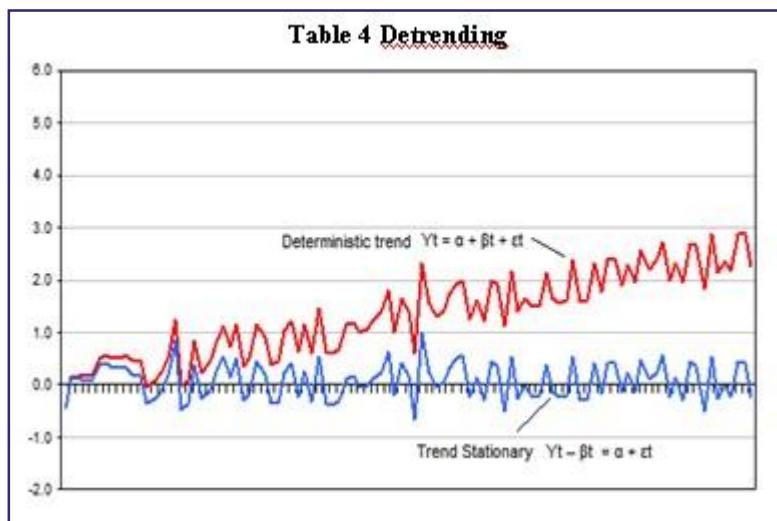
## Trend and Difference Stationary

A random walk with or without a drift can be transformed to a stationary process by differencing

(subtracting  $Y_{t-1}$  from  $Y_t$ , taking the difference  $Y_t - Y_{t-1}$ ) correspondingly to  $Y_t - Y_{t-1} = \varepsilon_t$  or  $Y_t - Y_{t-1} = \alpha + \varepsilon_t$  and then the process becomes difference-stationary. The disadvantage of differencing is that the process loses one observation each time the difference is taken.



A non-stationary process with a deterministic trend becomes stationary after removing the trend, or detrending. For example,  $Y_t = \alpha + \beta t + \varepsilon_t$  is transformed into a stationary process by subtracting the trend  $\beta t$ :  $Y_t - \beta t = \alpha + \varepsilon_t$ , as shown in Figure 4 below. No observation is lost when detrending is used to transform a non-stationary process to a stationary one.



In the case of a random walk with a drift and deterministic trend, detrending can remove the deterministic trend and the drift, but the variance will continue to go to infinity. As a result, differencing must also be applied to remove the stochastic trend.

Using non-stationary time series data in financial models produces unreliable and spurious results and leads to poor understanding and forecasting. The solution to the problem is to transform the time series data so that it becomes stationary. If the non-stationary process is a random walk with or without a drift, it is transformed to stationary process by differencing. On the other hand, if the time series data analyzed exhibits a deterministic trend, the spurious results can be avoided by detrending. Sometimes the nonstationary series may combine a stochastic and deterministic trend at the same time and to avoid obtaining misleading results both differencing and detrending should be applied, as differencing will remove the trend in the variance and detrending will remove the deterministic trend.

# Autocorrelation

## Autocorrelation

In finance, serial correlation is used by technical analysts to determine how well the past price of a security predicts the future price.

Problems created by autocorrelation: OLS are unbiased but not efficient.

## Durbin Watson Test

The Durbin Watson d statistic is given as

$$d = \sum_{t=2}^T (e_t - e_{t-1})^2 / \sum_{t=1}^T e_t^2$$

Where  $e$  is the residual from the regression at time  $t$  and  $T$  is the number of observations in a time series. The critical values of the DW table can be found at many places on the internet such as

[http://www.stat.ufl.edu/~winner/tables/DW\\_05.pdf](http://www.stat.ufl.edu/~winner/tables/DW_05.pdf)

**Note:** this and most DW tables assume a constant term in the regression and no lagged dependent variables.

The DW statistic  $d$  is approximately equal to  $2(1 - r)$ , where  $r$  is the sample autocorrelation of the residuals. If  $d = 2$  it indicates no autocorrelation. The value of  $d$  lies between 0 and 4.

If  $d > 2$ , successive error terms are, on average, much different in value from one another, i.e., negatively correlated.

To test for positive autocorrelation at significance  $\alpha$ , the test statistic  $d$  is compared to lower and upper critical values ( $d_{L,\alpha}$  and  $d_{U,\alpha}$ ):

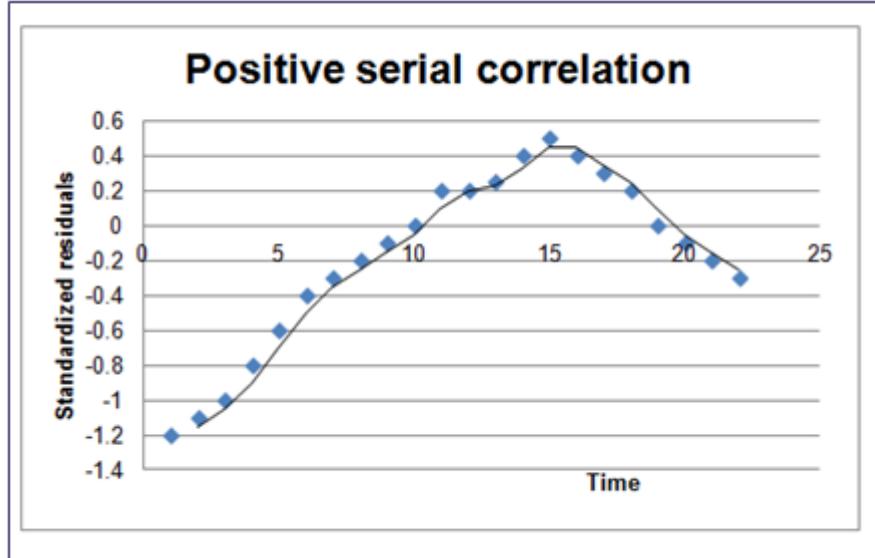
- If  $d < d_{L,\alpha}$ , there is statistical evidence that the error terms are positively autocorrelated.
- If  $d > d_{U,\alpha}$ , there is no statistical evidence that the error terms are positively autocorrelated.
- If  $d_{L,\alpha} < d < d_{U,\alpha}$ , the test is inconclusive.

Positive serial correlation is serial correlation in which a positive error for one observation increases the chances of a positive error for another observation.

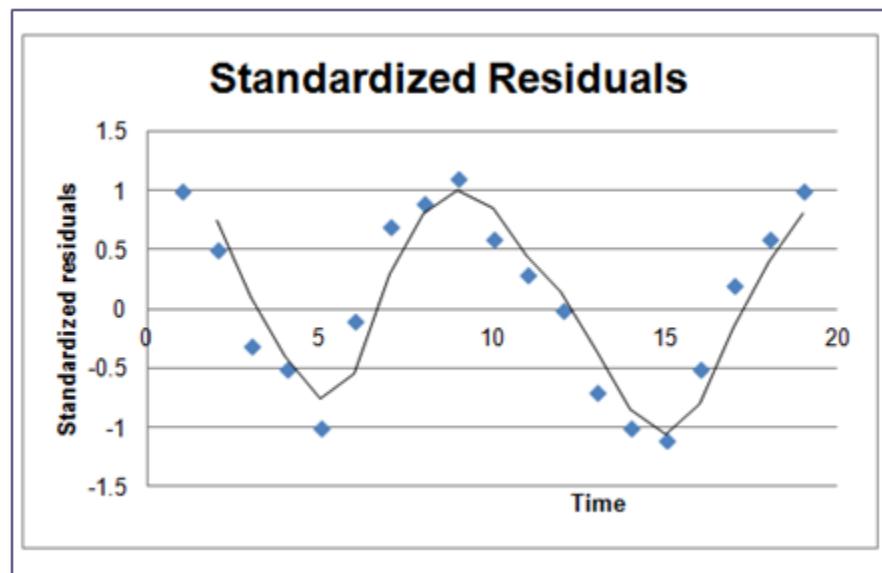
To test for negative autocorrelation at significance  $\alpha$ , the test statistic  $(4 - d)$  is compared to lower and upper critical values ( $d_{L,\alpha}$  and  $d_{U,\alpha}$ ):

- If  $(4 - d) < d_{L,\alpha}$ , there is statistical evidence that the error terms are negatively autocorrelated.
- If  $(4 - d) > d_{U,\alpha}$ , there is no statistical evidence that the error terms are negatively autocorrelated.
- If  $d_{L,\alpha} < (4 - d) < d_{U,\alpha}$ , the test is inconclusive.

**Positive Serial Correlation** (thought to be more common than the negative case) time series process in which positive residuals tend to be followed over time by positive error terms and negative residuals tend to be followed over time by negative residuals.



### Negative Serial Correlation Example:



## Hypothesis Testing with the DW—Positive Test

Suppose you had 30 observations( $n=30$ ) and two independent variables in the regression. You use the 5% level of confidence. From the DW table you should find the lower bound ( $d_L$ ) of the table equal to 1.28 and the upper bound ( $d_U$ ) equal to 1.57. If the statistic that is produced by the regression is say .4 (below the lower bound) we would reject the null hypothesis below in favor of the alternative:

$H_0$ : no autocorrelation

$H_1$ : positive autocorrelation

Let's say instead that the computed regression DW stat was 1.45. This is the "inconclusive" area of the DW analysis and we cannot say one way or the other. Suppose the DW stat from the regression were 1.90 then we do not reject the null hypothesis that there exists no serial correlation.

## Hypothesis Testing with the DW—Negative Test

Suppose you had 30 observations( $n=30$ ) and two independent variables in the regression. You use the 5% level of confidence. From the DW table you should find the lower bound ( $d_L$ ) of the table equal to 1.28 and the upper bound ( $d_U$ ) equal to 1.57. If the statistic that is produced by the regression is say .4 (below the lower bound) we would reject the null hypothesis below in favor of the alternative:

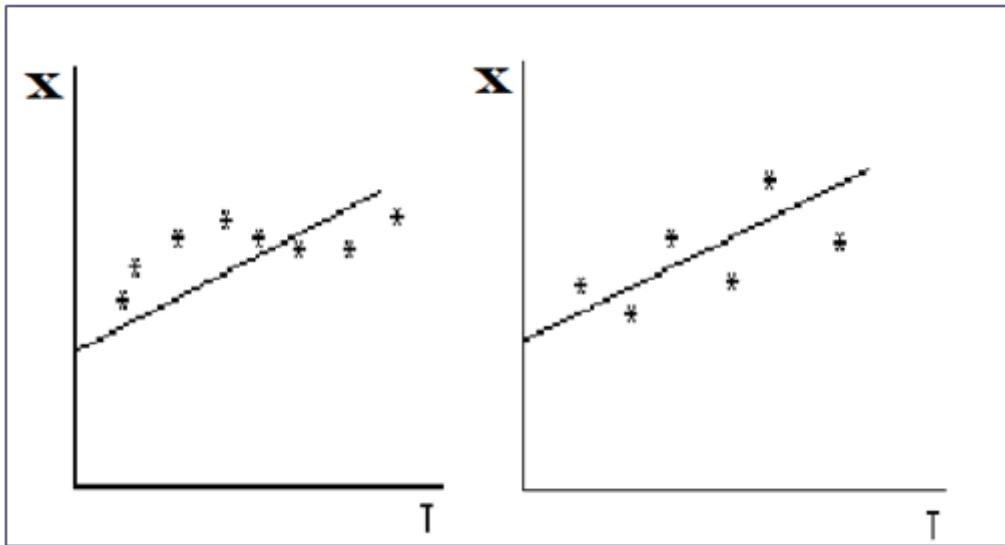
$H_0$ : no autocorrelation

$H_1$ : positive autocorrelation

Let's say instead that the computed regression DW stat was 1.45. This is the "inconclusive" area of the DW analysis and we cannot say one way or the other. Suppose the DW stat from the regression were 1.90 then we do not reject the null hypothesis that there exists no serial correlation.

**Exercise:**

For the time-series process, X, which of the graphs displays a. negative serial correlation, b positive serial correlation?



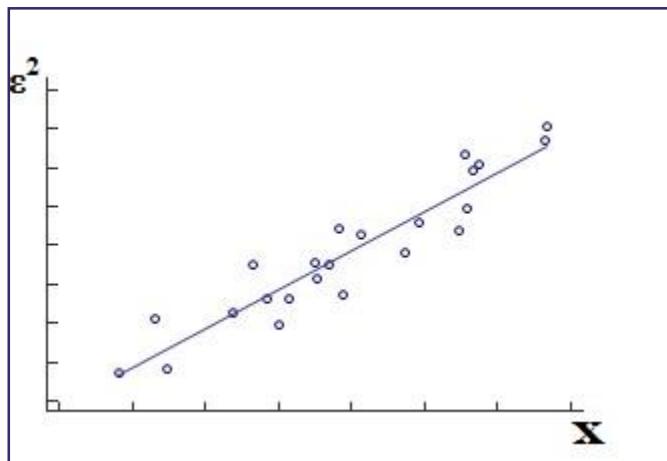
# Heteroscedasticity

## Heteroscedasticity

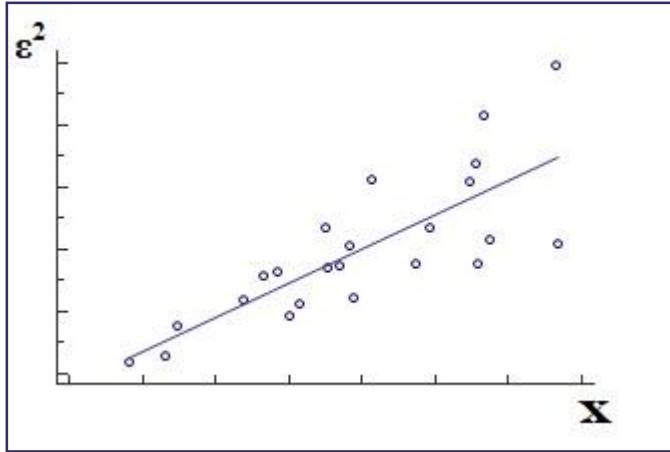
One of the assumptions of the classic linear regression model is homoscedasticity (same dispersion of errors). The error variances are constant. When unequal dispersion occurs, we have heteroscedasticity. Unequal variances of the error terms  $\varepsilon_i$ , occurs when

$$\text{var}(\varepsilon_i) = E[(\varepsilon_i)^2] - [E(\varepsilon_i)]^2 = E[(\varepsilon_i)^2] = (\sigma_i)^2.$$

## Homoscedasticity:



## Heteroscedasticity:



If the model we are regressing is:

$$\text{Model: } y = \beta_0 + \beta_1 x_1 + \dots + \beta_k x_k + \varepsilon^2$$

If homoscedasticity occurs

$$\text{Var}(\varepsilon | x_1, \dots, x_k) = \sigma^2$$

For heteroscedasticity

$$\text{Var}(\varepsilon | x_1, \dots, x_k) = \sigma^2 f(x_1, \dots, x_k)$$

For example:

$f = \delta_0 + \delta_1 x_1 + \delta_2 x_4$  (Only the first and forth regressors are implicated in the heteroscedasticity.)

In some instances, only one regressor may be responsible for the heteroscedasticity so we can observe the relationship graphically.

For example, consider a 4x4 matrix

$\sigma, 0, 0, 0$   
 $0, \sigma, 0, 0$   
 $0, 0, \sigma, 0$   
 $0, 0, 0, \sigma$

Where sigma is constant throughout. This is called homoscedasticity or constant variance.

For heteroscedasticity,

$$\text{Var}(u_i | X_i) = \sigma^2_i$$

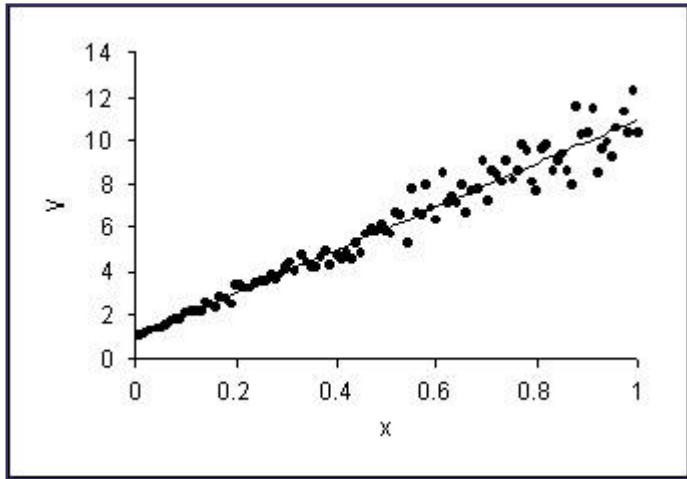
If heteroscedasticity is present the variance is not stationary so the above matrix would be written as

$\sigma_1, 0, 0, 0, 0,$   
 $\sigma_2, 0, 0$   
 $0, 0, \sigma_3, 0$   
 $0, 0, 0, \sigma_4$

In the bivariate relationship in the graph below the error terms increase as x increases.

## Tests for Heteroscedasticity

Graph the residuals against suspected explanatory variables as below:



In this instance the variance increases as x increases.

The problem here is that the heteroscedasticity may not be result of one variable but instead from a combination of the regressors:

$$\text{Var}(\varepsilon | x_1, \dots, x_k) = \sigma^2 f(x_1, \dots, x_k)$$

For example:

$$f = \delta_0 + \delta_1 x_1 + \delta_2 x_4 + \delta_3 x_5 + \delta_4 x_7$$

White test states that if disturbances are homoscedastic then squared errors are, on average, constant. Thus, regressing the squared residuals against explanatory variables should result in a low R squared.

Steps of the White Test:

1. Regress  $Y$  against your various explanatory variables using OLS
2. Compute the OLS residuals,  $e_1 \dots e_n$  and square them.
3. Regress squared residuals  $e_i^2$  against a constant, all of the explanatory variables, their squares and possible interactions ( $x_1$  times  $x_2$ ) between the explanatory variables ( $p$  slopes total)
4. Compute  $R^2$  from (c)
5. Compare  $nR^2$  to the critical value from the Chi-squared distribution with  $p$  degrees of freedom.

## Whites Test in Python

```

import pandas as pd
import numpy as np
import statsmodels.formula.api as smf

url= 'c:/users/gib/documents/d5.hetero.csv'

dat = pd.read_csv(url)

results = smf.ols('y ~ x + z', data=dat).fit()

print results.summary()

```

OLS Regression Results						
Dep. Variable:	y	R-squared:	0.987			
Model:	OLS	Adj. R-squared:	0.986			
Method:	Least Squares	F-statistic:	625.1			
Date:	Sun, 09 Nov 2014	Prob (F-statistic):	6.50e-16			
Time:	10:49:08	Log-Likelihood:	-20.824			
No. Observations:	19	AIC:	47.65			
Df Residuals:	16	BIC:	50.48			
Df Model:	2					
coef std err t P> t  [95.0% Conf. Int.]						
Intercept	6.1365	10.031	0.612	0.549	-15.129	27.402
x	0.3912	0.050	7.806	0.000	0.285	0.497
z	-0.3145	0.542	-0.580	0.570	-1.464	0.835
Omnibus: 7.704 Durbin-Watson: 0.627						
Prob(Omnibus): 0.021 Jarque-Bera (JB): 5.430						
Skew: 1.277 Prob(JB): 0.0662						
Kurtosis: 3.576 Cond. No. 1.68e+03						

The residuals are found as follows:

```
Y = dat['y']
```

```

pr = results.predict() # produces the predicted values of y using
the regression supra res=Y-pr # produces the
residuals ressq=res*res #square of residuals

```

### Table: Chi-Square Probabilities

The areas given across the top are the areas to the right of the critical value. To look up an area on the left, subtract it from one, and then look it up (ie: 0.05 on the left is 0.95 on the right)

df	0.995	0.99	0.975	0.95	0.90	0.10	0.05	0.025	0.01	0.005
1	---	---	0.001	0.004	0.016	2.706	3.841	5.024	6.635	7.879
2	0.010	0.020	0.051	0.103	0.211	4.605	5.991	7.378	9.210	10.597
3	0.072	0.115	0.216	0.352	0.584	6.251	7.815	9.348	11.345	12.838
4	0.207	0.297	0.484	0.711	1.064	7.779	9.488	11.143	13.277	14.860
5	0.412	0.554	0.831	1.145	1.610	9.236	11.070	12.833	15.086	16.750
6	0.676	0.872	1.237	1.635	2.204	10.645	12.592	14.449	16.812	18.548

The squared and interaction term is given below;

```
X = dat['x']      # pulls the x variable out of the data set dat  
Z = dat['z']      # pulls the z variable out of the data set dat  
XX=X*X          # X squared  
ZZ=Z*Z          # Z squared  
XZ=X*Z          # interaction term
```

### Run the regression with squared and interaction terms:

```
results = smf.ols('ressq ~ X + Z+XX+ZZ+XZ',
data=dat).fit() print results.summary()
```

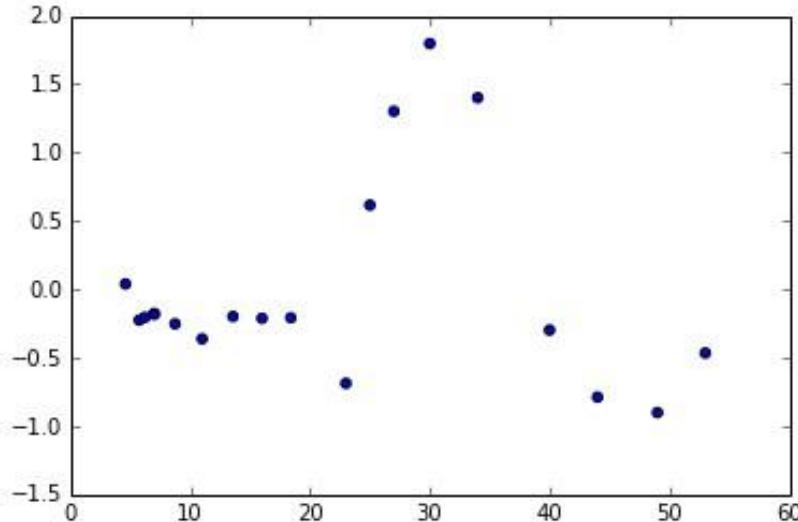
OLS Regression Results						
Dep. Variable:	ressq	R-squared:	0.486			
Model:	OLS Adj.	R-squared:	0.288			
Method:	Least Squares	F-statistic:	2.458			
Date:	Sun, 09 Nov 2014	Prob (F-statistic):	0.0889			
Time:	11:28:36	Log-Likelihood:	-17.356			
No. Observations:	19	AIC:	46.71			
Df Residuals:	13	BIC:	52.38			
Df Model:	5					
coef	std err	t	P> t	[95.0% Conf. Int.]		
Intercept	-608.4115	438.676	-1.387	0.189	-1556.114	339.291
X	6.5041	4.687	1.388	0.189	-3.621	16.629
Z	64.0950	46.685	1.373	0.193	-36.762	164.952
XX	-0.0189	0.013	-1.480	0.163	-0.047	0.009
ZZ	-1.6893	1.242	-1.360	0.197	-4.372	0.993
XZ	-0.3359	0.249	-1.349	0.200	-0.874	0.202
Omnibus:	8.323	Durbin-Watson:		1.459		
Prob(Omnibus):	0.016	Jarque-Bera (JB):		6.009		
Skew:	0.888	Prob(JB):		0.0496		
Kurtosis:	5.106	Cond. No.		3.13e+0		

Reject the null hypothesis of homoscedasticity if  $nR^2 >$ Chi-Squared critical value with  $p=5$  df. Note the critical value from the table is 11.07 (5% significance level). The calculated  $nR^2$  ( $n=19$ ) is  $19 \times .486 = 9.23$ . We do not reject in this instance.

A graph of the residuals (res) and the variable X is shown below:

```
In [26]: import matplotlib.pyplot as plt
```

```
In [27]: plt.scatter(X,res)
```



### White Test in Python:

Try the following command for the White Test in python after the original model is run:

```
test = sms.het_white(results.resid, results.model.exog)
```

## Breusch-Pagan Tests

BP #1 The Breusch-Pagan Langrange-Multiplier (LM) Test is the same as the White Test except the econometrician selects the explanatory variables to include in the auxiliary equation. This may or may not result in a more powerful test than the White test.

Example: suppose the form chosen is

$$\text{Squared residuals} = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 Z^2 + \beta_4 XZ$$

Again, apply the chi-squared test but  $p=4$ . Compare  $nR^2$  to the critical value from the Chi-squared distribution with  $p$  degrees of freedom.

BP 2 The Breusch-Pagan F test is as follows: Suppose the auxiliary equation being tested is

$$\text{Aux: } \varepsilon^2 = \delta_0 + \delta_1 x_1 + \dots + \delta_k x_p + \varepsilon \quad (\text{p slope terms})$$

Using this equation calculate the following F:

$$F_c = (R^2 / p) / [(1-R^2) / (n - p - 1)]$$

If  $F_c > F_{p, n-p-1}$  reject the null hypothesis of homoscedasticity.

## Python Example:

```
from __future__ import print_function
import statsmodels import numpy as np
import pandas as pd
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
import matplotlib.pyplot as plt
# Load instructional data url =
'http://vincentarelbundock.github.io/Rdatasets/csv/HistData/
Gue rry.csv'
dat = pd.read_csv(url)
# Fit regression model (using the natural log of one of the
regressaors)
results = smf.ols('Lottery ~ Literacy + np.log(Pop1831)',
data=dat).fit()
print(results.summary())
```

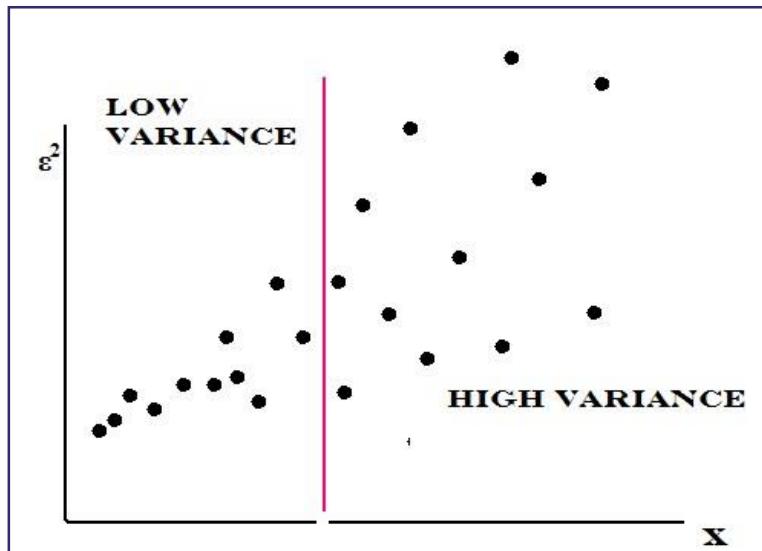
OLS Regression Results						
Dep. Variable:	Lottery	R-squared:	0.348			
Model:	OLS	Adj. R-squared:	0.333			
Method:	Least Squares	F-statistic:	22.20			
Date:	Tue, 11 Nov 2014	Prob (F-statistic):	1.90e-08			
Time:	08:44:53	Log-Likelihood:	-379.82			
No. Observations:	86	AIC:	765.6			
Df Residuals:	83	BIC:	773.0			
Df Model:	2					
coef	std err	t	P> t	[95.0% Conf. Int.]		
Intercept	246.4341	35.233	6.995	0.000	176.358	316.510
Literacy	-0.4889	0.128	-3.832	0.000	-0.743	-0.235
np.logPOP	<u>-31.3114</u>	5.977	-5.239	0.000	-43.199	-19.424
Omnibus:	3.713	Durbin-Watson:		019	2.	
Prob(Omnibus):	0.156	Jarque-Bera (JB):		3.394		
Skew:	-0.487	Prob(JB):		0.183		
Kurtosis:	3.003	Cond. No.		702		

#To get the Breusch Pagan LM and F input the following:

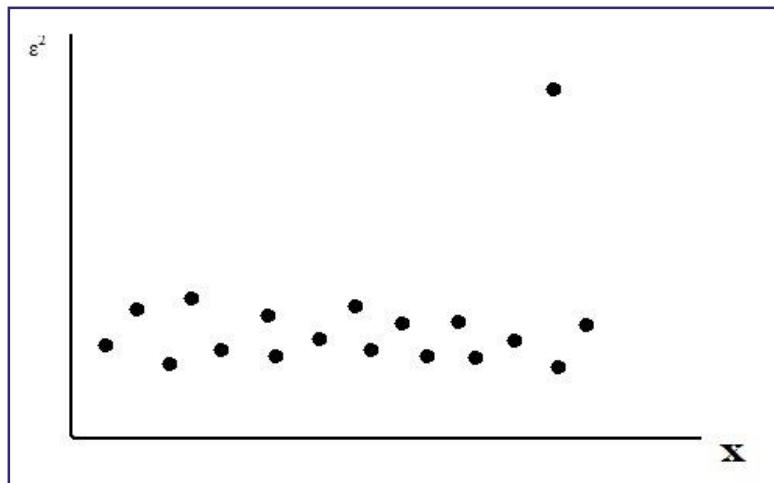
```
test = sms.het_breushpagan(results.resid, results.model.exog)
n
[10]: test
Out[10]:
(4.8932133740940529, #LM statistic
 0.086586905023517957, #LM p value
 2.5037159462564857, # F value
 0.087940287826726415) # F p value
```

## Goldfield Quandt Test

Graphing the squared errors against a suspected cause of the heteroscedasticity may reveal the nature of the heteroscedasticity. In the case below, the variance of the error terms increases as the exogenous variable,  $x$ , increases:



Graphical inspection may help to identify situations in which an outlier causes a test to show erroneously the presence of heterogeneity as in the graph below:



In this case the squares of the errors are fairly constant except for the one outlier.

The Goldfeld Quandt F statistic is calculated by:

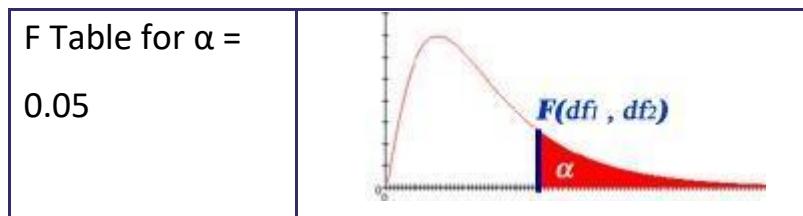
$$\text{Calculated } F: F^* = (\sum_{i=1, n_2} \varepsilon_i) / n_2 / (\sum_{i=1, n_1} \varepsilon_i) / n_1$$

Under the null hypothesis of homoscedasticity is distributed as  $\sim F_{n_2, n_1}$

**Note:** if the numerator is significantly different from the denominator, the calculated F will be significantly different from one therefore leading us to reject homoscedasticity.)

If  $F^* >$  critical value of  $F_{n_2, n_1}$  ( 5% level in the table below)

Use an F table in calculating the Goldfeld Quandt test as below:



For example, if  $n_1=n_2=9$  the critical value is 3.1789. If the calculated F is 99 as in the example below, we reject homoscedasticity.

/	$df_1=1$	2	3	4	5	6	7	8	9
$df_2=1$	161.44	199.50	215.7	224.5832	230.1619	233.9860	236.7684	238.8827	240.5433
2	18.5128	19.0000	19.1643	19.2468	19.2964	19.3295	19.3532	19.3710	19.3848
3	10.1280	9.5521	9.2766	9.1172	9.0135	8.9406	8.8867	8.8452	8.8123
4	7.7086	6.9443	6.5914	6.3882	6.2561	6.1631	6.0942	6.0410	5.9988
5	6.6079	5.7861	5.4095	5.1922	5.0503	4.9503	4.8759	4.8183	4.7725
6	5.9874	5.1433	4.7571	4.5337	4.3874	4.2839	4.2067	4.1468	4.0990
7	5.5914	4.7374	4.3468	4.1203	3.9715	3.8660	3.7870	3.7257	3.6767
8	5.3177	4.4590	4.0662	3.8379	3.6875	3.5806	3.5005	3.4381	3.3881
9	5.1174	4.2565	3.8625	3.6331	3.4817	3.3738	3.2927	3.2296	3.1789
10	4.9646	4.1028	3.7083	3.4780	3.3258	3.2172	3.1355	3.0717	3.0204

## Goldfeld Quandt in Python

```
from __future__ import print_function
import statsmodels import numpy as np
import pandas as pd
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
import matplotlib.pyplot as plt
```

```
rl = 'c:/users/gib/documents/d5.hetero.csv' # this imports
data set hetero with y,x,z
dat = pd.read_csv(url)
results = smf.ols('y ~ x + z', data=dat).fit()
In [9]: print(results.summary())
```

## OLS Regression Results

OLS Regression Results					
Dep. Variable:	y	R-squared:	0.987		
Model:	OLS	Adj. R-squared:	0.986		
Method:	Least Squares	F-statistic:	625.1		
Date:	Wed, 12 Nov 2014	Prob (F-statistic):	6.50e-16		
Time:	15:09:56	Log-Likelihood:	-20.824		
No. Observations:	19	AIC:	47.65		
Df Residuals:	16	BIC:	50.48		
Df Model:	2				
	coef	std err	t	P> t	[95.0% Conf. Int.]
Intercept	6.1365	10.031	0.612	0.549	-15.129 27.402
x	0.3912	0.050	7.806	0.000	0.285 -0.497
z	-0.3145	0.542	-0.580	0.570	-1.464 0.835
Omnibus:	7.704			Durbin-Watson:	0.627
Prob(Omnibus):	0.021			Jarque-Bera (JB):	5.430
Skew:	1.277			Prob(JB):	0.0662
Kurtosis:	3.576			Cond. No.	1.68e+03

```
# in this example the middle 3 observations are dropped test =
sms.het_goldfeldquandt(results.resid, results.model.exog,drop=3)
```

In [20]: test Out[20]: (F= 99.20209368232976,

Prob=0.00026499422141413004, 'increasing')

The very high F indicates the probability of the null hypothesis of homoscedasticity is very low.

## Park Test

The Park test assumes a specific model of heteroscedasticity. Specifically, it assumes that the heteroscedasticity is proportional to some exponent of an independent variable ( $X$ ) in the model. This assumption can be expressed as

$$\text{The form: } \sigma^2_i = \sigma X^\alpha$$

$$\text{In log form: } \ln \sigma^2_i = \ln \sigma + \alpha \ln X$$

As in the other cases, the squared of the regression residuals functions as a proxy for  $\sigma^2_i$ . Estimate the auxiliary equation:

$$\ln \varepsilon^2_i = \ln \sigma + \alpha \ln X$$

$$\text{estimating eq: } \ln \varepsilon^2_i = \delta_0 + \delta_1 \ln X + v_i \quad (v \text{ is iid (0, constant variance)})$$

If the coefficient for the  $X$  term  $\delta_1$  is significant, we reject homoscedasticity.

## Park Test in Python

```

import statsmodels
import numpy as np
import pandas as pd
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
import matplotlib.pyplot as plt
url = 'c:/users/gib/documents/d5.hetero.csv'
dat = pd.read_csv(url)
results = smf.ols('y ~ x + z', data=dat).fit()      # produces
original equation regression results
res=results.resid                                     # gets
residuals from the original regression model
ressq= res*res
# square of residuals
logerror=np.log(ressq)
# produces log of squared residuals
X = dat['x']
# retrieve the X variable suspected for heteroscedasticity
logX=np.log(X)
# take natural log of X variable
results = smf.ols('logerror ~ X', data=dat).fit()    #
auxiliary regression
print(results.summary())
# print summary of auxiliary regression

```

OLS Regression Results						
Dep. Variable:	logerror	R-squared:	0.439			
Model:	OLS	Adj. R-squared:	0.406			
Method:	Least Squares	F-statistic:	13.29			
Date:	Thu, 13 Nov 2014	Prob (F-statistic):	0.00200			
Time:	15:42:32	Log-Likelihood:	-32.880			
No. Observations:	19	AIC:	69.76			
Df Residuals:	17	BIC:	71.65			
Df Model:	1					
coef	std err	t	P> t	[95.0% Conf. Int.]		
Intercept	-3.7502	0.586	-6.398	0.000	-4.987	-2.514
X	0.0791	0.022	3.646	0.002	0.033	0.125
Omnibus:	0.431	Durbin-Watson:		1.117		
Prob(Omnibus):	0.806	Jarque-Bera (JB):		0.132		
Skew:	0.197	Prob(JB):		0.936		
Kurtosis:	2.895	Cond. No.		47.8		

Since the coefficient for the variable X is significant, we reject the null hypothesis of homoscedasticity

## Glejser Test

When we use the Glejser, we're looking for a scaling effect

The procedure:

- Perform the original regression (it can also be multivariate) analysis and save the residuals.
- Regress the absolute value of the residuals (errors) on possible sources of heteroscedasticity
- A significant coefficient indicates heteroscedasticity
- Regress  $|\varepsilon_i|$  against independent variable(s)
- You can run different kinds of regressions:

$$|\varepsilon_i| = \delta_0 + \delta_1 X$$

(model a)  $|\varepsilon_i| = \delta_0 + \delta_1 (1/X)$

(model b)

$$|\varepsilon_i| = \delta_0 + \delta_1 vX \quad (\text{model c})$$

```

import statsmodels
import numpy as np
import pandas as pd
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
import matplotlib.pyplot as plt
url = 'c:/users/gib/documents/d5.hetero.csv'
dat = pd.read_csv(url)
results = smf.ols('y ~ x + z', data=dat).fit()      # produces
original equation regression results

res=results.resid                                     # gets
residuals from the original regression model
absol=np.abs(res)
#absolute values of the residuals
#Example: Model a
results_a = smf.ols('absol ~ x', data=dat).fit()
print(results_a.summary())

```

### OLS Regression Results

Dep. Variable:	absol	R-squared:	0.291			
Model:	OLS	Adj. R-squared:	0.249			
Method:	Least Squares	F-statistic:	6.967			
Date:	Thu, 13 Nov 2014	Prob (F-statistic):	0.0172			
Time:	18:11:20	Log-Likelihood:	-9.7137			
No. Observations:	19	AIC:	23.43			
Df Residuals:	17	BIC:	25.32			
Df Model:	1					
coef	std err	t	P> t	[95.0% Conf. Int.]		
Intercept	0.1657	0.173	0.957	0.352	0.200	-0.531
x	0.0169	0.006	2.640	0.017	0.003	0.030
Omnibus:	9.635	Durbin-Watson:	0.965			
Prob(Omnibus):	0.008	Jarque-Bera (JB):	6.776			
Skew:	1.278	Prob(JB):	0.0338			
Kurtosis:	4.422	Cond. No.	47.8			

□

```
# example model b
x_invert=1/X

results_b = smf.ols('absol ~ x_invert', data=dat).fit()

print(results_b.summary())
```

OLS Regression Results						
Dep. Variable:	absol	R-squared:	0.364			
Model:	OLS	Adj. R-squared:	0.327			
Method:	Least Squares	F-statistic:	9.732			
Date:	Thu, 13 Nov 2014	Prob (F-statistic):	0.00624			
Time:	18:52:11	Log-Likelihood:	-8.6768			
No. Observations:	19	AIC:	21.35			
Df Residuals:	17	BIC:	23.24			
Df Model:	1					
coef	std err	t	P> t	[95.0% Conf. Int.]		
Intercept	0.9266	0.154	6.017	0.000	0.602	1.251
x_invert	-4.8469	1.554	-3.120	0.006	-8.125	-1.569
Omnibus:	6.553		Durbin-Watson:	1.038		
Prob(Omnibus):	0.038		Jarque-Bera (JB):	4.052		
Skew:	1.050		Prob(JB):	0.132		
Kurtosis:	3.841		Cond. No.	16.9		

## Solutions: GLS / WLS

In a perfect world, we would actually know what heteroscedasticity we could expect—and we would then use ‘weighted least squares’.

WLS essentially transforms the entire equation by dividing through every part of the equation with the square root of whatever it is that one thinks the variance is related to.

In other words, if one thinks one's variance of the error terms is related to  $X_1^2$ , then one divides through every element of the equation (intercept, each  $b_x$ , residual) by  $X_1$ .

In this way, one creates a transformed equation, where the variance of the error term is now constant (because you've "weighted" it appropriately).

Note, however, that since the equation has been "transformed", the parameter estimates are different than in the non-transformed version—in the example above, for  $b_2$ , you have the effect of  $X_2/X_1$  on  $Y$ , not the effect of  $X_2$  on  $Y$ . So, you need to think about that when you are interpreting your results.

We almost never know the precise form that we expect heteroscedasticity to take.

So, in general, we ask the software package to give us White's Heteroskedastic-Constant

Variances and Standard Errors (White's robust standard errors). (alternatively, less commonly, Newey-West is similar.)

(For those of you who have dealt with clustering—the basic idea here is somewhat similar, except that in clustering, you identify an X that you believe your data are “clustered on”. When I have repeated states in a database—that is, multiple cases from California, etc.—I might want to cluster on state (or, if I have repeated legislators, I could cluster on legislator. Etc.) In general, it’s a recognition that the error terms will be related to those repeated observations—the goodness of fit within the observations from California will be better than the goodness of fit across the observations from all states.)

### **WLS Assignment Exercise**

Variance Proportional to Square of Mean problem. In this test for heteroscedasticity:

1. Regress the square of model errors
  - $e_i^{*2} = (y_i - \alpha^* - \beta^* X_i)^2$  [the \* indicates the estimate from OLS]
  - on a constant and predicted y's squared in the OLS model or
  - $(y_i^*)^2 = (\alpha^* + \beta^* X_i)^2$
2. Do a F test and t test for this regression. If significant reject homoscedasticity.
3. Use the  $(y_i^*)^2$  as the diagonal elements in the W matrix of the WLS

# Multicollinearity

## Multicollinearity

Multicollinearity is a condition where two or more independent variables are strongly correlated with each other. In an extreme case called perfect multicollinearity one may be a multiple of the other  $z=3*x$  or variable z is exactly 3 times variable x. OLS cannot estimate both z and x.

- Problem:  $x_1 = 2 + 3*x_2$
- Does perfect multicollinearity exist?

Problem:

You have two measures of gold produced in Chile over a span of years. In one-year data assert that 1,000 kg were produced. Another data source states that for the same year 2,204.62 pounds were produced. (Did you know there are 2.20462 pounds in a kilogram?)

It can occur because the variables may measure the same concepts

Only existence of multicollinearity is not a violation of the OLS assumption.

Symptoms of multicollinearity may be observed in situations:

- Small changes in the data cause wide swings in the parameter estimates.  
Big problem!
- Coefficients may have very high standard errors and low significance levels

Coefficients may have the “wrong” sign or implausible magnitude

**Where does it occur in finance?** You are trying to estimate the amount of company investment. You have two variables  $x$ = income of the company and  $z$ =corporate income tax paid by the company. Are they likely to be multicollinear? What about the current and quick ratios of firms?

## Detecting Multicollinearity

1. Check for the correlation between the predictor variables. If it is high this may be a MC warning
2. Construct the VIF (variance inflation factor) by regressing a predictor variable against all the other predictor variables you are using. If it is greater than 5 this may be another sign of MC

## Correcting

1. Remove one of the offending variables if the other can stand in for it. You have to do this if there exists perfect multicollinearity.
2. Standardize the predictor variables

## Detecting MC

### Variance Inflation Factor (VIF)

To compute the VIF, the auxiliary regressions of each independent variable on all the other K- independent variables (regressors) are performed. For the variable i, the R-squared is  $R_i^2$ . Then the VIF for independent variable j is defined as:

$$VIF_i = 1 / (1 - R_i^2)$$

For example the regression  $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$  Regressions are:

$$\begin{aligned} x_1 &= y_1 + y_{12} x_2 + \\ y_{13} x_3 &\quad x_2 = y_2 + \\ y_{21} x_1 + y_{23} x_3 &\quad x_3 \\ &= y_3 + y_{31} x_1 + y_{32} \\ &\quad x_2 \end{aligned}$$

So for example if the  $R^2 = .2$  the  $VIF = 1/(1-.2) = 1/.8 = 1.25$ . In contrast if the Pearson product moment correlation coefficient is high say .95 the VIF is:

$$1/(1-.9) = 1/(.05) = 20$$

## Diagnostics:

As a rule of thumb, collinearity is potentially a problem for values of VIF>10 (sometimes VIF>5).

The average VIF is the average  $VIF_i$  across the K independent variables. If the VIFs are .2,.4, .1, and .8 The average VIF is:

$(.2+.4+.1+.8) / 4 = .375$ . Rule of thumb: an average VIF >1 indicates multicollinearity.

Square root of the VIF how much larger the standard error is, compared with what it would be if that variable was uncorrelated with the other predictor variables. For example, if  $VIF_i = 6$  then  $\sqrt{VIF_i} = 2.446$ . So the standard error for the coefficient of that predictor variable is 2.446 times as large as it would be if that (ith) predictor variable were uncorrelated with the other predictor variables.

- Tolerance: Tolerance (  $\beta_i$  ) =  $1/VIF_i = 1-R^2_i$

If a variable has a VIF=10 then the (low) tolerance = $1/10 = .1$  indicating MC whereas VIF=1 indicates a high tolerance of  $1/1=1$ .

## Condition Number (Reported by Python)

The condition number (often called kappa) of a data matrix is  $\kappa(X)$  and measures the sensitivity of the parameter estimates to small changes in the data matrix (Belsley, Kuh and Welsh 1980, Belsley 1982). It is calculated by taking the ratio of the largest to the smallest singular values from the singular value decomposition of X. Kappa is

$$\text{Sqrt}(\text{maxeigenvalue}(X^T X) / \text{mineigenvalue}(X^T X))$$

A condition number above 30 is considered to be an indication of multicollinearity but some say as low as 15 (again rules of thumb).

# Structural Breaks

## Structural Breaks

“A structural break (change) is a sudden event that changes the structure of the econometric model” (Kapetanios and Tzavalis, 2004).

Factors that generate structural breaks:

6. Change in political regime
7. Change in corporate leadership (CEO)
8. Change in the economic environment (changes of the monetary policy, fiscal policy or the imposition of an embargo)
9. Any other structural event

There are the following types of breaks:

- 1) Breaks in Mean
- 2) Breaks in Variance
- 3) Breaks in Relationships
- 4) Single Breaks
- 5) Multiple Breaks
- 6) Continuous Breaks

Taking into consideration structural breaks are very important as they can generate forecasting errors and unreliable results. There is no good theory about how to forecast in the presence of breaks. One possible solution was provided by Pesaran and Timmermann (2007). According to their research, in a regression with a single break, the optimal window for estimation includes all of the observations after the break and some of the observations before the break. More observations decrease the variance but bring some bias. One simple rule for dealing with breaks is to split the sample at the estimated break.

## Chow Test

Chow test is used for checking the existence of a structural break. The test assumes the following steps:

1. Model data using the following regression:

$$y_t = a + bx_{1t} + cx_{2t} + \varepsilon$$

2. Split the data in two groups and then apply regression analysis for each group:

$$yt = a1 + b1x1t + c1x2t + \varepsilon$$

$$yt = a2 + b2x1t + c2x2t + \varepsilon$$

Null hypothesis:  $a_1 = a_2$ ,  $b_1 = b_2$  and  $c_1 = c_2$

Alternative hypothesis: coefficients are different

$S_C$  – sum of squared residuals from the combined data

$S_1$  – the sum of squared residuals from the first group

$S_2$  – the sum of squared residuals from the second group

$N_1$  and  $N_2$ - the number of  
observations in each group  $k$  – the  
total number of parameters

The Chow test statistic is:

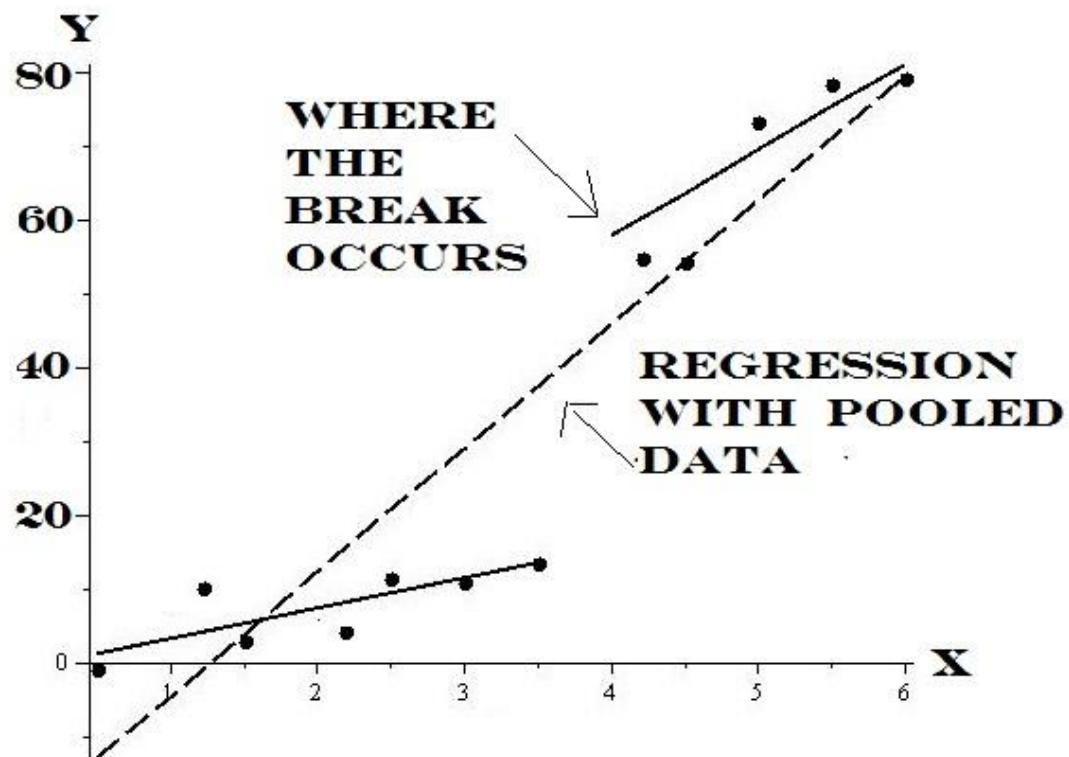
$$\frac{(S_C - (S_1 + S_2))/k}{(S_1 + S_2)/(N_1 + N_2 - 2k)}$$

The test statistic follows an F distribution with  $k$  and  $N_1 - N_2 - 2k$  degrees of freedom.

The following graph illustrates the basic idea about structural breaks.

The dotted line – regression line considering the entire period

The two small lines – two regression lines considering that the period was divided in two groups. The three regression lines have different slopes.



**Example 1:**

	GDP in billions of chained 2009 dollars
1. 1984	2. 7,285.0
3. 1985	4. 7,593.8
5. 1986	6. 7,860.5
7. 1987	8. 8,132.6
9. 1988	10.8,474.5
11.1989	12.8,786.4
13.1990	14.8,955.0
15.1991	16.8,948.4
17.1992	18.9,266.6
19.1993	20.9,521.0
21.1994	22.9,905.4
23.1995	24.10,174.8
25.1996	26.10,561.0
27.1997	28.11,034.9
29.1998	30.11,525.9
31.1999	32.12,065.9
33.2000	34.12,559.7
35.2001	36.12,682.2
37.2002	38.12,908.8
39.2003	40.13,271.1
41.2004	42.13,773.5
43.2005	44.14,234.2
45.2006	46.14,613.8

	GDP in billions of chained 2009 dollars
47.2007	48.14,873.7
49.2008	50.14,830.4
51.2009	52.14,418.7
53.2010	54.14,783.8
55.2011	56.15,020.6
57.2012	58.15,354.6
59.2013	60.15,583.3
61.2014	62.15,961.7
63.2015	64.16,345.0

Source: <http://www.bea.gov/national/index.htm#gdp> We assume that there was a structural break in U.S. GDP in 2009. We divide GDP data in two: 1984 – 2008 and 2009 – 2015. We can implement now two regressions considering the two periods (the first one does not include the structural break while the second one includes the structural break).

## Example 2:

Fuentes, Gredig and Larraín (2007) and Magud and Medina (2011) provide another solution to deal with structural breaks. Magud and Medina estimate the potential output for the natural resource and the non-natural resource sectors in Chile using a SVAR with long-run restrictions. They find a structural break in 1998 due to the Asian crisis, which diminished the potential output in Chile to 3-4 % from 7 % in the previous period. Their methodology is the following:

1. Identify the structural break (Chow test)
2. Divide the period considered in the analysis in two separate sub-periods

Sub-period 1

$$[\nu_1, \nu_2, \dots, \nu_n]$$

$\nu_n$  corresponds to the structural break period

Sub-period 2

$$[\nu_{n+1}, \nu_{n+2}, \dots, \nu_{n+m}]$$

3. Calculate mean values for the two separate sub-periods
4. De-mean the time series

De-means variables for sub-period 1 = variables – mean value for sub-period 1

De-means variables for sub-period 2 = variables – mean value for sub-period 2

5. Aggregate the two sub-periods containing demeaned values
6. Perform econometric analysis

# Outliers

## Outliers

- An outlier is an extreme observation. Typically points further than three or four standard deviations from the mean are considered as outliers.
- Possible causes:
  - An event not known to the researcher such as a secret nuclear accident causing population in an area under study to have shrank. (Think of this as an omitted variable.)
  - A spike in crime rate caused by unusual event such as a riot or the presence of a party convention.
- Outliers will have more influence on the regression than other data points.

## Outliers and Quantitative Finance

- Quantitative analysts have to take into consideration the following:
- Outliers significantly influence OLS estimates and can determine unreliable information
- Outliers are identified among equity and market returns
- Outliers occur because a stock makes a sudden and unusual move
- Outliers are generally present at companies with small capitalizations compared to companies with large capitalizations

- One solution is to eliminate outliers from the time series and then perform regression analysis. Outliers are considered a benchmark value from the mean or variance. All values above that benchmark should be eliminated according to this approach

## Cook's Distance

- Outliers are usually identified visually via a graph
- Cook's D can be used to identify possibly influential outliers, known as Cook's Distance

$$D_i = (\sum_{j=1, n} [y^*j - y^*j(i)]^2) / (k+1) \text{ MSE}$$

$y^*j(i)$  is the prediction of  $y_j$  when the outlier is removed.

K is the number of regressors

MSE is the mean squared error

If Cook's distance is higher than 1 the observation may be considered as influential. Some analysts set a threshold of  $D \geq 4/N$  or  $D \geq 4/(n-[k+1])$

(These are Rules of Thumb and often tend to identify too many points as potential outliers).

Code for D:

```
def CookDistance(X,Y, hii=None,resids=None,MRSS = None):
n = len(X)
p = len(X[0])
fit = None
if resids is None:
fit = ols(X,Y)
ei = fit.resids
else:
ei = resids
if hii is None:
hii = diaghmatix(X)
if MRSS is None:
if fit is None:
fit = ols(X,Y)
MRSS = fit.MRSS
Di = [0] * n
invden = 1.0 / (p * MRSS)
for i in range(n):
Di[i] = ei[i]**2 * invden * hii[i]/ (1 - hii[i])**2
return Di
```

## Treatment of Outliers

- Identification of an outlier does not suggest the point(s) should be removed. Such action may result in the loss of important information.
- The identification of an outlier may suggest examination of data to ascertain no measurement error (off by a decimal point or transposing of numbers [19 to 91]).

Hence the first question that should be asked is whether there exists some substantive information about these points that suggests that they should be removed.

- Do they involve special properties or circumstances not relevant for the situation under investigation?
- Do they involve possible measurement errors?

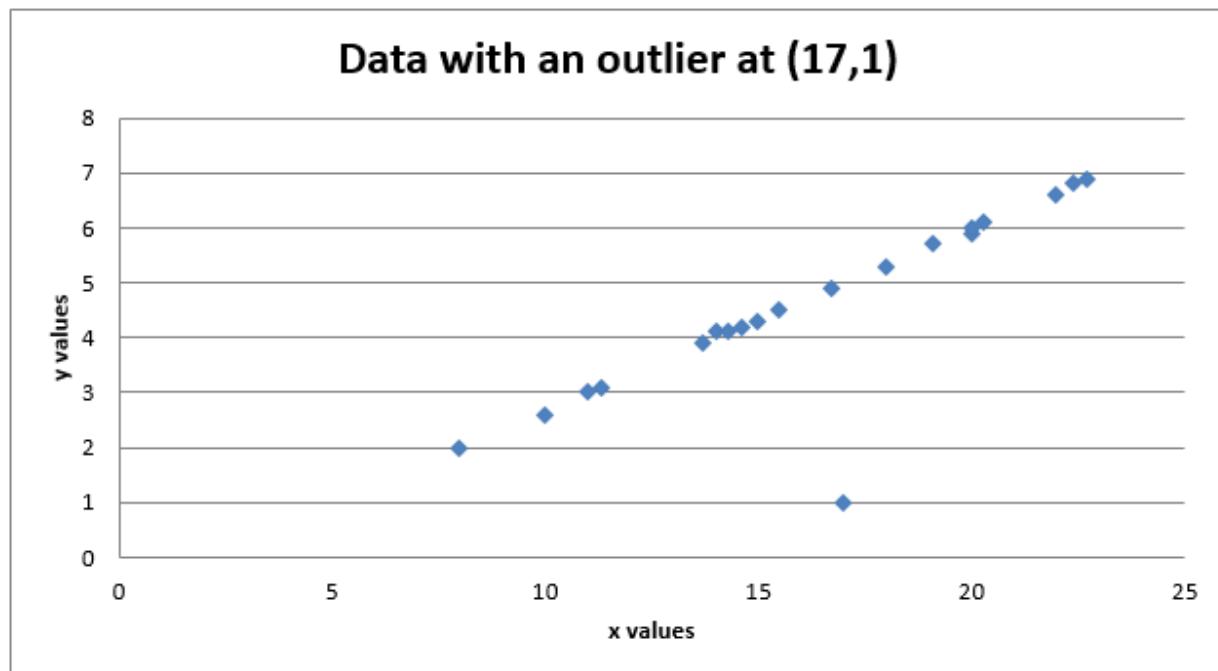
If no such distinguishing features can be found, then there are no clear grounds for eliminating outliers.

An alternative approach is to perform the regression both with and without these outliers, and examine their specific influence on the results. If this influence is minor, then it may not matter whether or not they are omitted. On the other hand, if their influence is substantial, then it is probably best to present the results of both analyses, and simply alert the reader to the fact that these points may be questionable.

## Dummy Variable Technique

- For a particular observation(s) that is identified as an outlier create a dummy for that (those) observations.
- D=0 if the observation is not an outlier
- D=1 if the observation is an outlier
- Example
- The x y data are used in the regression below:
- y 2 3 2.6 3.1 4.2 4.1 3.9 4.3 4.5 4.1 4.9 1 5.3 5.7 5.9 6 6.1 6.8 6.6 6.9
- And x data:
- x 8 11 10 11.3 14.6 14 13.7 15 15.5 14.3 16.7 17 18 19.1 20 20 20.3  
22.4 22 22.7

Which is graphed below:



## Regress y on x without Removing Outlier x=17 and y=1

- Import NumPy as np
- Import pandas as pd
- url = 'c:/users/gib/documents/d5.temp2.csv'
- dat = pd.read\_csv(url)
- x = dat['x']
- DUM = dat['DUM'] # not used in first reg
- y = dat['y']
- import statsmodels.formula.api as smf
- results = smf.ols('y ~ x ', data=dat).fit()
- print results.summary()

## Regression Full Data no Dummy

```
results = smf.ols('y ~ x', data=dat).fit()
In [49]: print results.summary()
```

OLS Regression Results						
Dep. Variable:	y	R-squared:	0.707			
Model:	OLS	Adj. R-squared:	0.691			
Method:	Least Squares	F-statistic:	43.39			
Date:	Sun, 02 Nov 2014	Prob (F-statistic):	3.46e-06			
Time:	12:09:19	Log-Likelihood:	-25.578			
No. Observations:	20	AIC:	55.16			
Df Residuals:	18	BIC:	57.15			
Df Model:	1					
coef	std err	t	P> t	[95.0% Conf. Int.]		
Intercept	-0.7151	0.825	-0.867	0.398	2.449	1.018
x	0.3234	0.049	6.587	0.000	0.220	0.427
Omnibus:	51.396			Durbin-Watson:	2.108	
Prob(Omnibus):	0.000			Jarque-Bera (JB):	241.965	
Skew:	-4.106			Prob(JB):	2.87e-53	
Kurtosis:	17.931			Cond. No.	67.9	

## Regression with Dummy for Outlier

OLS Regression Results						
Dep. Variable:	y	R-squared:	0.999			
Model:	OLS	Adj. R-squared:	0.999			
Method:	Least Squares	F-statistic:	1.563e+04			
Date:	Sun, 02 Nov 2014	Prob (F-statistic):	1.78e-28			
Time:	11:50:02	Log-Likelihood:	37.326			
No. Observations:	20	AIC:	-68.65			
Df Residuals:	17	BIC:	-65.67			
Df Model:	2					
	coef	std err	t	P> t	[95.0% Conf. Int.]	
Intercept	-0.6499	0.037	--17.773	0.000	-0.727	-0.573
x	0.3317	0.002	152.344	0.000	0.327	0.336
DUM	-3.9882	0.042	-95.669	0.000	-4.076	-3.900
Omnibus:	5.418			Durbin-Watson:	2.217	
Prob(Omnibus):	0.067			Jarque-Bera (JB):	4.152	
Skew:	0.246			Prob(JB):	0.125	
Kurtosis:	5.177			Cond. No.	77.3	

### Comment on the Dummy

- The dummy is highly significant with a t value of -95.669
- The regression without the dummy has a much lower R squared
- The intercept and x have a much lower t values than when the dummy is included

## Regression Excluding Outlier

OLS Regression Results						
Dep. Variable:	y	R-squared:	0.999			
Model:	OLS	Adj. R-squared:	0.999			
Method:	Least Squares	F-statistic:	2.321e+04			
Date:	Sun, 02 Nov 2014	Prob (F-statistic):	4.25e-28			
Time:	12:15:25	Log-Likelihood:	34.973			
No. Observations:	19	AIC:	-65.95			
Df Residuals:	17	BIC:	-64.06			
Df Model:	1					
	coef	std err	t	P> t	[95.0% Conf. Int.]	
Intercept	-0.6499	0.037	--17.773	0.000	-0.727	-0.573
x	0.3317	0.002	152.344	0.000	0.327	0.336
Omnibus:	4.807			Durbin-Watson:	2.233	
Prob(Omnibus):	0.090			Jarque-Bera (JB):	3.096	
Skew:	0.240			Prob(JB):	0.213	
Kurtosis:	4.918			Cond. No.	66.2	

## Influence Tests in Python

Influence tests

Once created, an object of class OLSInfluence holds attributes and methods that allow users to assess the influence of each observation. For example, we can compute and extract the first few rows of DFbetas by:

In [4]:

```
from statsmodels.stats.outliers_influence import OLSInfluence test_class =  
OLSInfluence(results) test_class.dfbetas[:5,:]
```

Out[4]:

```
array([[-0.00301154, 0.00290872, 0.00118179], [-0.06425662, 0.04043093,  
0.06281609], [ 0.01554894, -0.03556038, -0.00905336], [ 0.17899858,  
0.04098207, -0.18062352], [ 0.29679073, 0.21249207, -0.3213655 ]])
```

Explore other options by typing dir(influence\_test)

Useful information on leverage can also be plotted:

In [5]:

```
from statsmodels.graphics.regressionplots import plot_leverage_resid2 fig, ax  
= plt.subplots(figsize=(8,6)) fig = plot_leverage_resid2(results, ax = ax)
```

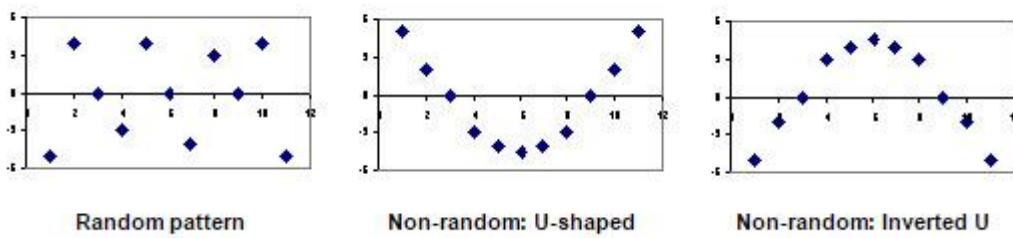
# Residuals and Predicted Values from a Linear Regression

## Residuals and Predicted Values

It is frequently the case that we use residuals and predicted values in data analysis. The analysis of regression residuals may give us information regarding the accuracy of predictions or the type of model used. If the regression residuals are normally distributed then the prediction intervals are considered to be accurate. If the regression residuals are not normal, then the prediction intervals are inaccurate.

If the residuals have a random pattern -> a linear regression is a good fit

If residuals are non-random and have U-shape or an inverted U shape -> the variables can be modelled better using a non-linear model



## Python Code:

```
import numpy as np
import statsmodels.api as sm
y=[3.4, 3.0, 2.6, 2.8, 2.8, 3.2, 3.5, 3.4, 3.0, 2.0, 0.8, -0.5, -1.6, -2.0, -2.0, -1.4, -0.3, 0.8, 1.6, 1.6, 1.4, 1.0, 1.1, 1.8, 2.2, 2.6, 2.5, 2.4, 2.2, 2.1, 2.4, 2.2, 2.5, 2.8, 3.0, 3.3, 3.2] # New Zealand's economic growth
x=[12.3, 10.4, 9.9, 9.6, 11.2, 13.4, 11.4, 7.7, 2.7, -4.5, -6.7, -9.0, -9.1, -3.1, 1.1, 5.4, 6.3, 3.1, 0.2, -1.6, -1.2, 0.8, 2.6, 2.8, 3.2, 4.0, 4.8, 6.9, 7.7, 9.0, 10.2, 9.2, 7.9, 7.1, 4.8, 6.3, 8.4] # house price evolution in New Zealand

x=sm.add_constant(x)           #add constant in the model
results=sm.OLS(y,x).fit()      #implement regression
estimated by OLS method
results.params
residuals=sm.OLS(y,x).fit().resid   #obtain residuals
residuals
y_pr=results.predict()          #obtain predicted values
y_pr
```

# Autoregressive (AR) Process

## The AR(1) Process

The Auto Regressive Model of order 1 is a stochastic process in discrete time and usually equally spaced. We call this AR(1). A stochastic process used in statistical calculations in which future values are estimated based on a weighted sum of past values. An autoregressive process operates under the premise that past values have an effect on current values. A process considered AR(1) is the first order process, meaning that the current value is based on the immediately preceding value. An AR(2) process has the current value based on the previous two values.

Autoregressive processes are used by investors in technical analysis. Trends, moving averages and regressions take into account past prices in an effort to create forecasts of future price movement. One drawback to this type of analysis is the past prices won't always be the best predictor of future movements, especially if the underlying fundamentals of a company have changed.

$$\begin{aligned}
 (1 - \phi_1 B)W_t &= e_t \\
 W_t - \phi_1 W_{t-1} &= e_t \\
 W_t - e_t &= \phi_1 W_{t-1} \\
 F_t = \phi_1 W_{t-1} &\quad (1)
 \end{aligned}$$

Where

$W_t$  - stationary time series  
 $e_t$  – is a white noise error term  
 $F_t$  – forecasting function

Now we derive the theoretical pattern of the ACF of an AR(1) process for identification purposes.

First, we note that (1) may be alternatively written in the form:

$$\begin{aligned} (1 - \phi B)W_t &= e_t \\ W_t &= (1 - \phi B)^{-1}e_t \\ W_t &= e_t + \phi e_{t-1} + \phi^2 e_{t-2} + \phi^3 e_{t-3} + \dots \end{aligned} \quad (2)$$

Second, we multiply the AR(1) process in (1) by  $W_{t-k}$  in expectations form

$$\begin{aligned} W_t W_{t-k} - \phi W_{t-1} W_{t-k} &= e_t W_{t-k} \\ E(W_t W_{t-k}) - \phi E(W_{t-1} W_{t-k}) &= E(e_t W_{t-k}) \\ \gamma_k - \phi \gamma_{k-1} &= (e_t W_{t-k}) \end{aligned} \quad (3)$$

Since we know that for  $k = 0$  the RHS of eq. (3) may be rewritten as

$$\begin{aligned} (e_t W_t) &= E[e_t(e_t + \phi e_{t-1} + \phi^2 e_{t-2} + \phi^3 e_{t-3} + \dots)] \\ (e_t W_t) &= E(e_t^2) = \sigma_{e_t}^2 \end{aligned} \quad (4)$$

and that for  $k > 0$  the RHS of eq. (3) is

$$(e_t W_{t-k}) = E[e_t(e_{t-k} + \phi e_{t-k-1} + \phi^2 e_{t-k-2} + \phi^3 e_{t-k-3} + \dots)] (e_t W_{t-k}) = 0$$

We may write the LHS of (3) as

$$\gamma_k - \phi\gamma_{k-1} = \begin{cases} = \gamma_0 - \phi\gamma_1 = \sigma_{et}^2 \text{ for } k = 0 \\ = 0 \text{ for } k \in N \end{cases} \quad (5)$$

From (5) we deduce

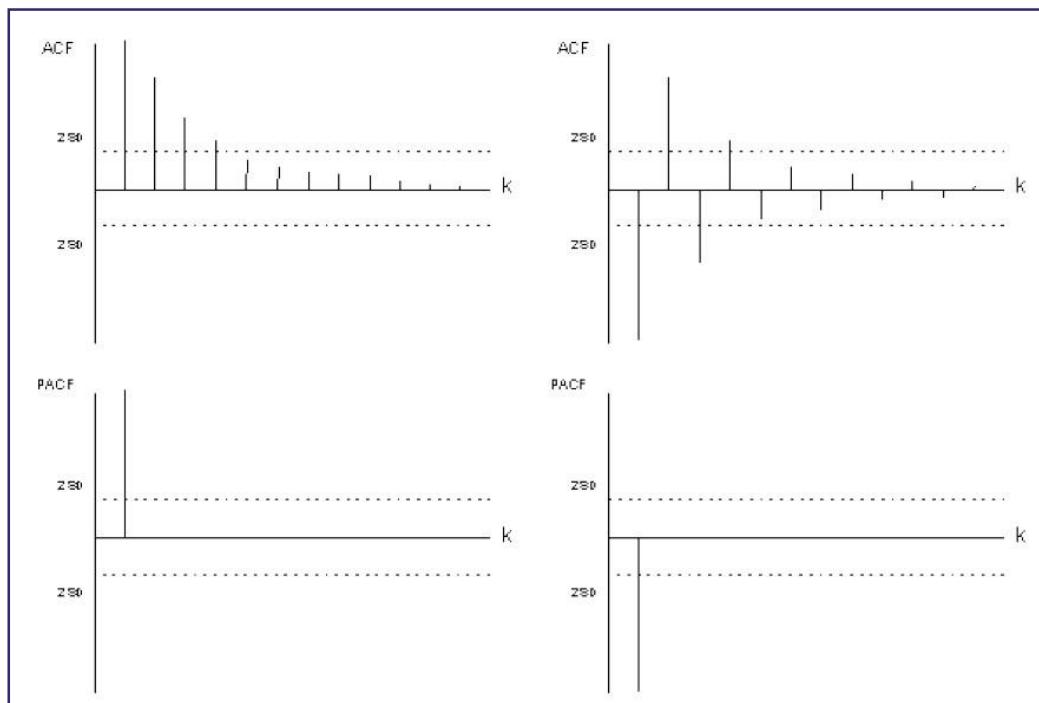
$$\gamma_0 - \phi\gamma_1 = \sigma_{et}^2$$

$$\gamma_0 - \phi^2\gamma_0 = \sigma_{et}^2$$

$$\gamma_0 = \frac{\sigma_{et}^2}{1-\phi^2} \quad (6)$$

and

$$\begin{aligned} \gamma_k - \phi\gamma_{k-1} &= 0 \\ \gamma_k &= \phi\gamma_{k-1} \\ \frac{\gamma_k}{\gamma_0} &= \frac{\phi\gamma_{k-1}}{\gamma_0} \\ \rho_k &= \phi\rho_{k-1} = \phi^2\rho_{k-2} = \phi^3\rho_{k-3} = \dots = \phi^k\rho_0 = \phi^k \end{aligned} \quad (7)$$



We can now easily observe how the theoretical ACF of an AR(1) process should look like. Note that we have already added the theoretical PACF of the AR(1) process since the first partial autocorrelation coefficient is exactly equivalent to the first autocorrelation coefficient.

**In general, a linear filter process is stationary if the  $\psi(\beta)$  polynomial converges**

Remark that the AR(1) process is stationary if the solution for  $(1 - \Phi\beta) = 0$  is larger in absolute value than 1 (the roots of  $\psi(\beta)$  are, in absolute value, less than 1).

This solution is  $\Phi^{-1}$ . Hence, if the absolute value of the AR(1) parameter is less than 1, then model is stationary which can be illustrated by the fact that

$$\psi(B) = (1 - \phi_1 B)^{-1} = \sum_{j=0}^{\infty} \phi_1^j B^j \quad (8)$$

For a general AR(p) model the solutions of

$$\phi(B) = \prod_{i=1}^p (1 - \xi_i B) \quad (9)$$

for which

$$\forall i[1,2, \dots, p]: |\xi_i| < 1 \quad (10)$$

must be satisfied in order to obtain stationarity.

**Exercise:**

Use the XLS.ar1 data to calculate all AR coefficients. Test for stationarity.

## Forecasting an AR(1)

The estimated AR(1) model for a series with T observations is say

$$y^*(t) = \alpha^* + \beta^* y(t-1)$$

To predict or forecast the value of y in the T+1 period we use

$$y^*(T+1) = \alpha^* + \beta^* y(T)$$

In the first instance we already know  $y(T)$ .

Example suppose  $\alpha^* = .2$   $\beta^* = .5$  and  $Y(t)=1.2$ :

$$y^*(t) = .2 + .5 y(t-1) \text{ Then}$$

$$y^*(T+1) = \alpha^* + \beta^*$$

$$y(T+1) = .2 + .5(1.2) = .8$$

## Desirable Forecasting

- Unbiased: We want  $E[y^*(t)] = y(t)$  or the expected value of the forecast error  $E(y(t) - y^*(t)) = E[u^*(t)] = 0$ .
- Models with small average errors are preferred.
- Models with small sums of squared errors are preferred.
- No pattern in  $u^*(t)$  series. A plot of the error terms is just random.

## Type of Forecast Errors

- Mean error

$\sum u^*(t) / T$  (where  $T$  is the number of periods forecast)

- Mean absolute (value) error:  $\sum$

$\sum |u^*(t)| / T$

- Mean squared error of forecast:

$\sum \{u^*(t)\}^2 / T$

## Theil's U

- Theil's  $U$  statistic is a relative accuracy measure that compares the forecasted results with the results of forecasting naively (based on last period). It also squares the deviations to give more weight to large errors:
- Theil's U Statistic

$$\sqrt{\frac{\sum_{t=1}^{n-1} \left( \frac{Y_{t+1}^* - Y_t}{Y_t} \right)^2}{\sum_{t=1}^{n-1} \left( \frac{Y_{t+1} - Y_t}{Y_t} \right)^2}}$$

where  $Y$  is the actual value of a point for a given time period  $t$ , (*for the forecast period*) and  $Y_t^*$  is the forecasted value.

Theil's  $U$  statistic is a relative accuracy measure that compares the forecasted results with the results of forecasting naively (based on last period). It also squares the deviations to give more weight to large errors.

## Theil's U Statistic Interpretation

- Less than 1 - The forecasting technique is better than guessing.
- Equal to 1 - The forecasting technique is about as good as guessing.
- More than 1 - The forecasting technique is worse than guessing.

## Forecasting with a Trend

- If the only independent variable used is time, forecasting is fairly easy.
- A start-up company has had sales in the past 12 months given by data

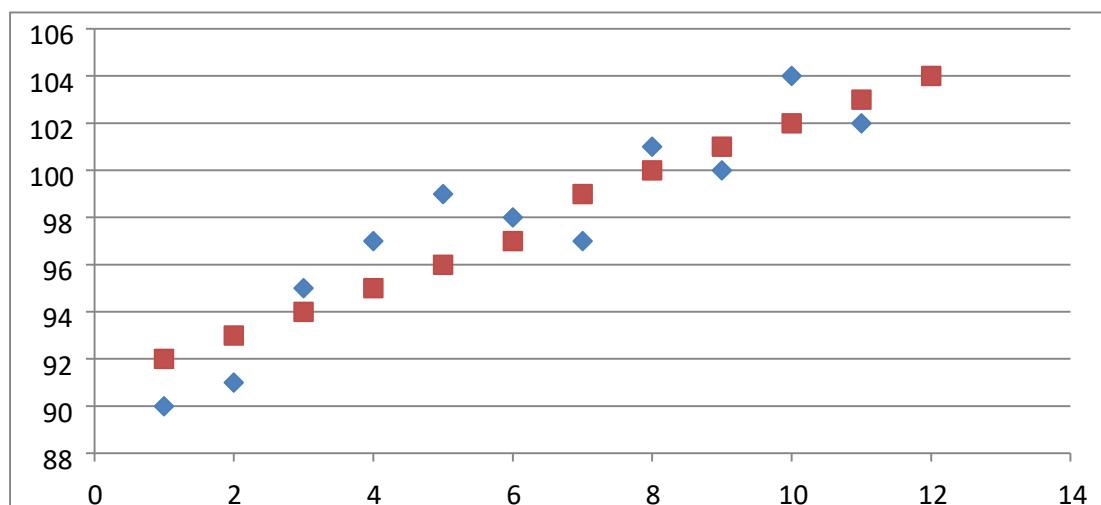
76 81 78 83 85 84 87

89 87 90 89 units. The trend line from this data is

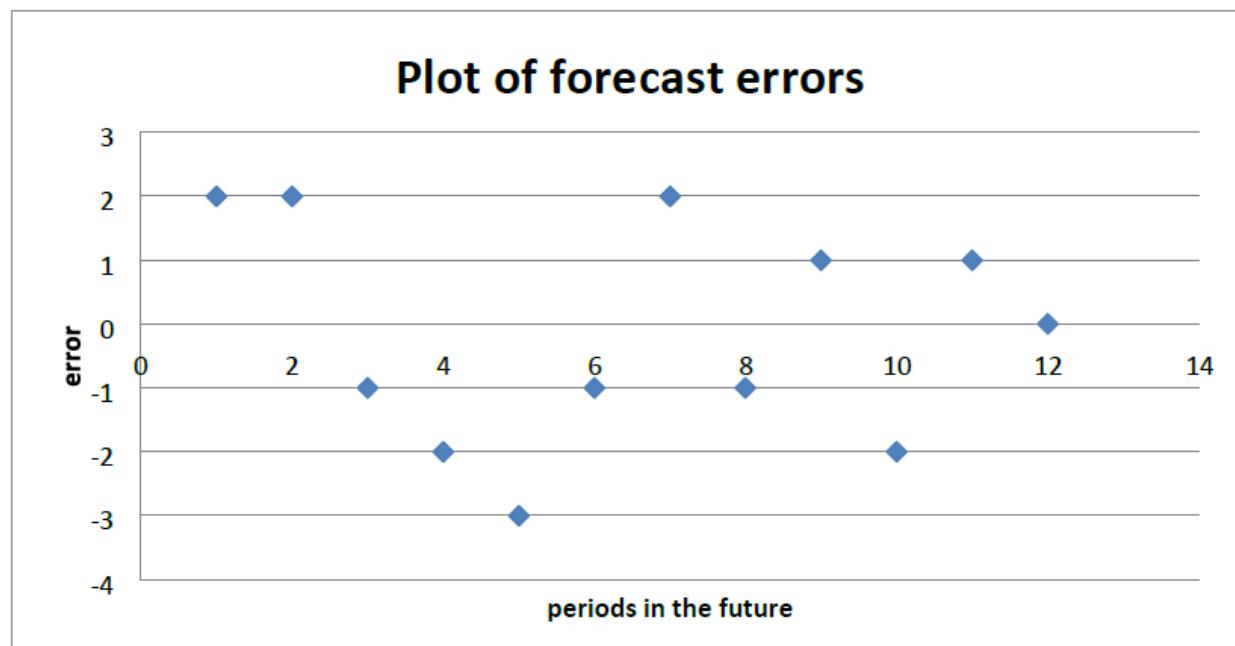
$$y = 1.1643x + 76.515 \quad (R^2 = 0.8395)$$

- After the second year in business the company has additional sales data and wants to know how well the model predicted the sales. (see d5.FORECAST.TREND)

## Sales vs. Forecast Sales for Trend Example



## Plot of Forecast Errors for the Trend Example



## General Case

Problems of an overestimated sales forecast:

1. Accumulation of inventory
2. Excessive labor force
3. Overbuilding of plant/equipment

Problems with under forecasting sales:

1. Customer complaints
2. Inadequate inventory

## More General Example and How to Get Residuals in Python

```
import numpy as np
import pandas as pd
import statsmodels.formula.api as smf
import csv
url = 'c:/users/gib/documents/d5.SALESINVENTORY.csv'
dat = pd.read_csv(url)
results = smf.ols('laginvent ~ lagsales', data=dat).fit()
residua=results.resid
residua=np.append(0,residua)
results = smf.ols('y ~ x + residua', data=dat).fit()
```

More general example using inventory data | print results.summary()

## OLS Regression Results

Dep. Variable:	y	R-squared:	0.394
Model:	OLS	Adj. R-squared:	0.390
Method:	Least Squares	F-statistic:	86.95
Date:		Prob (F-statistic):	
	8.30e-30	Time:	16:11:11
Log-Likelihood:	1099.1	No. Observations:	270
AIC:	-2192.		
Df Residuals:	267	BIC:	-2181. Df
Model:			2

coef	std err	t	P> t	[95.0% Conf. Int.]	
Intercept	0.0024	0.000	9.144	0.000	
0.002	0.003	x	0.1131	0.024	4.673
0.000	0.065		0.161	residua	-0.1077
0.010	-10.951		0.000	-0.127	-0.088

Omnibus:	36.496	Durbin-Watson:	0.893
Prob(Omnibus):	0.000	Jarque-Bera (JB):	53.657
Skew:	-0.835	Prob(JB):	2.23e-12
Kurtosis:	4.407	Cond. No.	96.3

# Moving-average (MA) Process

## MA Process

MA process is used in time series analysis for modeling univariate time series.

A MA process shows that the current values of the time series depends on the previous (unobserved) random shocks.

MA(q) process can be written as following:

$$X_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \dots + \theta_q \varepsilon_{t-q}$$

$\mu$  – the mean of the series

$\theta_1, \dots, \theta_q$  - model parameters

$\varepsilon_t, \varepsilon_{t-1}, \dots, \varepsilon_{t-q}$  – white noise error terms

$q$  – the order of the MA model

If we consider  $B$  as the lag operator, then MA(q) process can be written as following:

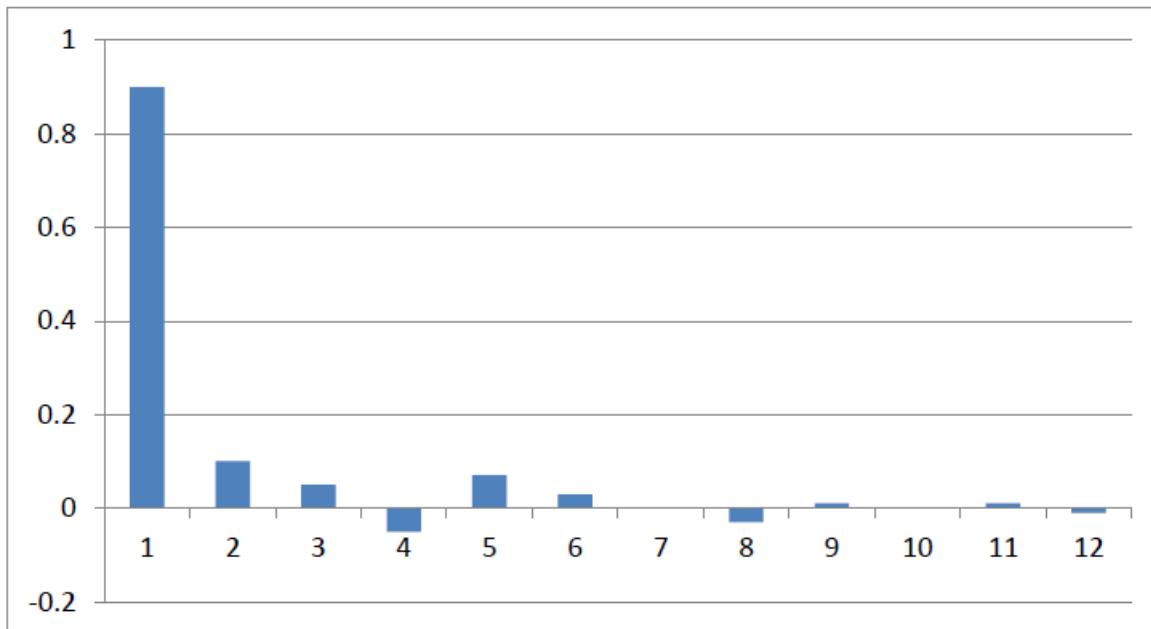
$$X_t = \mu + (1 + \theta_1 B + \dots + \theta_q B^q) \varepsilon_t$$

Why we choose a MA process instead of an AR process?

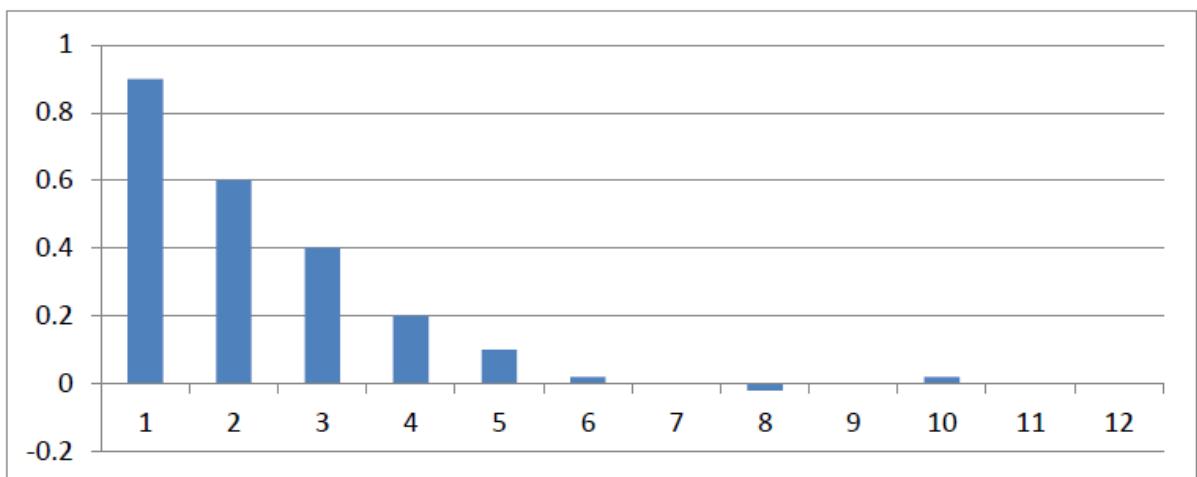
ACF and PACF represent good indicators for choosing a MA process or an AR process.

## ACF and PACF Graphs for a MA(1) Process

### ACF



### PACF



# ARMA Process

## Time Series Analysis

1. Plot the time series.

Look for trends, seasonal components, step changes, outliers.

2. Transform data so that residuals are **stationary**.
  - Remove trend and seasonal components.
  - Differencing.
  - Nonlinear transformations (log, p·).
3. Fit model to residuals.
4. Forecast time series by forecasting residuals and inverting any transformations.

How do we use data to decide on p, q?

1. Use sample ACF/PACF to make preliminary choices of model order.
2. Estimate parameters for each of these choices.
3. Compare predictive accuracy/complexity of each (using, e.g., AIC).

## AR(1) Process

$$y_t = \alpha_0 + \alpha_1 y_{t-1} + \varepsilon_t$$

$$E(y_t | y_{t-1}, \dots, y_0) = \alpha_0 + \alpha_1 y_{t-1}$$

The error term is white noise with expected mean equal zero.

Parameters for the constant and slope terms are found using maximum likelihood estimation.

$$L = f(y_0, \dots, y_T; \vartheta) = \prod_{i=0}^T f(y_{t-i}; \vartheta)$$

## ACF and PACF in AR(1)

Apply expectations operator:

$$E[y_t] = \alpha_0 / (1 - \alpha_1)$$

The unconditional variance is gamma,  $\gamma$ ,

$$\text{Gamma zero } \gamma_0 = E[(y_t - E[y_t])^2]$$

Which reduces to:

$$\text{Gamma zero } \gamma_0 = \sigma^2 / (1 - \alpha_1^2)$$

The kth-order autocovariance is

$$\text{Gamma } k \ \gamma_k = \alpha_1^k \ \gamma_0$$

For example:  $k=0: \gamma_0 = \alpha_1^0 \ \gamma_0 = \gamma_0$

The autocorrelation function (ACF) for AR(1) is:

$$\text{Greek letter rho } \rho_k = \gamma_k / \gamma_0 = \alpha_1^k$$

The ACF graphs produced by Python are from the following OLS regressions:

$$\text{Data } y_t = a_{10} + \rho_0 y_{t-1} + u_t$$

$$\text{Data } y_t = a_{20} + \rho_0 y_{t-2} + u_t$$

⋮

$$\text{Data } y_t = a_{ko} + \rho_0 y_{t-k} + u_t$$

The PACF graphs are generated using all the lag terms to k:

$$\text{Data } y_t = b_{10} + \varphi_{11} y_{t-1} + v_t$$

$$\text{Data } y_t = b_{20} + \varphi_{21} y_{t-1} + \varphi_{22} y_{t-2} + v_t$$

$$\text{Data } y_t = b_{20} + \varphi_{31} y_{t-1} + \varphi_{32} y_{t-2} + \varphi_{33} y_{t-3} + v_t$$

⋮

$$\text{Data } y_t = b_{ko} + \varphi_{k1} y_{t-1} + \varphi_{k2} y_{t-2} + \dots + \varphi_{kk} y_{t-k} + v_t$$

## Moving Average MA(1) Process

### The MA(1) process

The definition of the MA(1) process is given by ( $y_t$  is a stationary time series,  $e_t$  is a white noise error component):

$$\begin{aligned}
 W_t &= (1 - \theta_1 B) e_t & y_t &= e_t + \theta_1 e_{t-1} \\
 W_t &= e_t - \theta_1 e_{t-1} & & \\
 W_t - e_t &= -\theta_1(W_{t-1} - F_{t-1}) & \text{Forecast value of } y \text{ at time } t \text{ is equal} \\
 \text{to} & F_t = -\theta_1 W_{t-1} + \theta_1 F_{t-1} \\
 F_t &= -\theta_1 W_{t-1} + \theta_1(-\theta_1(W_{t-2} - F_{t-2})) \\
 F_t &= -\theta_1 W_{t-1} - \theta_1^2 W_{t-2} + \theta_1^2 F_{t-2} & F_t = \sum_{i=1}^{\infty} \theta_1^i y_{t-i} & \text{[aggregate from} \\
 &\dots & & i=1 \text{ to infinity]} \\
 F_t &= \sum_{i=1}^{\infty} -\theta_1^i W_{t-i}
 \end{aligned}$$

(V.I.1-139)

Where  $W_t$  is a stationary time series,  $e_t$  is a white noise error component, and  $F_t$  is the forecasting function on substituting  $\psi_1$  by  $-\theta$  and  $\psi_j = 0$  for all  $j > 1$  into eq. (V.I.1-46) and (V.I.1-45) we obtain

$$\begin{cases} \gamma_0 = \sigma_{e_t}^2(1 + \theta_1^2) \\ \gamma_1 = -\sigma_{e_t}^2 \theta_1 \\ \gamma_k = 0 \text{ for } k > 1 \end{cases} \quad (\text{V.I.1-140})$$

Therefore, the pattern of the **theoretical ACF** is

$$\begin{cases} \rho_1 = \frac{-\theta_1}{1 + \theta_1^2} \\ \rho_k = 0 \text{ for } k > 1 \end{cases} \quad (\text{V.I.1-141})$$

Note that from eq. (5.I.1-141) it follows that (V.I.1-142)

$$\theta_1 p_1 + \theta_1 + p_1 = 0 \quad 2 \quad (\text{V.I.1-142})$$

Which is a parabola. This quadratic equation must have real roots, thus  $-\frac{1}{2} < p_1 < \frac{1}{2}$ . It is interesting to note that on substituting  $\theta_1$  by  $1/\theta_1$  eq. (5.I.1 – 142) still holds.

- This implies that there exist at least two MA(1) processes which generate the same theoretical ACF.
- Since an MA process consists of a finite number of y weights it follows that the process is always stationary. However, it is necessary to impose the so-called invertibility restrictions such that the MA(q) process can be rewritten into a AR(infinity) model.
- The invertibility restriction can be easily shown to be  $|\theta_i| < 1$  such that converges.

$$W_t = \sum_{i=1}^{+\infty} \pi_i W_{t-1} + e_t \quad (\text{V.I.1-143})$$

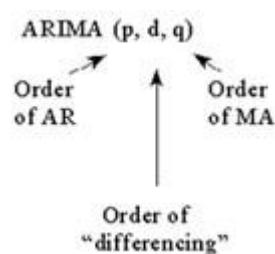
On using the Yule-Walker equations and eq. (V.I.1-141) it can be shown that the theoretical PACF is

$$\Phi_{kk} = \frac{-\theta_1^k (1 - \theta_1^2)}{1 - \theta_1^{2(k+1)}} \quad (\text{V.I.1-144})$$

Such that  $|\Phi_{kk}| < \theta_1^k$

Hence the theoretical PACF is dominated by an exponential function which decreases. If  $\theta_1$  is positive, it follows from (V.I.1-141) that  $\theta_1$  is smaller than 1 and therefore the PACF from (V.I.1 – 144) alternates in sign. Otherwise, if  $\theta_1 > 1$ , the theoretical PACF is negative as it damps out.

The model that you will build should match the history which can then be extrapolated into the future. The actual minus the fitted values are called the residuals. The residuals should be random around zero (i.e. Gaussian) signifying that the pattern has been captured by the model.



## Stationarity

Non-stationarity is a property common to many financial time series. It means that a variable has no clear tendency to return to a constant value or a linear trend.

- Fitting the MA estimates is more complicated than with AR models because the error terms are not observable. This means that iterative non-linear fitting procedures need to be used in place of linear least squares. MA models also have a less obvious interpretation than AR models.
- Sometimes the ACF and PACF will suggest that a MA model would be a better model choice and sometimes both AR and MA terms should be used in the same model.

## Autoregressive Integrated Moving Average (ARIMA)

Trend, cycles, and ARIMA models. But how can we apply an ARMA model to a nonstationary series? Most real series show a trend, an average increase or decrease over time. For example, inflation generally increases over time, and deaths due to measles have decreased over time. Series also show cyclic behavior.

For example, influenza deaths rise in winter and dip in summer. We can remove trends and cycles from a series through differencing.

For example, suppose  $t$  is in months and  $Y_t$  is a series with a linear trend. That is, every month the series increases on average (in expectation) by some constant amount  $C$ . Since  $Y_t = C + Y_{t-1} + N_t$  where  $N_t$  is a random "noise" component with expectation zero, then the differences,  $Y_t - Y_{t-1}$  are equal to  $C + N_t$ . Thus,  $N_t + C$  (or just  $N_t$ ) is a stationary series with the linear trend removed.

- We could now apply the ARMA model to  $Z_t = C + N_t$ , even though it is not appropriate to do so to  $Y_t$  directly.
- By analogy, higher order polynomial trends, (i.e., quadratic trends, cubic trends) can be removed by differencing more than once.
- Cycles can be removed by differencing at different lags.

For example, a yearly cycle in a monthly series of stock prices can often be removed by forming the difference  $Z_t = Y_t - Y_{t-12}$ , a 12th order (lag) difference by differencing several.

Once an ARMA model for  $Z_t$  is known, we could reverse the differencing to form the original  $Y_t$  from the  $z_t$ .

We term this process integration (although the word summation might seem more appropriate).

The combined model for the original series  $Y_t$ , which involves autoregression, moving average, and integration elements is termed the  $ARIMA(p, d, q)$  model (model of orders  $p$ ,  $d$ , and  $q$ ) with  $P$  AR terms,  $d$  differences, and  $q$  MA terms.

## Autocorrelations and Partial Autocorrelations

We use the autocorrelation and partial autocorrelation function to determine  $p$  orders and  $q$  orders and confirm the differencing that was suggested by the plot.

- The autocorrelation at lag  $k$ ,  $ACF(k)$ , is the (linear) Pearson correlation between observations  $k$  time periods (lags) apart.
- If the  $ACF(k)$  differs significantly from zero, the serial dependence among the observations must be included in the ARIMA model.
- Like the  $ACF(k)$ , the partial autocorrelation at lag  $k$ , or  $PACF(k)$ , measures correlation
- Observations  $k$  lags apart. However, the  $PACF(k)$  removes, or "partials out," all intervening lags.

In general, for *autoregressive* processes [ARIMA( $p, 0, 0$ )]

- the ACF declines exponentially
- the PACF spikes on the first  $p$  lag

By contrast, for *moving average* processes [ARIMA( $0, 0, q$ )]

- the ACF spikes on the first  $q$  lags
- the PACF declines exponentially

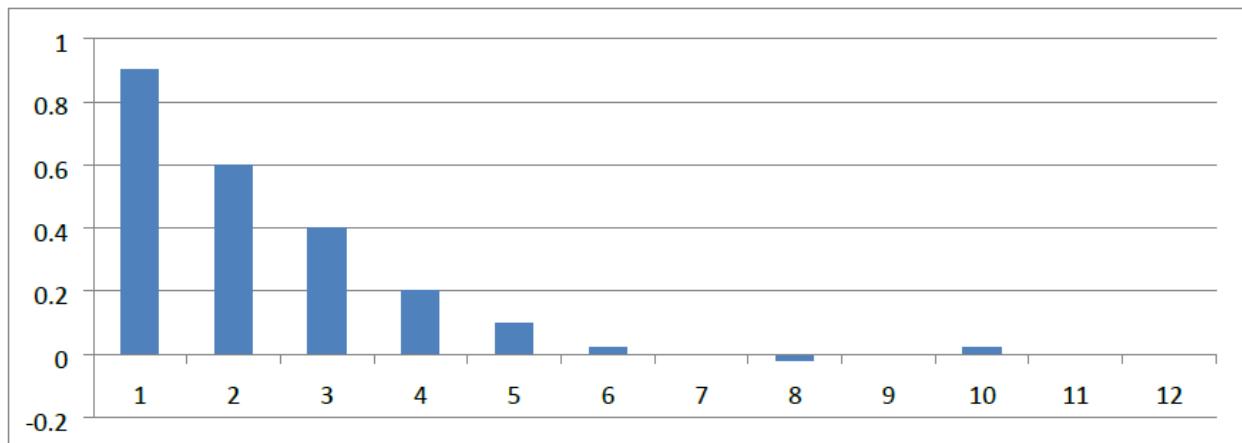
Mixed processes [ $ARIMA(p, d, q)$ ] decline on both ACF and the PACF. Finally, if the ACF or PACF declines slowly (i.e., has autocorrelations with  $t$  values  $> 1.6$  for more than five or six consecutive lags), the process is probably not stationary and therefore should be differenced. Usually, a large number of significant autocorrelations or partial autocorrelations indicates nonstationarity and a need for further differencing.

## Identifying an AR process

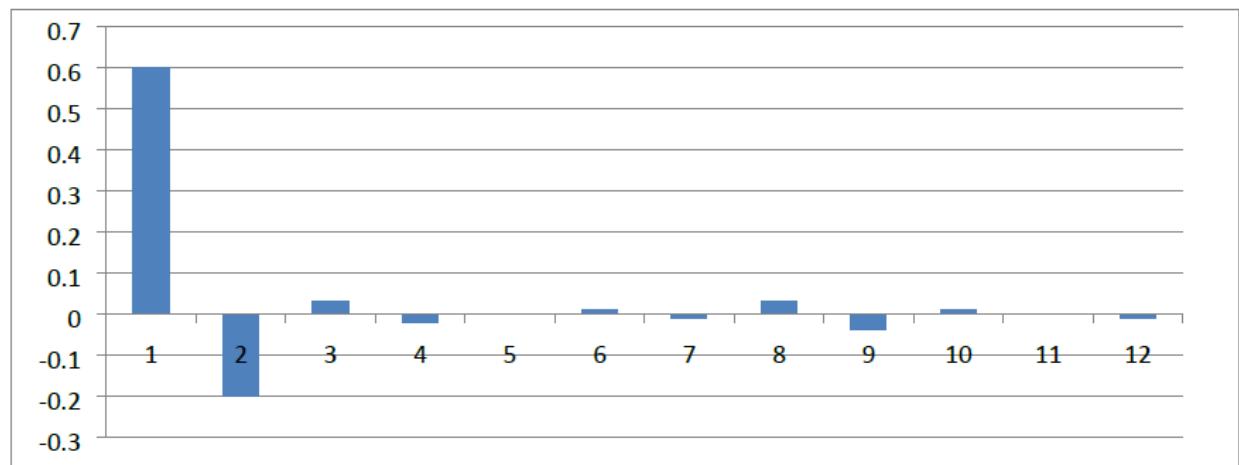
- If the PACF displays a sharp cutoff while the ACF decays more slowly (i.e., has significant spikes at higher lags), we say that the series displays an "AR signature"
- The lag at which the PACF cuts off is the indicated number of AR terms

**CASE 1:** The following ACF and PACF graphs suggests ARIMA(1,0,0) [The lag at which the PACF cuts off is the indicated number of AR terms.]

### ACF

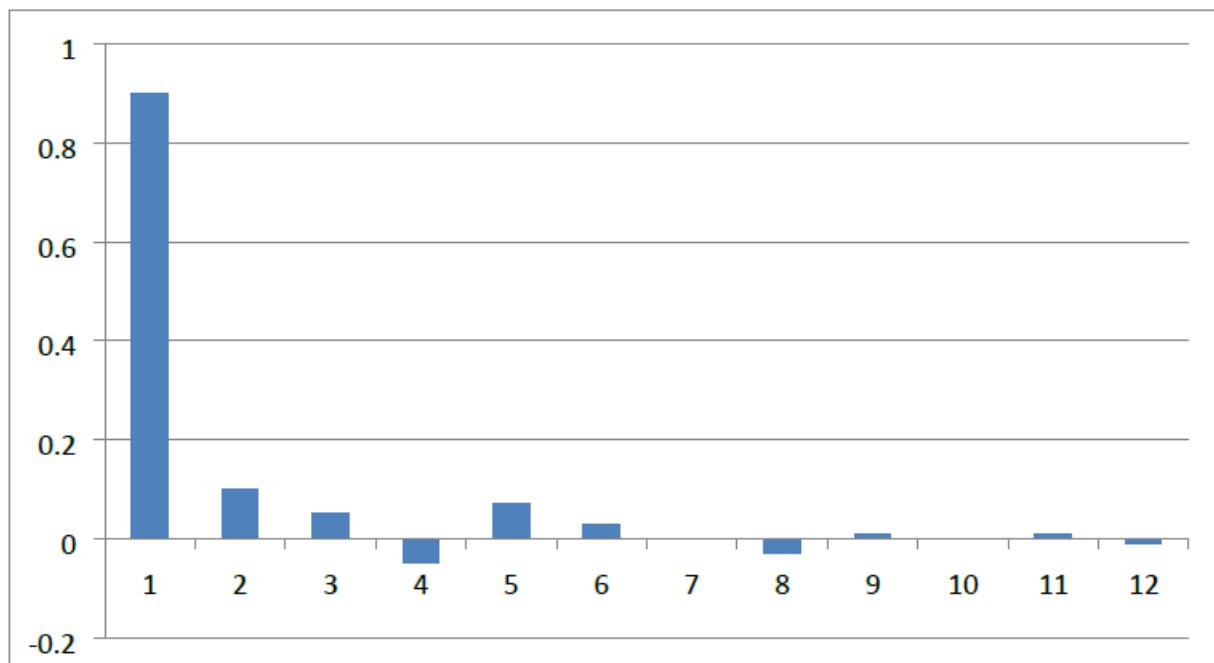


### PACF

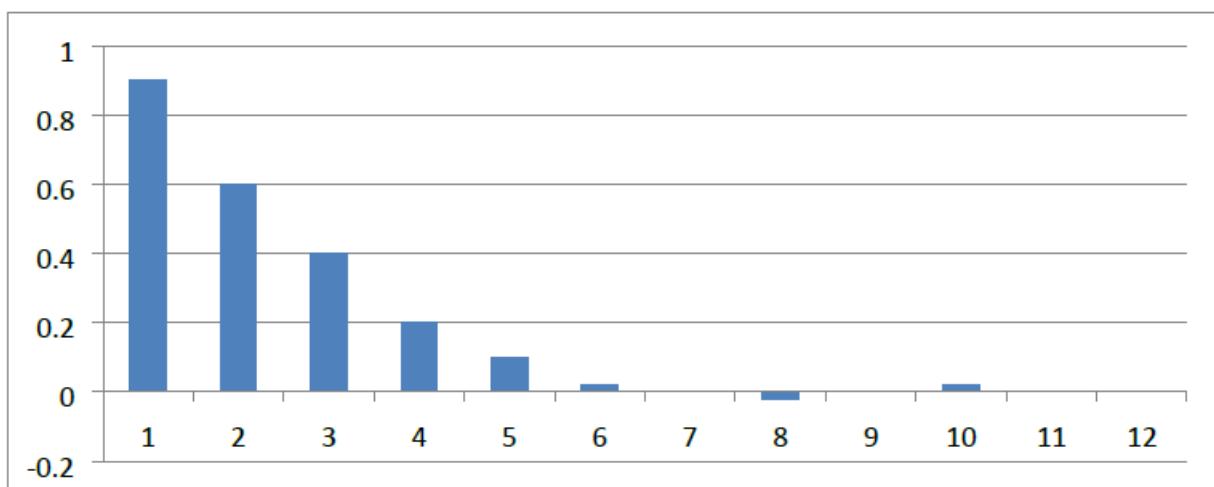


**CASE 2:** The following ACF and PACF graphs suggests ARIMA(0,0,1) Looks a lot like the obverse of case 1

### ACF

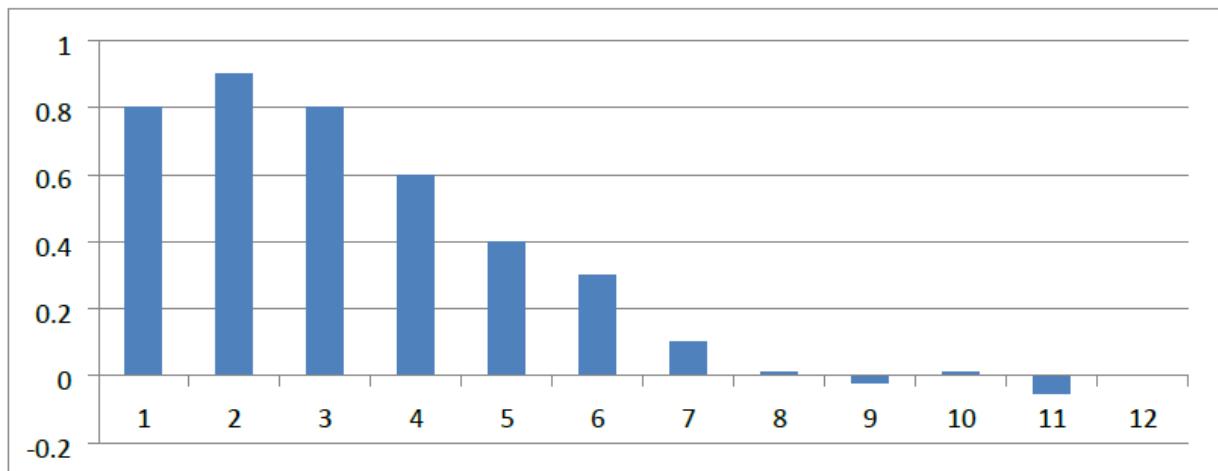


### PACF

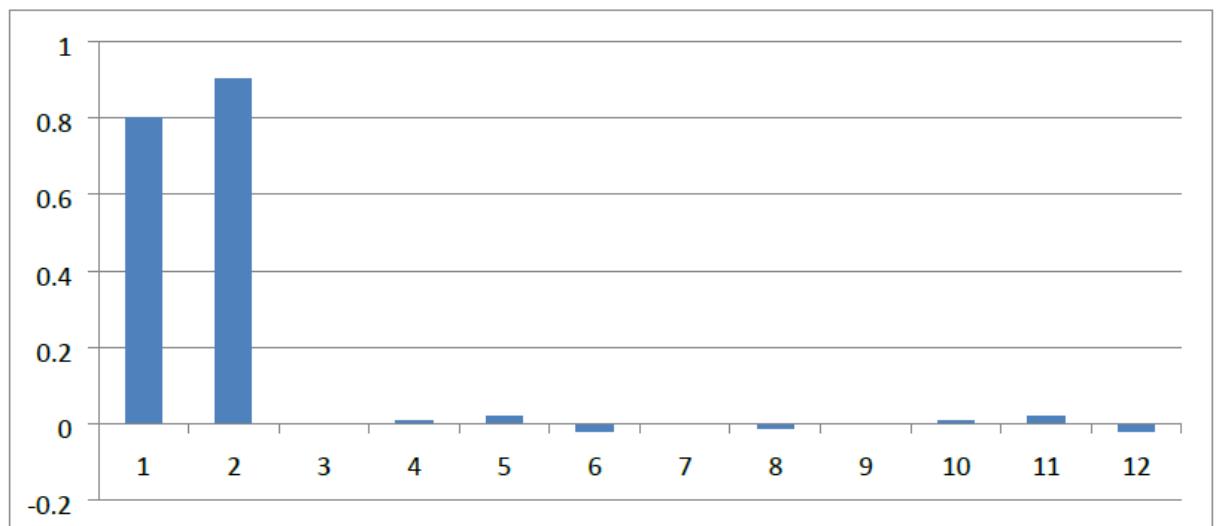


**Case 3:** The following ACF and PACF graphs suggests ARIMA(2,0,0) [The lag at which the PACF cuts off (at 2) is the indicated number of AR terms.]

### ACF

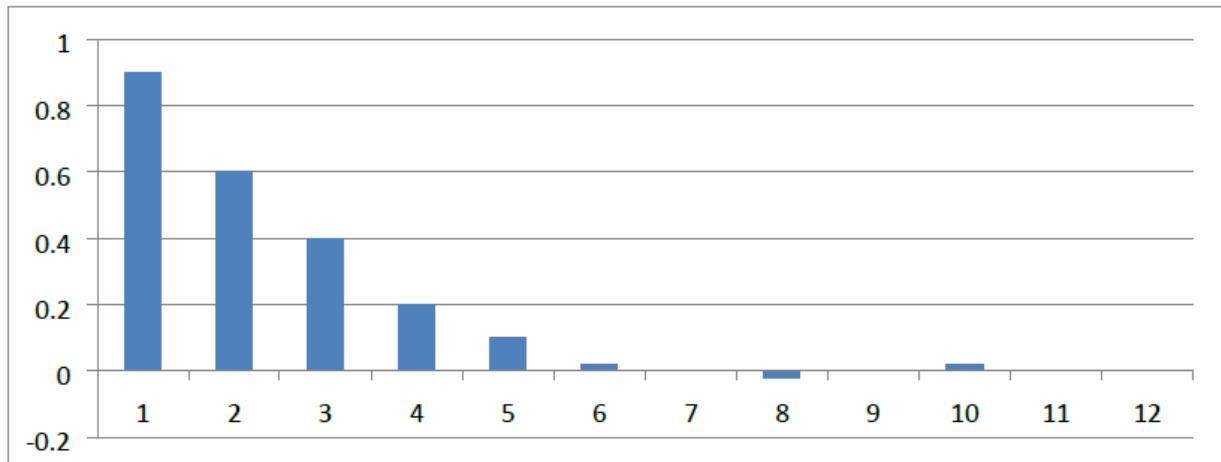


### PACF



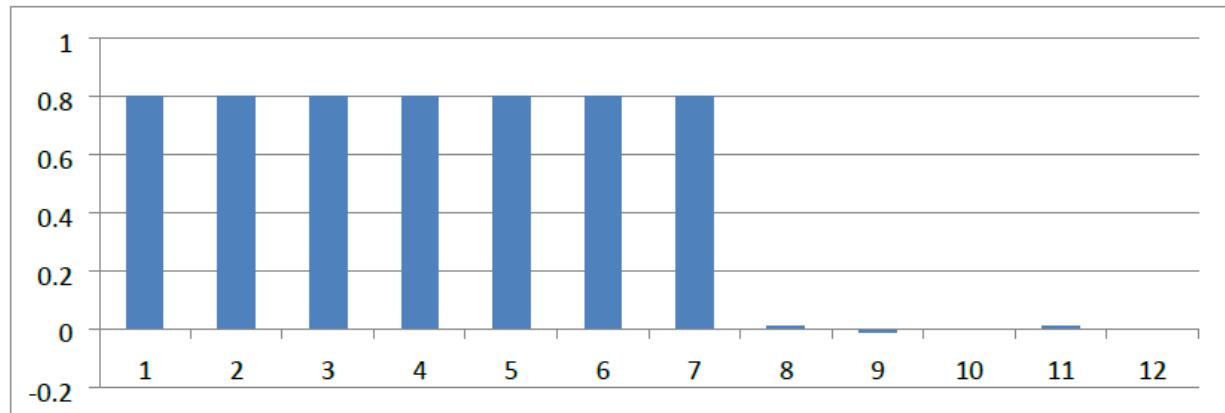
**Case 4:** The following ACF and PACF graphs suggests ARIMA(p,0,q).

Both ACF and PACF Look as Follows:

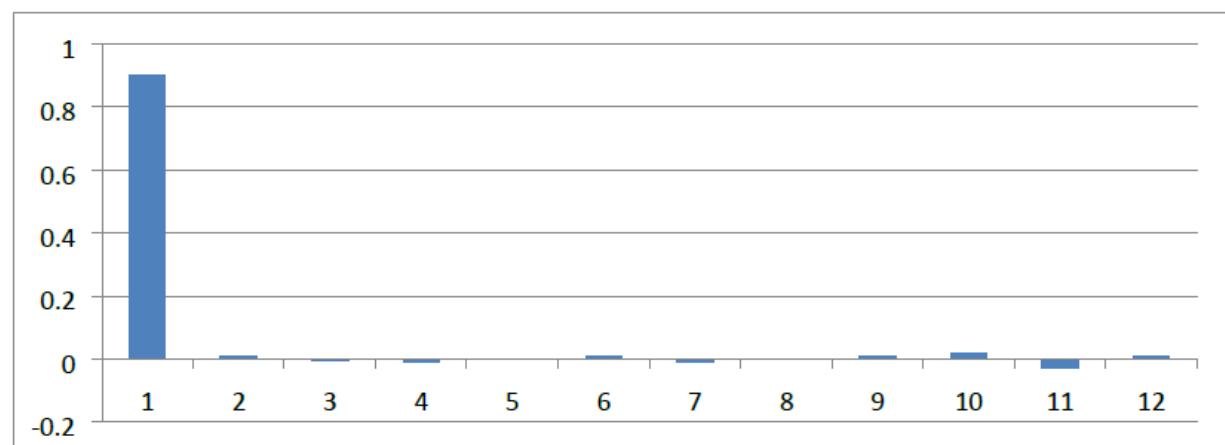


**Case 5:** The following ACF and PACF graphs suggests ARIMA(0,1,0).

### ACF

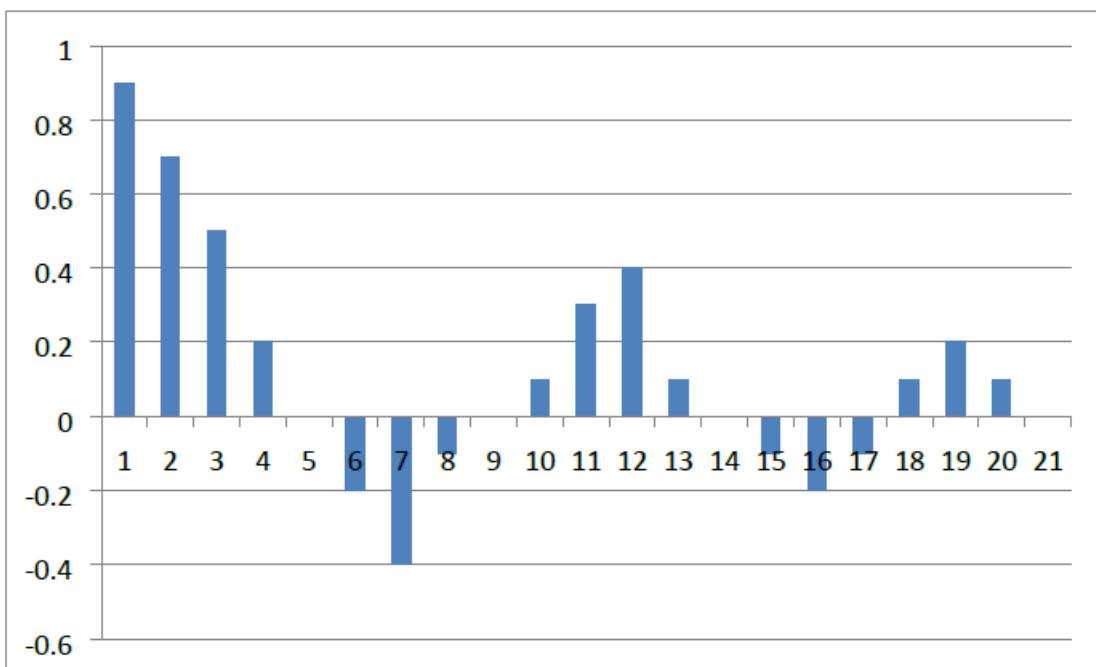


### PACF



**Case 6:** The following (called a *damp sine wave*) ACF and PACF graph may also suggest ARIMA(0,1,0).

**PACF (Same as in the Last Case but the ACF Looks as Follows):**



For a thorough treatment of the ACF/ PACF discussion see:

<http://datamfr.files.wordpress.com/2012/10/acf-and-pacf-arima-models.pdf>

# Box-Jenkins

## Introduction

Suppose we want to forecast house prices in the largest cities of Australia using an ARMA model. The million-dollar question is what model should we use? Are house prices in Australia following an AR process, a MA process or an ARMA process? Box-Jenkins method can provide us answers to this question.

## Box-Jenkins Method

Box-Jenkins method is applied at ARMA/ARIMA models for finding the best fit.

*Correlogram* – plot of the autocorrelation function (ACF)

*Partial correlogram* – plot of the partial autocorrelation function (PACF)

Steps in Box-Jenkins methodology	Description
1. Identification of the model	Choosing p, d and q parameters using the correlogram and the partial correlogram
2. Parameter estimation of the chosen model	The parameters are estimated using Maximum Likelihood Estimation (MLE) or non-linear least squares
3. Diagnostic checking	Are the residuals normally distributed? If they are, we go to step 4 else we go to step 1.
4. Forecasting	Forecast house prices in the Australian largest cities using the ARMA model

## Theoretical Patterns of ACF and PACF

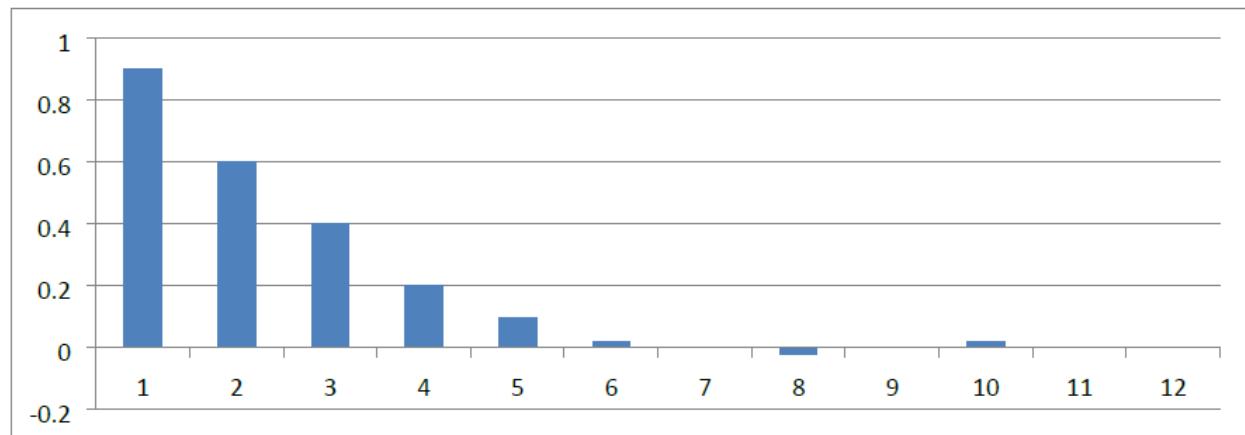
Type of model	Typical pattern of ACF	Typical pattern of PACF
AR(p)	Decays exponentially or with damped sine wave pattern or both	Significant spikes through lags p
MA(q)	Significant spikes through lags q	Declines exponentially
ARMA(p,q)	Exponential decay	Exponential decay

*Source: Damodar Gujarati – “Basic Econometrics”*

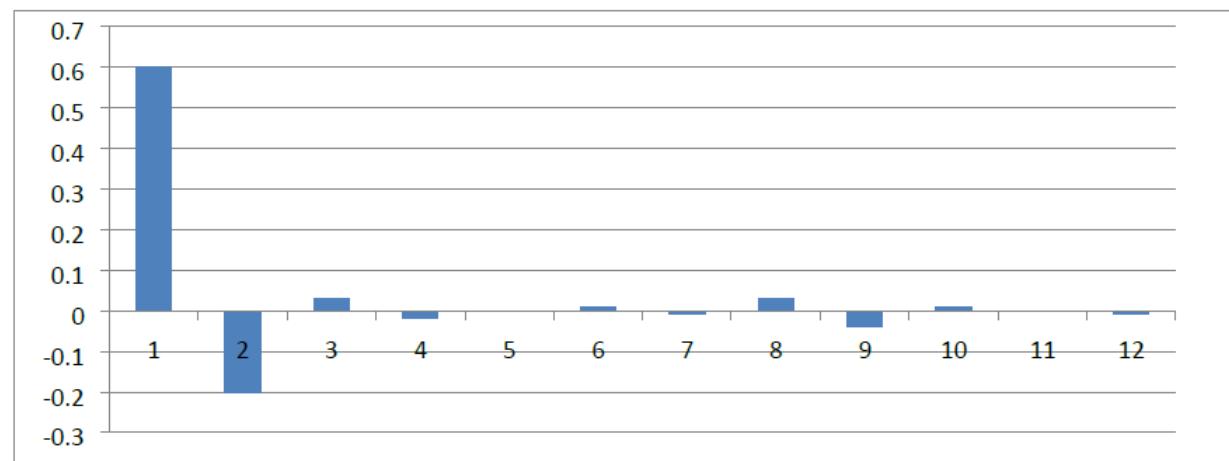
## ACF and PACF Examples

**Case 1:** The following ACF and PACF graphs suggests ARIMA(1,0,0) [The lag at which the PACF cuts off is the indicated number of AR terms]

### ACF

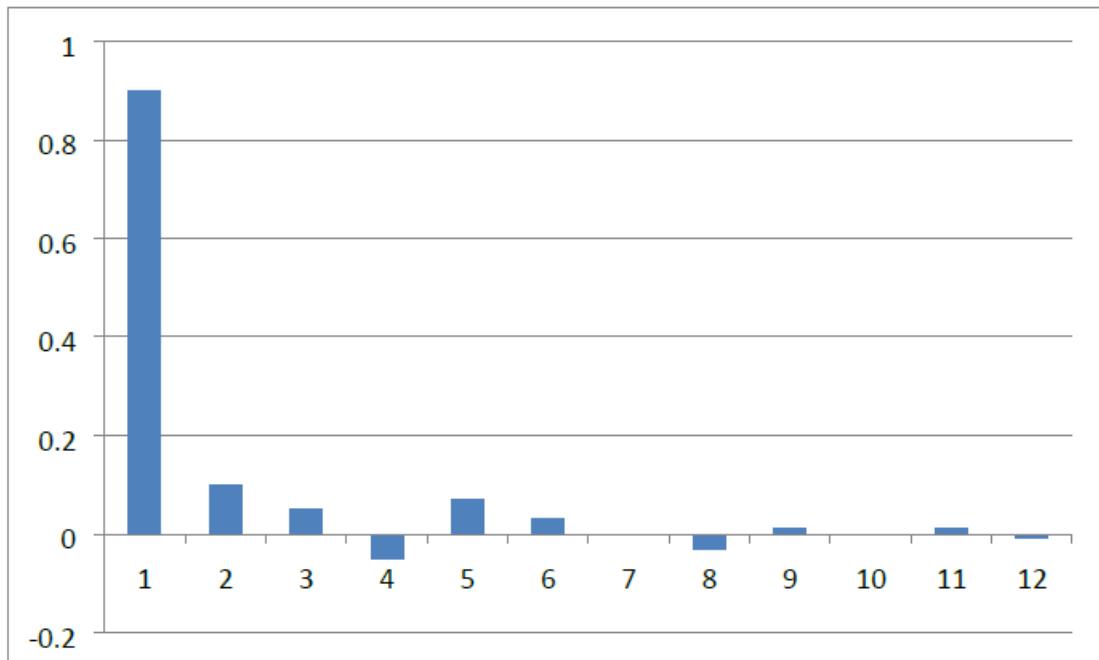


### PACF

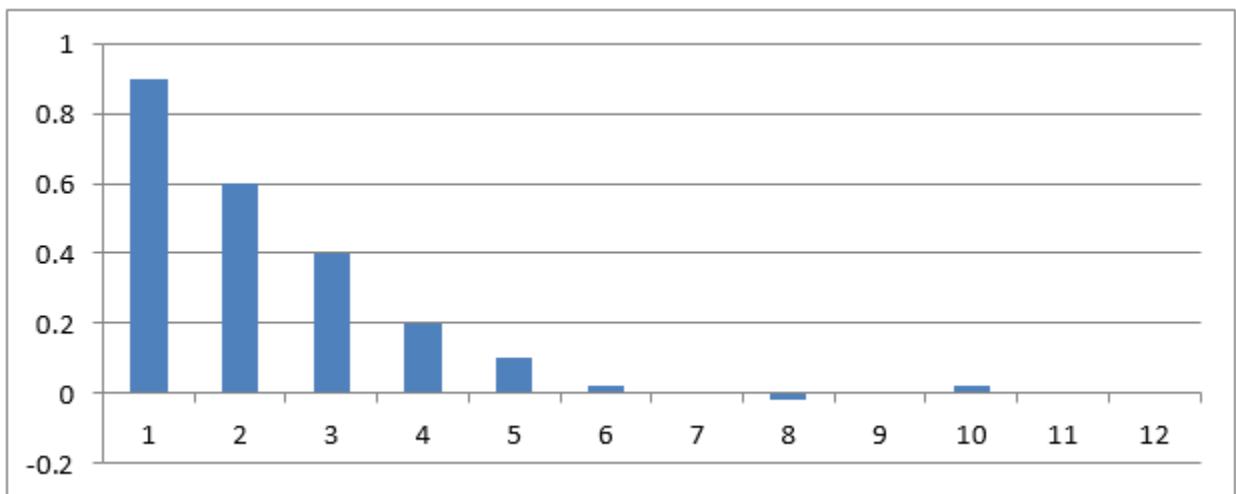


**Case 2:** The following ACF and PACF graphs suggests ARIMA(0,0,1) [Looks a lot like the obverse of case 1]

### ACF

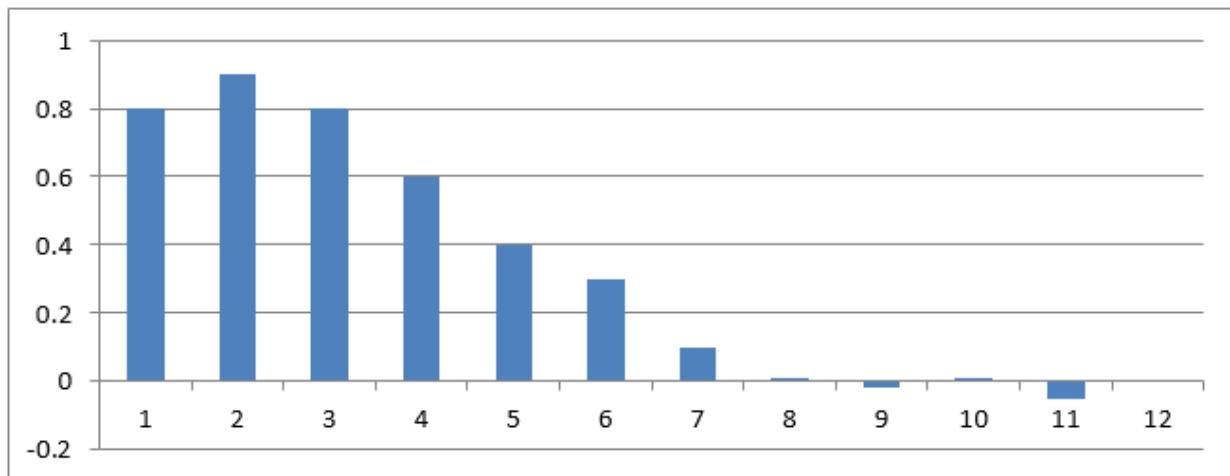


### PACF

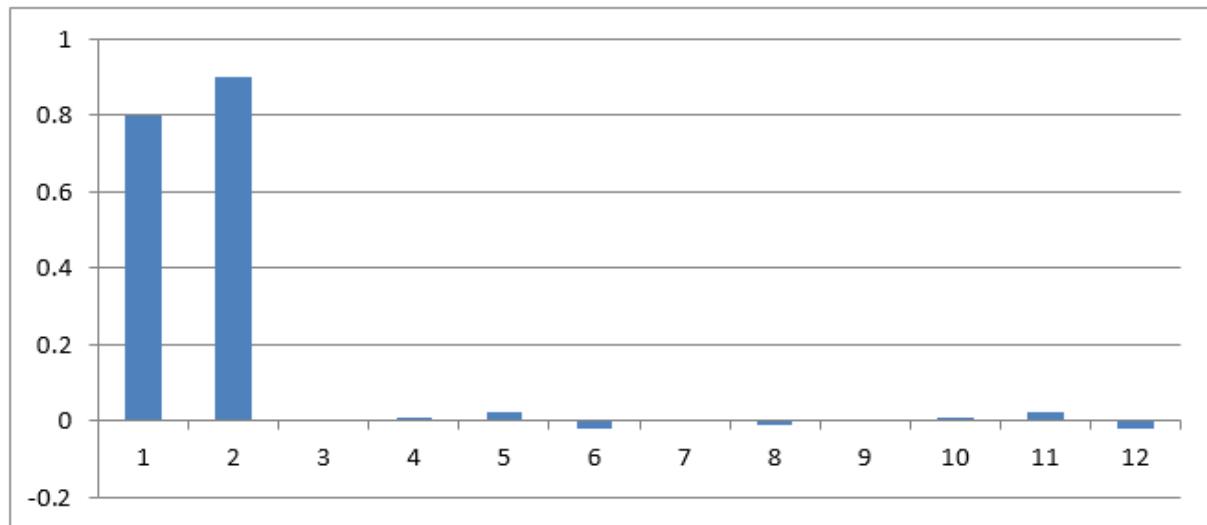


**Case 3:** The following ACF and PACF graphs suggests ARIMA(2,0,0) [The lag at which the PACF cuts off (at 2) is the indicated number of AR terms.]

### ACF

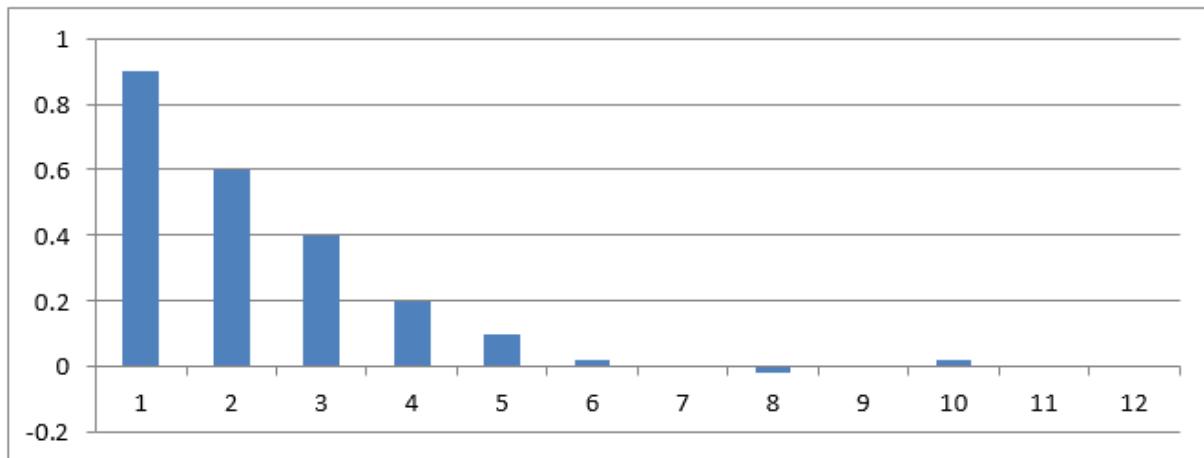


### PACF



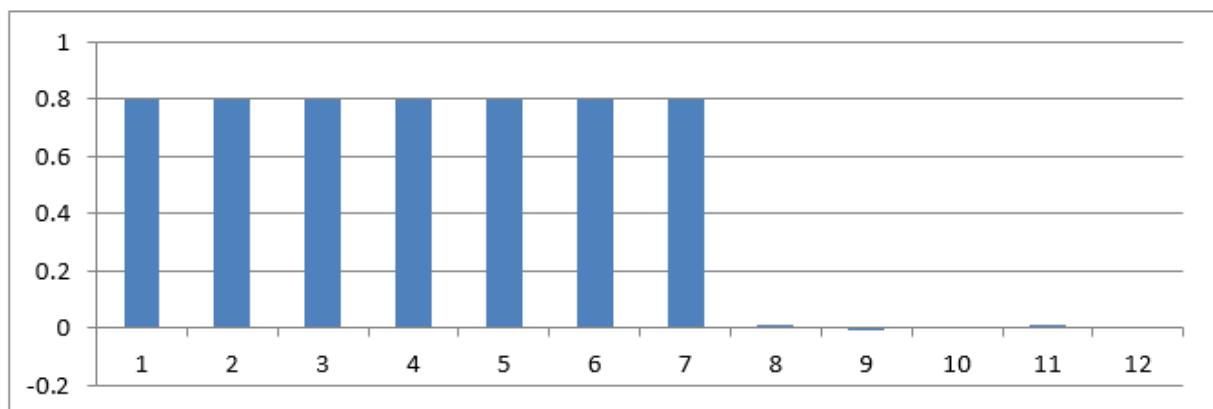
**Case 4:** The following ACF and PACF graphs suggests ARIMA(p,0,q).

**Both ACF and PACF Look as Follows**

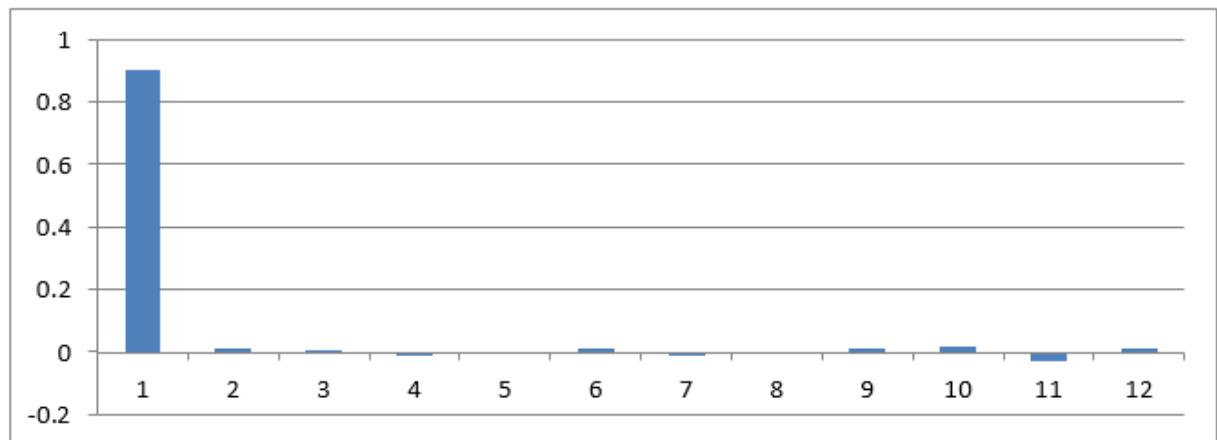


**Case 5:** The following ACF and PACF graphs suggests ARIMA(0,1,0).

### ACF

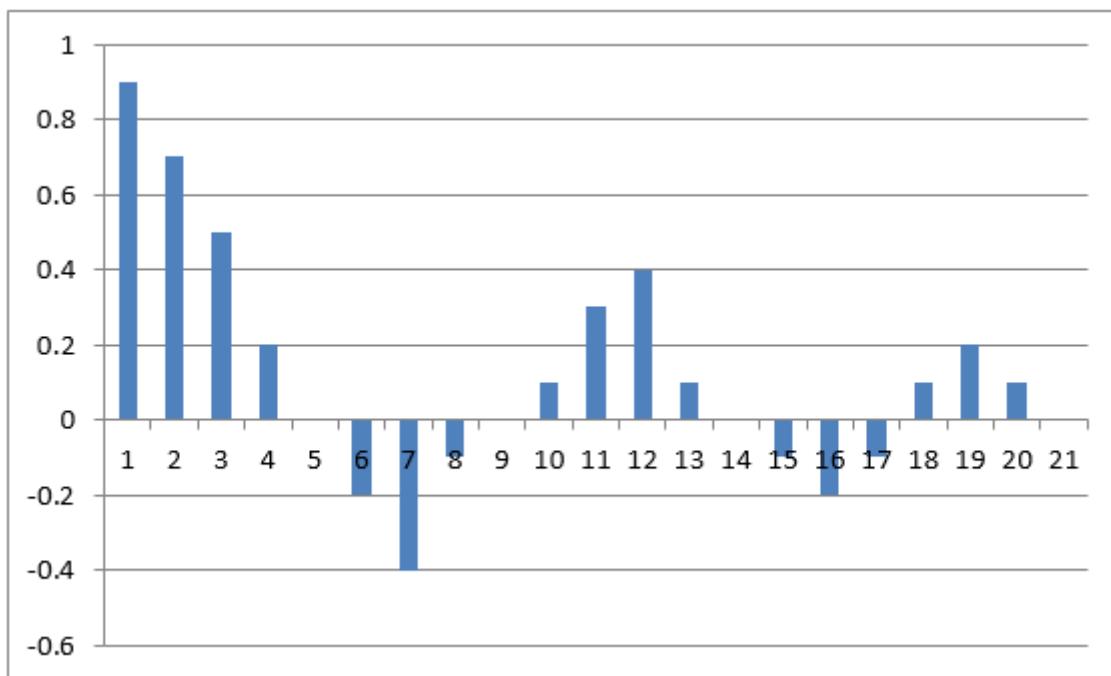


### PACF



**Case 6:** The following (called a *damp sine wave*) ACF and PACF graph may also suggest ARIMA(0,1,0).

### PACF (Same as in the Last Case but the ACF Looks as Follows):



- Choosing the right p and q parameters is rather an art. Why? Real ACFs and PACFs do not match exactly with the theoretical patterns as indicated in the above table.
- ARMA modeling requires practice as it is not an easy task for choosing the right p and q.
- Most of the time, quant analysts combine the information provided by ACF and PACF with the one provided by information criterion (Akaike Information Criterion, Bayesian Information Criterion and Hannan-Quinn Information Criterion).

## Modeling House Prices in The Largest Cities of Australia

We want to see what type of process follow the house prices in Australia. The steps are:

1. Collect the data

Source: Australian Bureau of Statistics:

<http://www.abs.gov.au/AUSSTATS/abs@.nsf/DetailsPage/6416.0Sep%202014?OpenDocument#Time>

We download Table 1. Residential Property Price Index, Index Numbers and Percentage Changes (xls document).

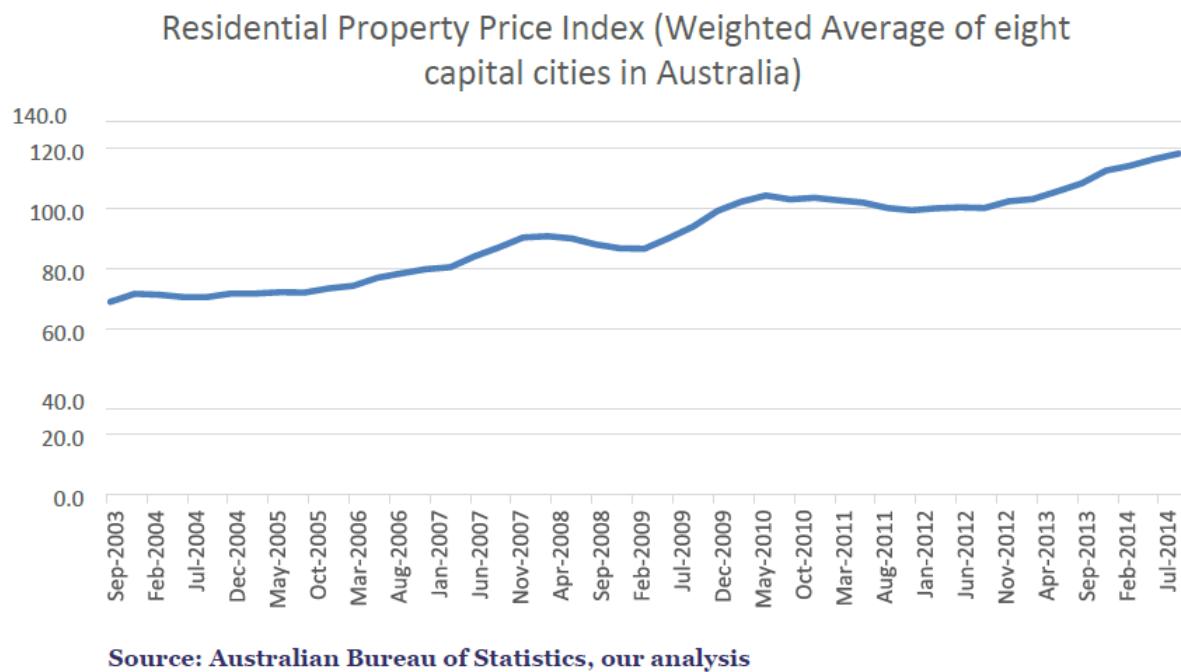
2. Select the indicator

We consider the following indicator in our analysis – Residential Price Index; Weighted average of eight capital cities (column J)

	A	D	E	F	G	H	I	J
1		Residential Property Price Index ; Brisbane ;	Residential Property Price Index ; Adelaide ;	Residential Property Price Index ; Perth ;	Residential Property Price Index ; Hobart ;	Residential Property Price Index ; Darwin ;	Residential Property Price Index ; Canberra ;	Residential Property Price Index ; Weighted average of eight capital cities ;
2 Unit	Index Numbers	Index Numbers	Index Numbers	Index Numbers	Index Numbers	Index Numbers	Index Numbers	Index Numbers
3 Series Type	Original	Original	Original	Original	Original	Original	Original	Original
4 Data Type	INDEX	INDEX	INDEX	INDEX	INDEX	INDEX	INDEX	INDEX
5 Frequency	Quarter	Quarter	Quarter	Quarter	Quarter	Quarter	Quarter	Quarter
6 Collection Month	3	3	3	3	3	3	3	3
7 Series Start	Sep-2003	Sep-2003	Sep-2003	Sep-2003	Sep-2003	Sep-2003	Sep-2003	Sep-2003
8 Series End	Sep-2014	Sep-2014	Sep-2014	Sep-2014	Sep-2014	Sep-2014	Sep-2014	Sep-2014
9 No. Obs	45	45	45	45	45	45	45	45
10 Series ID	A83728401F	A83728410J	A83728419C	A83728428F	A83728437J	A83728446K	A83728455L	
11 Sep-2003	64.2	62.2	48.3	61.2	40.5	68.3	69.0	
12 Dec-2003	69.4	63.9	50.6	66.5	43.3	70.9	71.6	
13 Mar-2004	70.7	64.8	52.5	68.7	45.5	69.9	71.3	
14 Jun-2004	71.7	65.9	53.7	71.8	45.2	70.1	70.6	
15 Sep-2004	71.6	67.1	54.3	70.2	45.3	70.2	70.6	

Source: Australian Bureau of Statistics, our analysis

### 3. Make a plot



Cities considered in the analysis: Sydney, Melbourne, Brisbane, Adelaide, Perth, Hobart, Darwin and Canberra

### 4. We apply natural logarithm to our data

### 5. We apply the first difference in order to obtain stationarity

### 6. Plot ACF and PACF

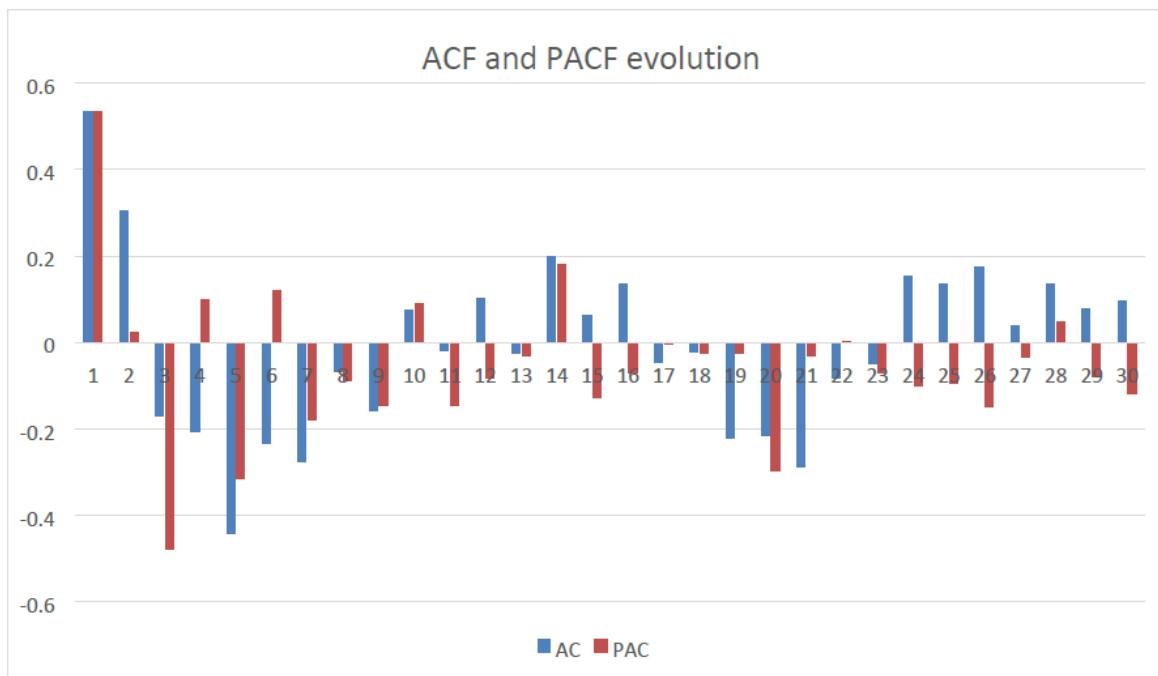
Date: 06/11/15 Time: 15:49  
 Sample: 2003Q3 2014Q3  
 Included observations: 44

	Autocorrelation	Partial Correlation	AC	PAC	Q-Stat	Prob
1			1	0.535	0.535	13.462 0.000
2			2	0.304	0.025	17.907 0.000
3			3	-0.169	-0.477	19.313 0.000
4			4	-0.207	0.100	21.476 0.000
5			5	-0.441	-0.315	31.574 0.000
6			6	-0.234	0.120	34.485 0.000
7			7	-0.276	-0.178	38.648 0.000
8			8	-0.068	-0.088	38.908 0.000
9			9	-0.157	-0.145	40.326 0.000
10			10	0.075	0.089	40.663 0.000
11			11	-0.019	-0.147	40.686 0.000
12			12	0.103	-0.083	41.357 0.000
13			13	-0.024	-0.032	41.394 0.000
14			14	0.199	0.181	44.068 0.000
15			15	0.063	-0.129	44.346 0.000
16			16	0.135	-0.069	45.665 0.000
17			17	-0.046	0.008	45.825 0.000
18			18	-0.023	-0.153	45.866 0.000
19			19	-0.221	-0.025	49.821 0.000
20			20	-0.214	-0.297	53.692 0.000
21			21	-0.287	-0.031	60.951 0.000
22			22	-0.083	0.003	61.585 0.000
23			23	-0.049	-0.071	61.815 0.000
24			24	0.154	-0.100	64.215 0.000
25			25	0.136	-0.095	66.189 0.000
26			26	0.176	-0.148	69.658 0.000
27			27	0.037	-0.033	69.820 0.000
28			28	0.134	0.046	72.103 0.000
29			29	0.078	-0.079	72.924 0.000
30			30	0.095	-0.119	74.222 0.000

## 7. Comment the results

- According to the results obtained, ACF and PACF decline with some spikes.
- ACF tends to have a damp sine wave evolution
- A slight damp sine wave evolution might be seen at PACF

According to Professor Walter Enders if ACF and PACF decay (either direct or oscillatory) then we have an ARMA process. According to our graph, ACF and PACF decline after lag 1 which may indicate that the house prices in the large Australian cities follow an ARMA(1,1) process. The damp sine wave evolutions of ACF and PACF might also indicate an ARIMA(0,1,0) process.



# Wold's Decomposition

## Wold's Decomposition

So far, we have focused on ARMA models, which are linear time series models.

Is there any relationship between a general covariance stationary process (maybe nonlinear) to linear representations? The answer is given by the Wold decomposition theorem:

**Proposition 2** (Wold Decomposition) Any zero-mean covariance stationary process  $x_t$  can be represented in the form

$$x_t = \sum_{j=0}^{\infty} \psi_j \varepsilon_{t-j} + V_t$$

Where

$$\psi_0 = 1 \text{ and } \sum_{j=0}^{\infty} \psi_j^2 < \infty$$

$$\varepsilon_t \sim W(0, \sigma_\varepsilon^2)$$

$$(\varepsilon_t V_s) = 0 \quad \forall s, t > 0$$

$\varepsilon_t$  is the error in forecasting  $x_t$  on the basis of a linear function of lagged  $x$ :

$$\varepsilon_t = x_t - E(x_t | x_{t-1}, x_{t-2}, \dots)$$

$V_t$  is a deterministic process and it can be predicted from a linear function of lagged  $x$

Wold decomposition says that any covariance stationary process has a linear representation: a linear deterministic component ( $v_t$ ) and a linear indeterministic component ( $\varepsilon_t$ ). If  $V_t = 0$ , then the process is said to be purely-non-deterministic, and the process can be represented as a  $MA(\infty)$  process. Basically,  $\varepsilon_t$  is the error from the projection of  $x_t$  on lagged  $x$ , therefore it is uniquely determined and it is orthogonal to lagged  $x$  and lagged  $\varepsilon$ . Since this error  $\varepsilon$  is the residual from the projections, it may not be true errors in the DGP of  $x_t$ . Also note that the error term ( $\varepsilon$ ) is a white noise process, and does not need to be iid.

# Vector Autoregressive (VAR) Models

## VAR

Vector autoregressive (VAR) models are multivariate econometric models used on a large scale in:

- Forecasting
- Obtaining the Impulse Response Functions
- Monetary policy analysis

A VAR model is a system of equations (regressions) where the number of equations is equal to the number of variables introduced in the model.

A vector autoregressive model of order 1, denoted as VAR(1) is as follows:

$$\begin{aligned}x_{t,1} &= \alpha_1 + \varphi_{11}x_{t-1,1} + \varphi_{12}x_{t-1,2} + \varphi_{13}x_{t-1,3} + w_{t,1} \\x_{t,2} &= \alpha_2 + \varphi_{21}x_{t-1,1} + \varphi_{22}x_{t-1,2} + \varphi_{23}x_{t-1,3} + w_{t,2} \\x_{t,3} &= \alpha_3 + \varphi_{31}x_{t-1,1} + \varphi_{32}x_{t-1,2} + \varphi_{33}x_{t-1,3} + w_{t,3}\end{aligned}$$

$x_{t,1}, x_{t,2}$  and  $x_{t,3}$  – time series variables.

Example: A quant analyst studies the connection between GDP and M2 money supply in U.S. economy.

Model: VAR (Source: FRED database)

## Frequency: Quarterly

Number of lags: 1

Period: 1999Q1 – 2015Q1

The equations are the following:

$$GDP = C(1,1)*GDP(-1) + C(1,2)*M2(-1) + C(1,3)$$

$$M2 = C(2,1)*GDP(-1) + C(2,2)*M2(-1) + C(2,3) \quad (1)$$

Therefore the coefficients  $C(1,1)$ ,  $C(1,2)$ ,  $C(1,3)$ ,  $C(2,1)$ ,  $C(2,2)$  and  $C(2,3)$  must be estimated.

The equations can also be written in matrix form:

$$x_t = \begin{pmatrix} GDP \\ M2 \end{pmatrix}$$

$$A_0 = \begin{pmatrix} C(1,3) \\ C(2,3) \end{pmatrix}$$

$$A_1 = \begin{pmatrix} C(1,1) & C(1,2) \\ C(2,1) & C(2,2) \end{pmatrix}$$

$$x_{t-1} = \begin{pmatrix} GDP(-1) \\ M2(-1) \end{pmatrix}$$

The model can also be written as following:

$$x_t = A_0 + A_1 x_{t-1}$$

A VAR(p) model can be written as following:

$$x_t = A_0 + A_1 x_{t-1} + A_2 x_{t-2} + \dots + A_p x_{t-p}$$

## Model Fitting

VAR model is implemented in Python using statsmodels module. The free code can be found on statsmodels website.

The Python code is the following:

```
>>> import pandas
>>> import numpy as np
>>> import statsmodels.api as sm
>>> import statsmodels.tsa.api as st
>>> mdata = sm.datasets.macrodata.load_pandas().data
>>> dates = mdata[['year', 'quarter']].astype(int).astype(str)
>>> quarterly = dates["year"] + "Q" + dates["quarter"]
>>> from statsmodels.tsa.base.datetools import dates_from_str
>>> quarterly = dates_from_str(quarterly)
>>> mdata = mdata[['realgdp','realcons','realinv']]
>>> mdata.index = pandas.DatetimeIndex(quarterly)
>>> data = np.log(mdata).diff().dropna()
>>> model=st.VAR(data)
>>> results = model.fit(2)
>>> results.summary()
```

Source: [http://statsmodels.sourceforge.net/devel/vector\\_ar.html](http://statsmodels.sourceforge.net/devel/vector_ar.html)

**Summary of Regression Results**

```
=====
Model:           VAR
Method:          OLS
Date:       Mon, 22, Jun, 2015
Time:        15:25:50
-----
No. of Equations:    3.00000   BIC:            -27.5830
Nobs:             200.000   HQIC:           -27.7892
Log likelihood:     1962.57   FPE:            7.42129e-13
AIC:            -27.9293   Det(Omega_mle):  6.69358e-13
```

**Results for equation realgdp**

```
=====
      coefficient      std. error      t-stat      prob
-----
const      0.001527      0.001119      1.365      0.174
L1.realgdp  -0.279435     0.169663     -1.647      0.101
L1.realcons  0.675016     0.131285      5.142      0.000
L1.realinv   0.033219     0.026194      1.268      0.206
L2.realgdp   0.008221     0.173522      0.047      0.962
L2.realcons   0.290458     0.145904      1.991      0.048
L2.realinv   -0.007321     0.025786     -0.284      0.777
=====
```

**Results for equation realcons**

```
=====
      coefficient      std. error      t-stat      prob
-----
const      0.005460      0.000969      5.634      0.000
L1.realgdp  -0.100468     0.146924     -0.684      0.495
L1.realcons  0.268640     0.113690      2.363      0.019
L1.realinv   0.025739     0.022683      1.135      0.258
L2.realgdp   -0.123174     0.150267     -0.820      0.413
L2.realcons   0.232499     0.126350      1.840      0.067
L2.realinv   0.023504     0.022330      1.053      0.294
=====
```

**Results for equation realinv**

```
=====
      coefficient      std. error      t-stat      prob
-----
const      -0.023903     0.005863     -4.077      0.000
L1.realgdp  -1.970974     0.888892     -2.217      0.028
L1.realcons  4.414162     0.687825      6.418      0.000
L1.realinv   0.225479     0.137234      1.643      0.102
L2.realgdp   0.380786     0.909114      0.419      0.676
L2.realcons   0.800281     0.764416      1.047      0.296
L2.realinv   -0.124079     0.135098     -0.918      0.360
=====
```

```
Correlation matrix of residuals
      realgdp  realcons  realinv
realgdp    1.000000  0.603316  0.750722
realcons   0.603316  1.000000  0.131951
realinv    0.750722  0.131951  1.000000
```

```
>>> mdata
      realgdp  realcons  realinv
1959-03-31  2710.349  1707.4  286.898
1959-06-30  2778.801  1733.7  310.859
1959-09-30  2775.488  1751.8  289.226
1959-12-31  2785.204  1753.7  299.356
1960-03-31  2847.699  1770.5  331.722
1960-06-30  2834.390  1792.9  298.152
1960-09-30  2839.022  1785.8  296.375
1960-12-31  2802.616  1788.2  259.764
1961-03-31  2819.264  1787.7  266.405
1961-06-30  2872.005  1814.3  286.246
1961-09-30  2918.419  1823.1  310.227
1961-12-31  2977.830  1859.6  315.463
1962-03-31  3031.241  1879.4  334.271
1962-06-30  3064.709  1902.5  331.039
1962-09-30  3093.047  1917.9  336.962
1962-12-31  3100.563  1945.1  325.650
1963-03-31  3141.087  1958.2  343.721
1963-06-30  3180.447  1976.9  348.730
1963-09-30  3240.332  2003.8  360.102
1963-12-31  3264.967  2020.6  364.534
1964-03-31  3338.246  2060.5  379.523
1964-06-30  3376.587  2096.7  377.778
1964-09-30  3422.469  2135.2  386.754
.....
```

# Forecasting, Impulse Response Analysis and Granger Causality

## Lag Order Selection

Lag order selection is a difficult task in VAR model. In the previous course, there was an example with a VAR(1) model with two macroeconomic variables: GDP and M2. If we consider that the money supply influences GDP with a lag of one quarter then we select lag one in our VAR model. We can use the information-based order criteria to select the optimal number of lags.

## Information Criteria-Based Order Selection

In the previous course, we implemented a VAR model with the following macroeconomic variables:

realgdp – Real GDP  
realcons

- Real consumption realinv
- Real investment

The next step is to determine the optimal number of lags. We can use the following Python code for implementing the information-based order selection.

```
| >>> model.select_order(10)
```

	VAR Order Selection			
	aic	bic	fpe	hqic
0	-27.72	-27.67	9.185e-13	-27.70
1	-28.05*	-27.85*	6.585e-13*	-27.97*
2	-28.05	-27.69	6.591e-13	-27.90
3	-28.04	-27.53	6.629e-13	-27.84
4	-28.03	-27.36	6.747e-13	-27.76
5	-28.02	-27.20	6.803e-13	-27.69
6	-27.97	-27.01	7.111e-13	-27.58
7	-27.94	-26.82	7.362e-13	-27.49
8	-27.95	-26.68	7.310e-13	-27.43
9	-27.97	-26.54	7.172e-13	-27.39
10	-27.93	-26.36	7.458e-13	-27.29

\* Minimum

```
{'fpe': 1, 'hqic': 1, 'bic': 1, 'aic': 1}
```

aic – Akaike Information Criterion

bic – Bayesian Information Criterion

fpe – Final Prediction Error (FPE)

hqic – Hannan-Quinn Information Criterion

Results: fpe, hqic, bic and aic indicate one lag.

## Forecasting

We noticed that a VAR(p) model can be written as following:

$$x_t = A_0 + A_1x_{t-1} + A_2x_{t-2} + \dots + A_p x_{t-p}$$

We know  $x_t, x_{t-1}, x_{t-2}, \dots, x_{t-p}$  values

Our task is to find the values in  $x_{t+1}$  matrix.

The solution is to write VAR as following:

$$x_{t+1} = A_0 + A_1x_t + A_2x_{t-1} + \dots + A_p x_{t-p+1}$$

Therefore, we can find  $x_{t+1}$  from the previous relation.

We can apply this algorithm in order to forecast the macroeconomic variables in our VAR model:

real gdp, real cons and realinv.

## Python Code:

```
lag_order

results.forecast(data.values[-lag_order:], 5) array([[ 0.00502587,
  0.0053712 ,  0.0051154 ],   [ 0.00593683,  0.00784779, -
 0.00302473],

 [ 0.00662889,  0.00764349,  0.00393308],

 [ 0.00731516,  0.00797044,  0.00657495],

 [ 0.00732726,  0.00808811,  0.00649793]])
```

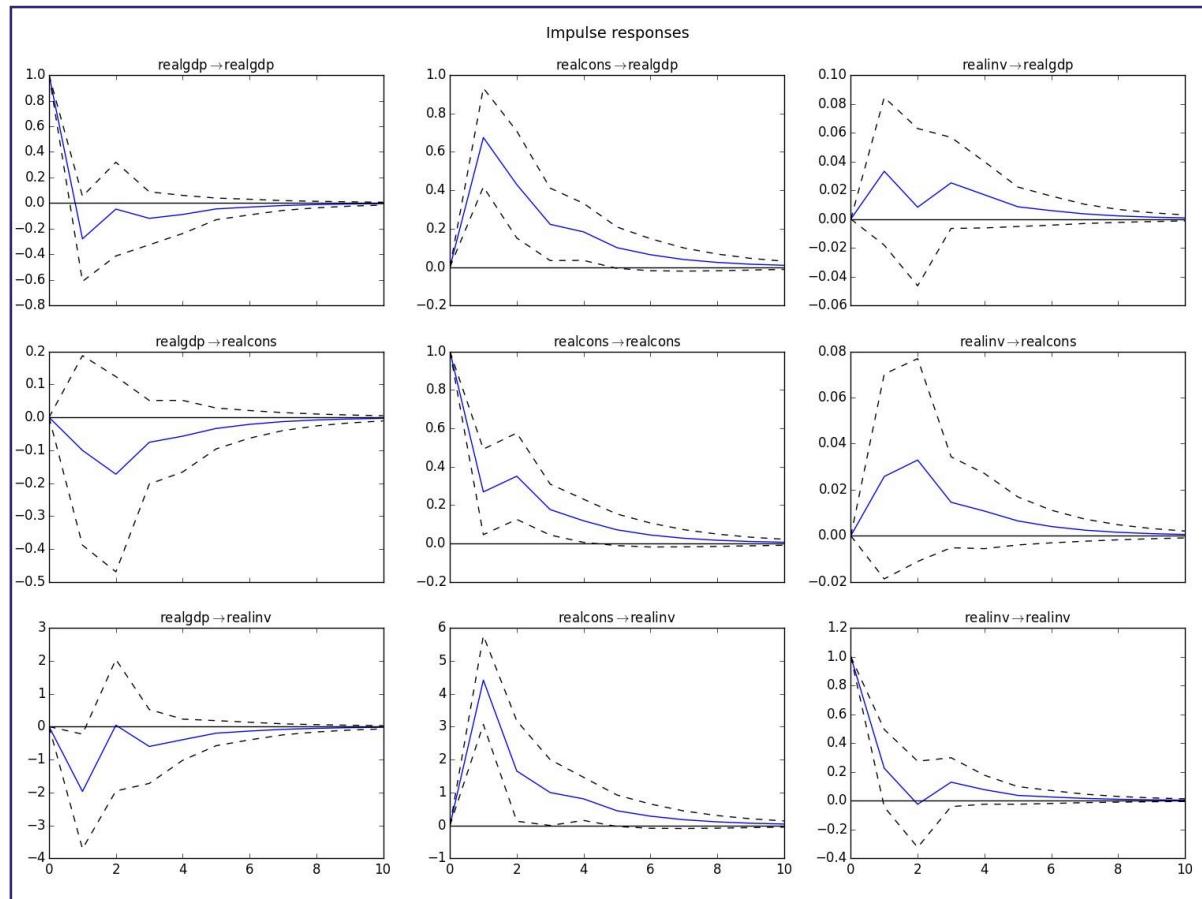
## Impulse Response Analysis

Impulse responses are the estimated responses of an economic variable to a unit impulse in another variable. This can be implemented using the  $M(\infty)$  representation of the VAR(p) process:

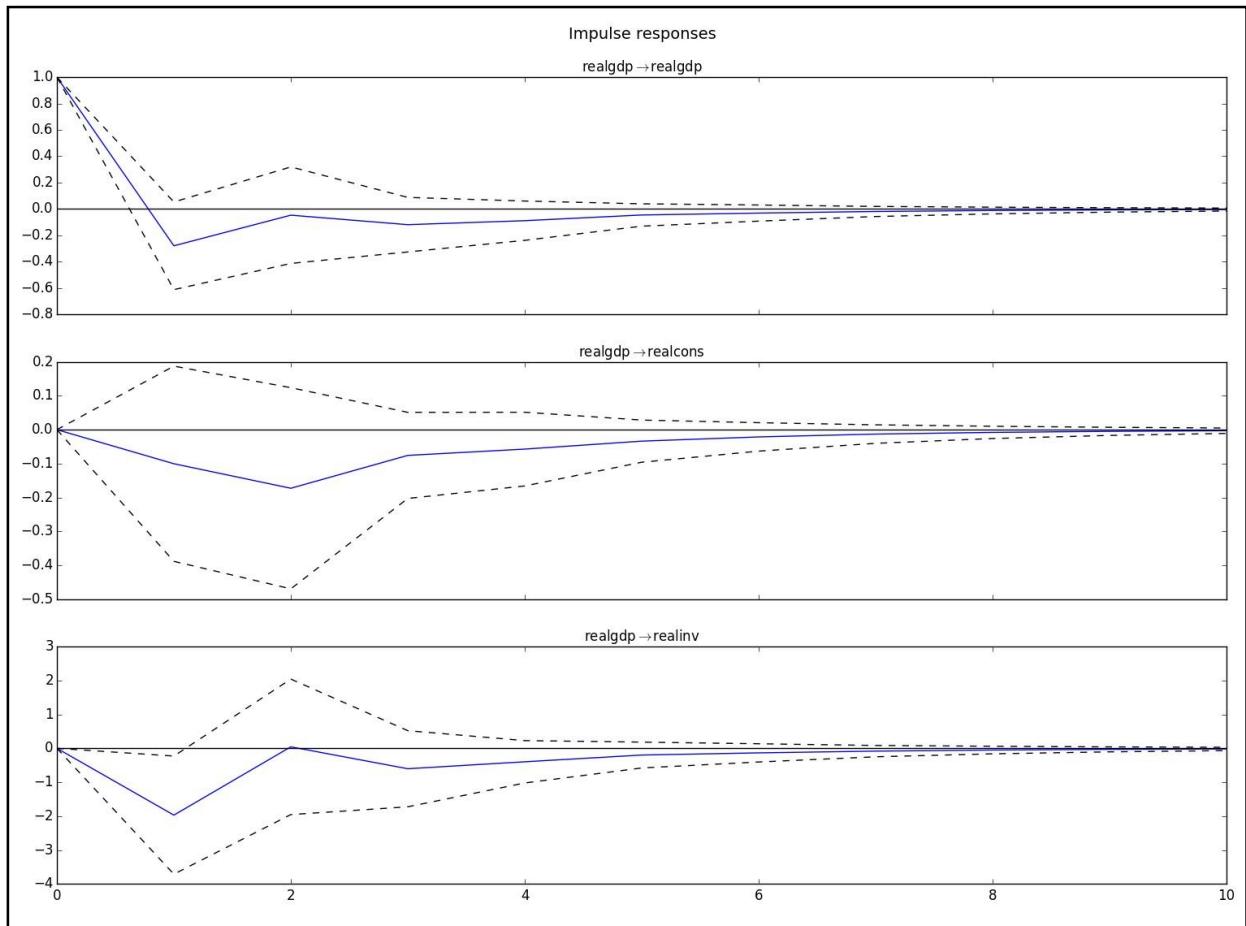
$$x_t = \mu + \sum_{i=1}^{\infty} \Phi_i u_{t-i}$$

We can implement it in Python using irf function as following:

```
irf = results.irf(10)
irf.plot(orth=False)
```



```
irf.plot(impulse='realgdp')
```



## Forecast Error Variance Decomposition (FEVD)

According to Walter Enders “the forecast error variance decomposition (FEVD) tells us the proportion of the movements in a sequence due to its “own” shocks versus shocks to the other variable”.

Forecast errors of component j to k in an i-step ahead forecast can be decomposed using the orthogonalized impulse responses  $\Theta_i$ :

$$\omega_{jk,i} = \sum_{i=0}^{h-1} \frac{(e_j' \Theta_i e_k)^2}{MSE_j(h)}$$

$$MSE_j(h) = \sum_{i=0}^{h-1} e_j' \Phi_i \sum_{u} \Phi_i' e_j$$

These are computed in Python using `fevd` function

```
fevd = results.fevd(5)
```

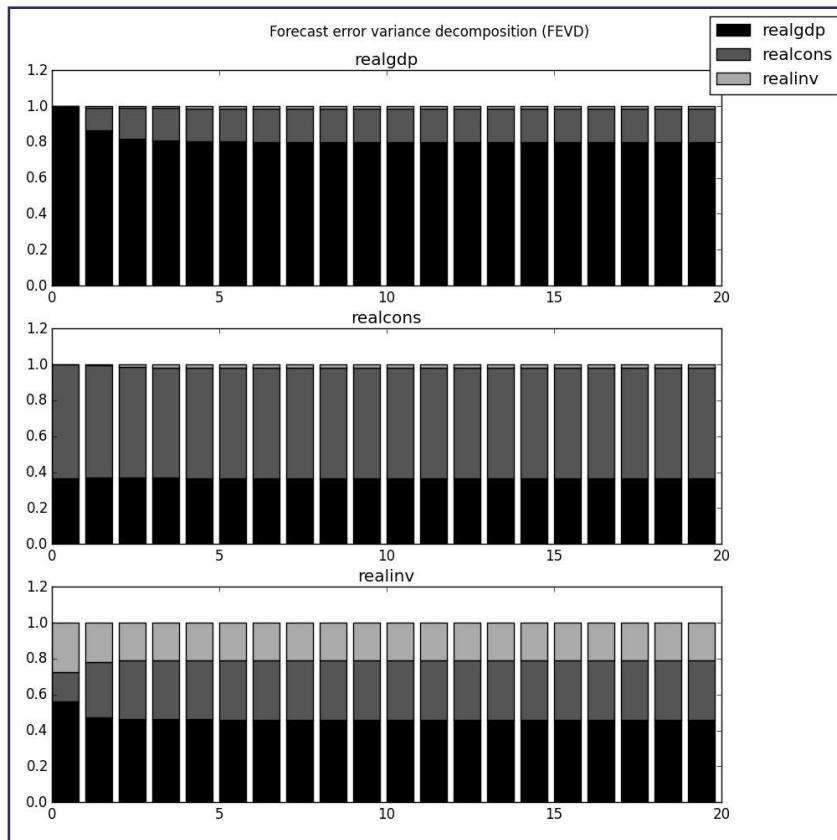
```
fevd.summary()
```

```
FEVD for realgdp
    realgdp  realcons  realinv
0   1.000000  0.000000  0.000000
1   0.863082  0.130030  0.006888
2   0.816610  0.176750  0.006639
3   0.808872  0.181086  0.010042
4   0.803461  0.185049  0.011490

FEVD for realcons
    realgdp  realcons  realinv
0   0.363990  0.636010  0.000000
1   0.369771  0.623928  0.006301
2   0.367706  0.616831  0.015463
3   0.367450  0.615517  0.017033
4   0.367197  0.614903  0.017901

FEVD for realinv
    realgdp  realcons  realinv
0   0.563584  0.161984  0.274432
1   0.471910  0.307875  0.220215
2   0.463240  0.328467  0.208292
3   0.462148  0.328914  0.208938
4   0.461211  0.330359  0.208430
```

```
results.fevd(20).plot()
```



## Granger Causality

Granger causality test is used for determining if one variable is useful in forecasting another.

```
Python code:  
results.test_causality('realgdp', ['realinv', 'realcons'],  
kind='f')
```

```
Granger causality f-test  
=====  
Test statistic Critical Value      p-value      df  
-----  
9.904841      2.387325      0.000 (4, 579L)  
=====  
H_0: ['realinv', 'realcons'] do not Granger-cause realgdp  
Conclusion: reject H_0 at 5.00% significance level  
{'df': (4, 579L), 'crit_value': 2.3873247573799259, 'statistic': 9.9048411456983754, 'signif': 0.05, 'pvalue': 9.3171728876318303e-08, 'conclusion': 'reject'}
```

## Conclusion

Real investments and real consumption Granger cause real GDP. It is a logical conclusion considering that investments and consumption influence GDP evolution.

# Vector Error Correction Models

## Vector Error Correction Models

- Granger and Engle in their *Econometrica* article (1987) showed that two or more integrated, I(1), nonstationary time series might be cointegrated and that some linear combination of these series could be stationary even though each series is not.
- Allows for a wide variety of dynamic patterns in the data.
- Takes a short and long-run view of the model relationship
- Error-correction term is the lagged residuals,  $v(t) = y_{t-1} - \delta_0 - \delta_1 x_{t-1}$  where  $y$  and  $x$  are cointegrated) from the cointegrating regression, of one of the series on the other.

Suppose that inventories,  $Y$ , are proportional to business sales,  $X$ , or

$$Y_t = pX_t \quad (\text{where } p \text{ is the long-run proportionality constant})$$

$$\ln Y_t = y_t = \ln(p) + \ln X_t = k + x_t$$

The dynamic Vector Error Correction model has the form:

$$y_t = \beta_0 + \beta_1 x_2 + \beta_2 x_{t-1} + \alpha y_{t-1} + u_t$$

The VEC model assumes a convergence in the long run such that

$$y_t = y_{t-1} = y^* \text{ and } x_t = x_{t-1} = x^*$$

LR Equilibrium:

$$y^* = \beta_0 + \beta_1 x^* + \beta_2 x^* + \alpha y^*$$

Or

$$(1-\alpha)y^* = \beta_0 + (\beta_1 + \beta_2)x^*$$

$$y^* = \frac{\beta_0}{1-\alpha} + \left[ \frac{\beta_1 + \beta_2}{1-\alpha} \right] x^*$$

Based on the LR proportionality theory above we expect

$$\frac{\beta_1 + \beta_2}{1-\alpha} = 1$$

Define

$$y = 1 - \alpha$$

So:

$$\begin{aligned} y_t &= \beta_0 + \beta_1 x_t + (\gamma - \beta_1)x_{t-1} + (1-\gamma)y_{t-1} + u_t \\ &= \beta_0 + \beta_1 x_t - \beta_1 x_{t-1} + \gamma x_{t-1} + y_{t-1} - \gamma y_{t-1} + u_t \\ &- y_{t-1} = \beta_0 + \beta_1(x_t - x_{t-1}) + \gamma(x_{t-1} - y_{t-1}) + u_t \\ \Delta y_t &= \beta_0 + \beta_1 \Delta x_t + \gamma(\Delta x_{t-1} - \Delta y_{t-1}) + u_t \end{aligned}$$

Becomes the VEC model estimating equation

## Example Python Using Sales/Inventory

```

import pandas as pd
import numpy as np
import pandas as pd
import statsmodels.formula.api as smf
import csv
url = 'c:/users/gib/documents/d5.SALESINVENTORY.csv'
dat = pd.read_csv(url)
results = smf.ols('y ~ x + xminy', data=dat).fit()
print results.summary()

```

```

* : OLS Regression Results
* =====
* Dep. Variable:          y      R-squared:     0.200
* Model:                 OLS    Adj. R-squared:  0.194
* Method:                Least Squares   F-statistic:   33.42
* Date:                  2013-02-12   Prob (F-statistic):  1.12e-13
* Time:                  10:51:40   Log-Likelihood: -2117.
* No. Observations:      270      AIC:         -2117.
* Df Residuals:          267      BIC:         -2106.
* Df Model:               2
* =====
*            coef      std err      t      P>|t|      [95.0% Conf. Int.]
* -----
* Intercept    0.0096    0.001    6.494      0.000      0.007    0.013
* x           0.1697    0.027    6.264      0.000      0.116    0.223
* xminy       0.0237    0.005    5.094      0.000      0.015    0.033
* =====
* Omnibus:            70.974 Durbin-Watson:        0.897
* Prob(Omnibus):      0.000  Jarque-Bera (JB):  166.756
* Skew:              -1.246  Prob(JB):        6.16e-37
* Kurtosis:           5.935  Cond. No.         97.8
* =====

```

## General Case

$$yt = \beta_0 + \beta_1 xt + \beta_2 xt-1 + \alpha yt-1 + ut$$

Subtract  $y_{t-1}$  from the RHS and LHS of the equation

$$\Delta yt = yt - y_{t-1} = \beta_0 + \beta_1 xt + \beta_2 xt-1 - (1 - \alpha)yt-1 + ut$$

Add and subtract  $\beta_1 x_{t-1}$  on the RHS:

$$\Delta yt = \beta_0 + \beta_1 xt - \beta_1 xt-1 + \beta_1 xt-1 + \beta_2 xt-1 - (1 - \alpha)yt-1 + ut$$

$$\Delta yt = \beta_0 + \beta_1 \Delta xt + (\beta_1 + \beta_2)xt-1 - (1 - \alpha)yt-1 + ut$$

**Note:**  $\Delta y_t$  and  $\Delta x_t$  are stationary [(0)] if  $y_t$  and  $x_t$  are [ $I(1)$ ] nonstationary.

Rewrite as follows:

$$\Delta yt = \beta_0 + \beta_1 \Delta xt - \lambda(yt-1 - \delta_0 - \delta_1 xt-1) + ut$$

In parenthesis is the long-run cointegrated relationship between x and y.

Where  $\lambda = 1 - \alpha$  and  $\delta_1 = \frac{\beta_1 + \beta_2}{1 - \alpha}$

## General Example and How to Get Residuals in Python

```

import numpy as np
import pandas as pd
import statsmodels.formula.api as smf
import csv
url = 'c:/users/gib/documents/d5.SALESINVENTORY.csv'
dat = pd.read_csv(url)
results = smf.ols('laginvent ~ lagsales', data=dat).fit()
residua=results.resid
residua=np.append(0,residua)
results = smf.ols('y ~ x + residua', data=dat).fit()

```

## General Example Using Inventory Data

```

■ print results.summary()
■               OLS Regression Results
■ =====
■ Dep. Variable:          y                      R-squared:      0.394
■ Model:                 OLS                     Adj. R-squared:  0.390
■ Method:                Least Squares          F-statistic:   86.95
■ Date:                  16:11:11                Prob (F-statistic): 8.30e-30
■ Time:                  16:11:11                Log-Likelihood: 1099.1
■ No. Observations:      270                   AIC:            -2192.
■ Df Residuals:          267                   BIC:            -2181.
■ Df Model:              2
■ =====
■             coef    std err     t   P>|t|   [95.0% Conf. Int.]
■ -----
■ Intercept   0.0024    0.000    9.144    0.000    0.002    0.003
■ x           0.1131    0.024    4.673    0.000    0.065    0.161
■ residua    -0.1077   0.010   -10.951   0.000   -0.127   -0.088
■ =====
■ Omnibus:            36.496  Durbin-Watson:    0.893
■ Prob(Omnibus):       0.000  Jarque-Bera (JB): 53.657
■ Skew:                -0.835  Prob(JB):        2.23e-12
■ Kurtosis:             4.407  Cond. No.         96.3
■ =====

```

## Food for Thought

- The P and E series are often thought to be related (“associated”). Do you think cointegration exists? Evaluate the P/E during a bubble.
- Again, consider cointegration in the context of multi dwelling housing. Rents and dwelling prices might seem to be cointegrated. What happens in a boom when a lot of new structures are built?

# Filtering and Smoothing

## Filtering and Smoothing

Filtering is an iterative process that enables us to estimate a model's parameters. The parameters rely upon a large quantity of observable and unobservable data. A time series of observable data called  $z_k$  (stock prices, interest rates, futures prices) exists in models with unobservable time series  $x_k$  (volatility, correlation, convenience yield).

This allows us to construct an algorithm containing a transition equation linking two consecutive unobservable states and a measurement equation relating the observed data to this hidden state.

Two steps are involved:

1. First estimate the hidden state by using all the information prior to that time step.
2. Using the predicted value together with the new observation we obtain a conditional a posteriori estimation of the state.

There are a variety of types of smoothing and filtering techniques:

- Additive smoothing
- Butterworth filter
- Digital filter
- Kalman filter
- Laplacian smoothing ↳ Loss-pass filter
- Savitzky-Golay Filter
- Smoothing filter
- Local Regression Smoothing
- Ramer-Douglas-Peucker Algorithm
- Moving average
- Exponential smoothing

## Kalman Filter

Kalman Filter model assumes that the state of a system of time t evolves from the prior state at time t-1 according to the equation:

$$x_t = F_t \times x_{t-1} + B_t \times u_t + w_t$$

$x_t$  – state vector containing the terms of interest for the system (volatility of returns) at time t

$u_t$  – the vector containing any control inputs (steering angle for example at physics)

$F_t$  – state transition matrix which applies the effect of each system state parameter at time t-1 on the system state at time t (example the position and velocity at time t-1 both affect the position at time t)

$B_t$  – the control-input matrix which applies the effect of each control input parameter in the vector  $u_t$  on the state vector

$w_t$  – the vector containing the process noise terms for each parameters in the state vector  $w_t \sim (0, Q_t)$

An observation (or measurement)  $z_t$  of the true state  $x_t$  is made according to

$$z_t = H x_t + v_t$$

Where  $H_t$  is the observation model which maps the true state space into the observed space and  $v_t$  is the observation noise which is assumed to be zero mean Gaussian white noise with covariance  $R_t$

$$v_t \sim (0, R_t)$$

### Python Code:

```
# Kalman filter example demo in Python

# A Python implementation of the example given in pages 11-15 of "An

# Introduction to the Kalman Filter" by Greg Welch and Gary Bishop,

# University of North Carolina at Chapel Hill, Department of Computer

# Science, TR 95-041,

# http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html

# by Andrew D. Straw
```

```

import numpy
import pylab
# intial parameters n_iter = 50
sz = (n_iter,) # size of array
x = -0.37727 # truth value (typo in example at top of p. 13 calls this z)
z = numpy.random.normal(x,0.1,size=sz) # observations (normal about x,
sigma=0.1)

Q= 1e-5 # process variance

# allocate space for arrays xhat=numpy.zeros(sz)    # a posteri estimate of x
P=numpy.zeros(sz)      # a posteri error estimate xhatminus=numpy.zeros(sz)
# a priori estimate of x Pminus=numpy.zeros(sz)    # a priori error estimate
K=numpy.zeros(sz)      # gain or blending factor

= 0.1**2 # estimate of measurement variance, change to see effect

# intial guesses xhat[0] = 0.0
P[0] = 1.0

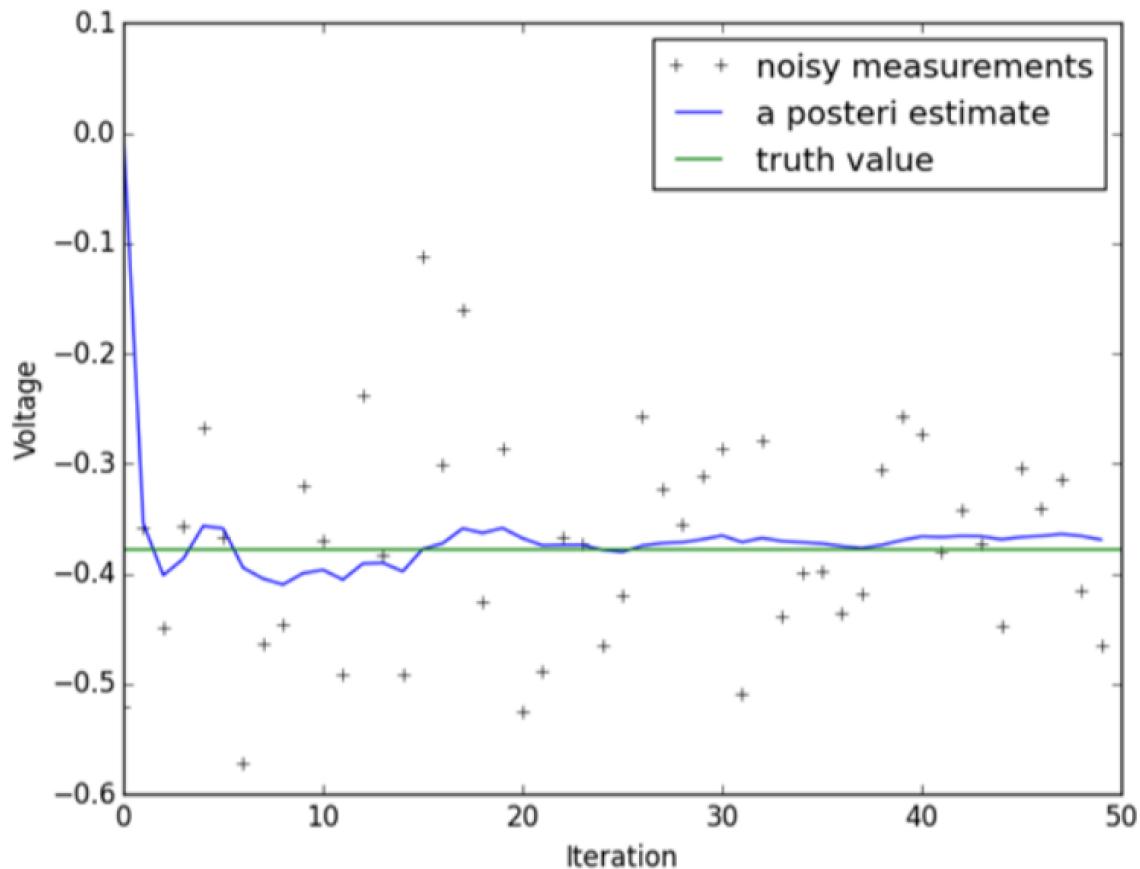
for k in range(1,n_iter):  # time update  xhatminus[k] = xhat[k-1]
Pminus[k] = P[k-1]+Q

# measurement update
K[k] = Pminus[k]/( Pminus[k]+R )
xhat[k] = xhatminus[k]+K[k]*(z[k]-xhatminus[k])
P[k] = (1-K[k])*Pminus[k]

pylab.figure() pylab.plot(z,'k+',label='noisy measurements') pylab.plot(xhat,'b-',
',label='a posteri estimate') pylab.axhline(x,color='g',label='truth value')
pylab.legend() pylab.xlabel('Iteration')
pylab.ylabel('Voltage')

pylab.figure()
valid_iter = range(1,n_iter) # Pminus not valid at step 0
pylab.plot(valid_iter,Pminus[valid_iter],label='a priori error estimate')
pylab.xlabel('Iteration') pylab.ylabel('$\$(Voltage)^2$')
pylab.setp(pylab.gca(),'ylim',[0,.01])
pylab.show()

```



Source: <http://wiki.scipy.org/Cookbook/KalmanFiltering>

# ARCH Model

## ARCH(p)

- **Autoregressive (AR):** tomorrow's variance (or volatility) is a regressed function of today's variance – it regresses on itself
- **Conditional (C):** tomorrow's variance depends – is conditional on – the most recent variance. An unconditional variance would not depend on today's variance
- **Heteroskedastic (H):** variances are not constant, they flux over time

The ARCH(p) process captures the conditional heteroscedasticity as a weighted average of past squared unexpected returns:

$$\sigma^2_t = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \dots + \alpha_p \varepsilon_{t-p}^2$$

$\alpha_0 > 0, \alpha_1, \dots, \alpha_p \geq 0 \quad \varepsilon_t | I_t \sim N(0, \sigma^2_t)$

ARCH models are rarely used in finance as simple GARCH models perform so much better.

## Simulating an ARCH(1) Process

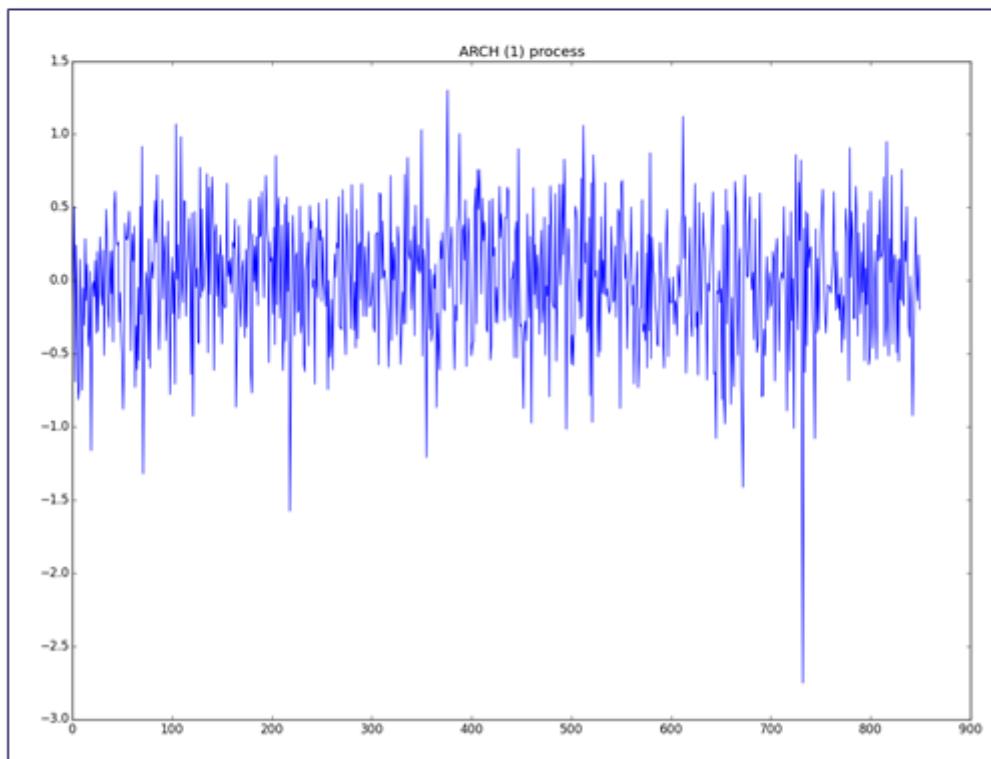
The equation for an ARCH(1) process is the following:

$$\sigma^2_t = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2$$

$\sigma^2_t$  – variance at time t  
 $\alpha_0, \alpha_1$  - coefficients  
 $\varepsilon_t | I_t$  – the squared error term for t-1 period

The Python code is the following:

```
>>> from matplotlib.pyplot import *
>>> import scipy as sp
>>> v=850 # v is the number of observations
>>> v1=250 # we need to drop the first several observations
>>> v2=v+v1 # sum of two numbers
>>> alpha=(0.15,0.25) # ARCH (1) coefficients alpha0 and
alpha1, see Equation
>>> errors=sp.random.normal(0,1,v2)
>>> t1=sp.zeros(v2)
>>> t1[0]=sp.random.normal(0,sp.sqrt(alpha[0]/(1-alpha[1])),1)
>>> for i in range(1,v2-1):
...   t1[i]=errors[i]*sp.sqrt(alpha[0]+alpha[1]*t1[i-1]**2)
>>> y=t1[v1-1:-1] # drop the first v1 observations
>>> title('ARCH (1) process')
>>> x=range(v)
>>> plot(x,y)
```



# Correlation

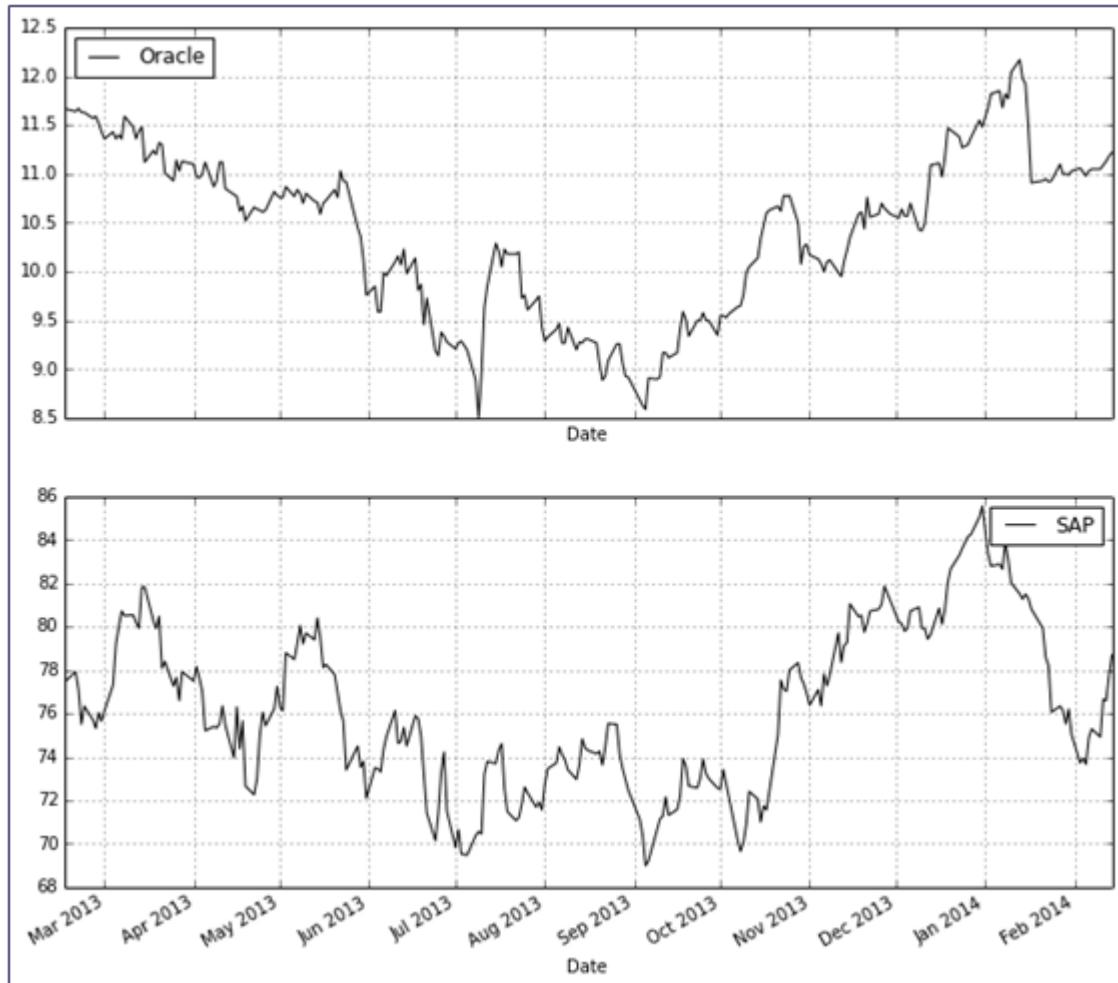
## Dynamic Correlation Models

- Dynamic models account for time-dependent changes for any given state in a system.
- Steady-state or static models are time-invariant and calculate the system in equilibrium.
- Correlation modeling (Pearson, Spearman, or Kendall) views the co-movement between the changes in two arrays of data such as time-series arrays of two sets of stock-price data.

Plotting correlation of two stocks (SAP and ORC) in Python:

```
import pandas as pd
import pandas.io.data as web
import datetime
%pylab inline

oracle_data = web.get_data_yahoo('ORC', '2013-02-15','2014-02-15')
sap_data = web.get_data_yahoo('SAP', '2013-02-15','2014-02-15')
oracle = oracle_data['Adj Close']
sap = sap_data['Adj Close']
df = pd.concat([oracle,sap],axis=1)
df.columns =[['Oracle','SAP']]
plt.rcParams['figure.figsize'] = 11, 10
df.plot(subplots = True); plt.legend(loc='upper right')
```



Correlation may only be meaningfully computed for stationary processes.

Covariance stationarity for a time series,  $y_t$ , is defined as:

- Constant, finite mean
- Constant, finite variance
- Covariance  $(y_t, y_{t-s})$  depends only on the lags

For financial data, this implies that correlation is only meaningful for variates such as rates of return or normally transformed variates,  $z$ , such that:

$$z = (x - \mu)/\sigma$$

Where  $x$  is non-stationary and  $\mu$  is the mean of  $x$  and  $\sigma$  the standard deviation.

For non-stationary variates like prices, correlation is not usually meaningful.

A more coherent measure of relatedness is cointegration. Cointegration uses a two-step process:

- Long-term equilibrium relationships are established
- A dynamic correlation of returns is estimated

Cointegration will not be discussed in these ERM sessions, however, it is very important in developing dynamic hedges that seek to keep stationary tracking error within preset bounds. Hedging using correlation measures typically is not able to achieve such control.

However, instantaneous and terminal measures of correlation are used in various applications such as developing stochastic interest rate generators.

## Definitions of Correlation

### Pearson's Correlation Formula

Linear relationships between variables can be quantified using the Pearson Product-Moment Correlation Coefficient, or

$$r_{XY} = \frac{\sum(X - \bar{X})(Y - \bar{Y})}{nS_X S_Y}$$

The value of this statistic is always between -1 and 1, and if X and Y are unrelated it will equal zero.

### Spearman's Correlation Method

A nonparametric (distribution-free) rank statistic proposed by Spearman in 1904 as a measure of the strength of the associations between two variables (Lehmann and D'Abrera 1998). The Spearman rank correlation coefficient can be used to give an  $R$ -estimate, and is a measure of monotone association that is used when the distribution of the data makes Pearson's correlation coefficient undesirable or misleading.

The Spearman rank correlation coefficient is defined by

$$r' \equiv 1 - 6 \sum \frac{d^2}{N(N^2 - 1)}, \quad (1)$$

Where  $d$  is the difference in statistical rank of corresponding variables, and is an approximation to the exact correlation coefficient

$$r \equiv \frac{\sum xy}{\sqrt{\sum x^2 \sum y^2}} \quad (2)$$

Computed from the original data. Because it uses ranks, the Spearman rank correlation coefficient is much easier to compute.

The variance, kurtosis, and higher-order moments are

$$\sigma^2 = \frac{1}{N - 1} \quad (3)$$

$$\gamma_2 = -\frac{114}{25N} - \frac{6}{5N^2} - \dots \quad (4)$$

$$\gamma_3 = \gamma_5 = \dots = 0. \quad (5)$$

Student was the first to obtain the variance.

## The Simple Formula for $r_s$ , for Rankings without Ties

Here is the same table you saw above, except now we also take the difference between each pair of ranks ( $D = X - Y$ ), and then the square of each difference. All that is required for the calculation of the Spearman coefficient are the values of  $N$  and  $\Sigma D^2$ , according to the formula:

wine	X	Y	D	$D^2$
a	1	2	-1	1
b	2	1	1	1
c	3	5	-2	4
d	4	3	1	1
e	5	4	1	1
f	6	7	-1	1
g	7	8	-1	1
h	8	6	2	4
$N = 8$		$\Sigma D^2 = 14$		

$$r_s = 1 - \frac{6 \sum D^2}{N(N^2 - 1)}$$

There is no generally accepted method for computing the standard error for small samples.

## Kendall's Tau Coefficient

Spearman's  $r$  treats rank as scores and computes the correlation between two sets of ranks. Kendall's tau is based on the number of inversions in rankings.

Although there is evidence that Kendall's Tau holds up better than Pearson's  $r$  to extreme nonnormality in the data that seems to be true only at quite extreme levels.

Let  $\text{inv} :=$  number of inversions, i.e. reversals of pair-wise rank orders between  $n$  pairs. Equal rankings need an adjustment.

$$\text{Kendall's Tau} = 1 - \frac{2 * \text{inv}}{\text{number of pairs of objects}}$$

$$= 1 - \frac{2 * \text{inv}}{(n * (n-1)/2)} = 1 - \frac{4 * \text{inv}}{(n * (n-1))}$$

The preceding correlation definitions weight all observations equally. Hence, clustering of data associated with periods of high and low volatility is not reflected. High volatility periods often display higher levels correlation risk. For example, when the Russian Rouble became distressed and highly volatile in 1998, the third world debt market experienced contagion.

The Spearman and Kendall Tau measures are more appropriate for non-normal distributions.

The above definitions of correlation are absolute and will only accidentally replicate implied market correlations. Bayesian or conditional definitions of correlation are more appropriate for time series and stochastic models as relationships such as volatility smiles may be captured.

## Instantaneous Correlation

$\rho_{ij}(u) :=$  instantaneous correlation between variables with indices i,j at moment u. (see Rebonato, Modern Pricing of Interest-Rate Derivatives, 2002)

Instantaneous correlation in a hedging model may be based on forecasts of future correlation. An empirical function relating the value of a position to the associated correlation may be derived. The empirical function is then used to vary instantaneous correlation as the underlying value of the position changes. Instantaneous correlations may then be used to compute the associated terminal correlation.

## Terminal Correlation

$$\text{term\_}\rho_{ij}(T) = [\int_0^T \sigma_i(u) \sigma_j(u) \rho_{ij}(u) du] / \sqrt{(v_i v_j)}$$

where:  $v_i = \int_0^T \sigma_i(u)^2 du$

Terminal correlation is usually the more germane measure of correlation for pricing and hedging. (see Rebonato, Modern Pricing of Interest-Rate Derivatives, 2002)

## Implied Correlation

The implied volatility on the cross rate on a foreign exchange process may be expressed as:

$$\sigma_{x-y}^2 = \sigma_x^2 + \sigma_y^2 - 2\sigma_x\sigma_y\rho$$

which may be solved options are available on the f/x rates and cross, i.e.:

$$\rho = (\sigma_x^2 + \sigma_y^2 - \sigma_{x-y}^2) / (2\sigma_x\sigma_y)$$

In the above example,  $\rho$  is the implied correlation. (see Alexander, *Market Models*, 2001)

## Confidence Intervals for Correlation Point Estimates: Fisher-z Transformation

Let  $\rho$  be a point estimate of the correlation  $r$ , define the transformation  $z$ :

$$z = \frac{1}{2} \ln[(1+\rho)/(1-\rho)]$$

When the sample is large (25 or more is a useful rule of thumb), the distribution  $z$  is approximately normal with an approximate mean and variance:

$$E(z) = \frac{1}{2} \ln[(1+\rho)/(1-\rho)] \text{ and } \sigma(z) = 1/\sqrt{(n-3)}$$

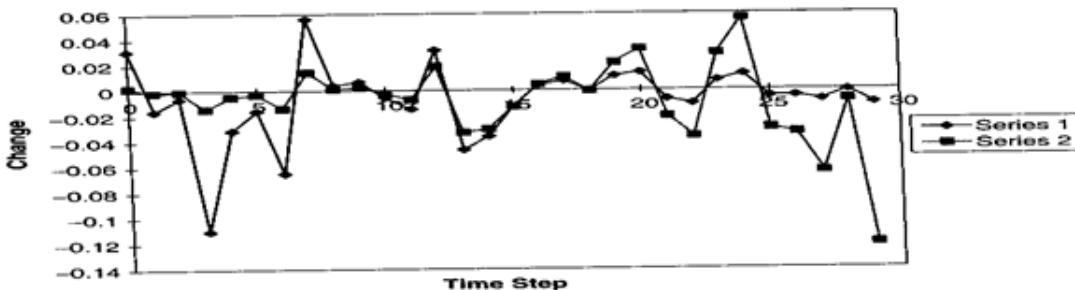
The standardized statistic,  $(z - E(z))/\sigma(z)$ , is approximately a standard normal variable. Therefore, approximate  $1 - \alpha$  confidence limits for  $z$  are:  $E(z) \pm (1 - \alpha/2)\sigma(z)$ . The  $1 - \alpha$  limits for  $\rho$  are then obtained by transforming the limits on  $z$  by means of:

$$\rho = (\exp(2z) - 1)/(\exp(2z) + 1)$$

## Relationship Between Correlation and Volatility

In *Volatility and Correlation in Option Pricing*, 1999, in the context of two imperfectly correlated variables, Ricardo Rebonato states,

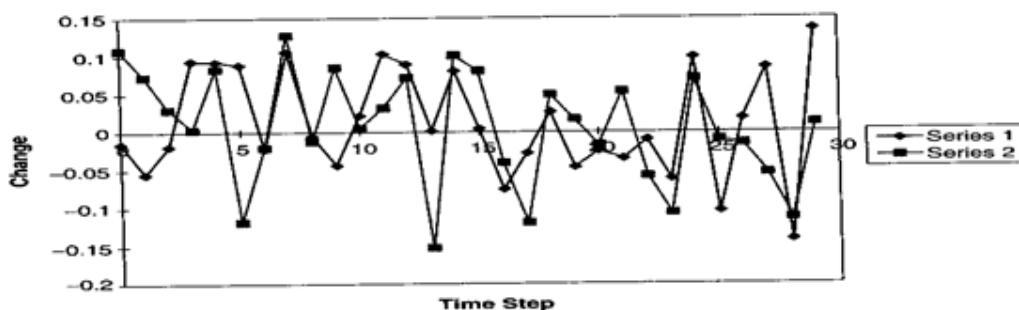
“Under these assumptions we can now run two simulations, one with a constant, identical volatility for both variables and with imperfect correlation, and the other with different instantaneous imperfect correlation, and the other with instantaneous volatilities ...but perfect correlation. One can then evaluate correlation, calculated along the path, between the changes in the log of the two variables in the two cases. ...As is apparent from the two figures, the same sample correlation can be obtained despite the fact that the two de-correlation-generating mechanisms are very different.”



**Figure 3.1** Changes in the variables  $x_1$  and  $x_2$ . The two variables were subjected to the same random shocks (instantaneous correlation = 1). The first variable (Series 1) had an instantaneous volatility given by  $\sigma_1(t) = \sigma_0 \exp(-\nu t)$ ,  $0 \leq t \leq T$ , with  $\sigma_0 = 20\%$  and  $\nu = 0.64$  and  $T = 4$  years. The second variable (Series 2) had an instantaneous volatility given by  $\sigma_2(t) = \sigma_0 \exp(-\nu(T-t))$ ,  $0 \leq t \leq T$ . The empirical sample correlation turned out to be 34.89%

54

#### Volatility and Correlation

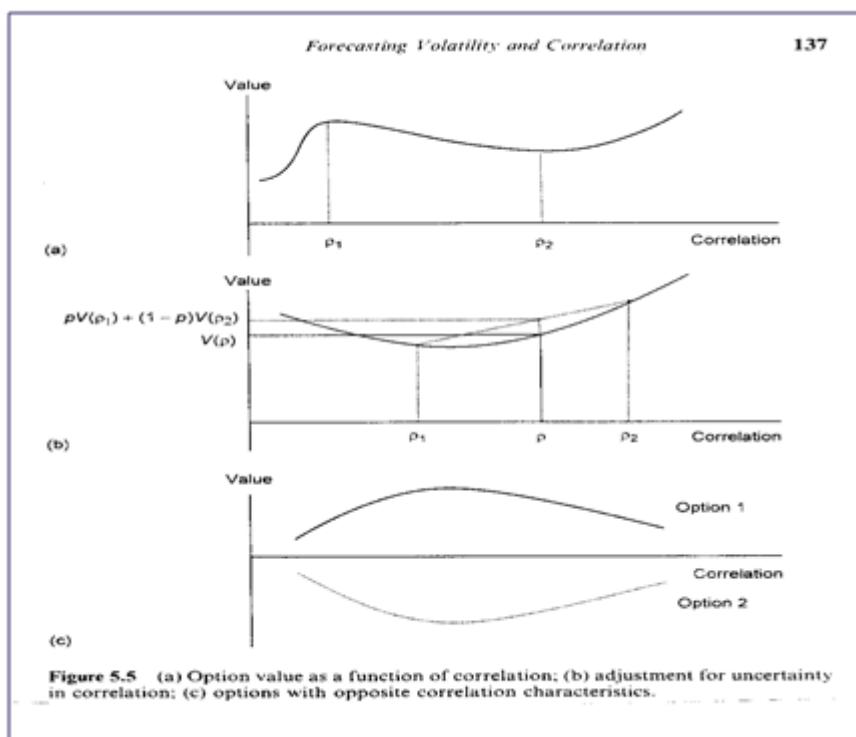


**Figure 3.2** Changes in the variables  $x_1$  and  $x_2$ . The two variables were subjected to different random shocks (instantaneous correlation = 35.00%). Both variables had the same constant instantaneous volatility of  $\sigma_0 = 20\%$ . The empirical sample correlation turned out to be 34.89%

In *Market Models*, 2001, Carol Alexander observes,

“Unlike prices, volatility and correlation are not directly observable in the market. They can only be estimated in the context of a model. It is important to understand that implied and statistical volatility models provide estimates or forecasts of the same thing—that is, the volatility parameter in some assumed underlying price process.

The following empirically derived graphs show instantaneous correlations as a function of option value. These relationships allow forecast and simulation processes to vary correlation conditional on the level of the option price.



# GARCH Model

## GARCH Model

The GARCH model is an extension of the autoregressive conditional heteroscedasticity (ARCH) model developed by Engle in 1982. The acronym "GARCH" means "generalized autoregressive condition heteroscedasticity" model.

We model the log-return time series by:

$$\varepsilon_t = \sigma_t z_t$$

where  $z_t$  is an independent, identically distributed sequence of properly scaled Gaussian random numbers. The variance is dynamic and is governed by the equation

$$\sigma^2_t = a\varepsilon_{t-1}^2 + b\sigma_{t-1}^2$$

We refer to  $a$  as the "state memory factor" and as the "variance memory factor". If  $a=0$  and  $b=1$ , then the variance does not change and we obtain a discrete white noise. Note that you also have control over the initial variance  $\sigma_0^2$ .

We make three views available to the user. The first view features the volatility series  $\sigma$ . The most important feature of a GARCH model is the non-constant volatility series. Notice, e.g. the first snapshot, that making the variance memory factor too small causes the volatility series to tend to zero—which produces an unrealistic model of a real asset's returns.

The second view features the log-return series. This series is used to construct the asset price in the third view. Again, notice that a poor choice of parameters—making the state memory or variance memory factor too large—generally causes the variance to explode.

GARCH models are generally quite sensitive to parameter choices. This sensitivity is problematic when estimating GARCH parameters from real data.

Instead of having the variance of today's return depend on yesterday's variance and yesterday's squared return, we could have allowed today's return to depend on the variance and squared returns from multiple prior days. In general, if the process depends on the past p days' squared returns and the past q days' variances, the process is called a GARCH(p,q) process. For sake of simplicity, we simulate only the log-returns and associated asset price of a GARCH(1,1) process.

GARCH regresses on “lagged” or historical terms. The lagged terms are either variance or squared returns. The generic GARCH(p,q) model regresses on (p) squared returns and (q) variances. Therefore,

GARCH(1,1) “lags” or regresses on last period’s squared return (i.e., just 1 return) and last period’s variance (i.e., just 1 variance).

GARCH(1,1) given by the following equation:

$$\sigma^2_t = a + b\sigma^2_{t-1} + c\epsilon^2_{t-1}$$

## Types of GARCH Models

### I-GARCH

$\alpha + \beta < 1$  if the returns process is to be stationary. It is necessary for mean reverting processes.

If  $\alpha + \beta = 1$ , the return process is a random walk, and the GARCH(1,1) process may be expressed as:

$$\sigma^2_t = \omega + (1 - \lambda)$$

$$\varepsilon^2_{t-1} + \lambda \sigma^2_{t-1} \lambda =$$

$$\beta, 1 \geq \lambda \geq 0$$

Such models are called *integrated* GARCH or I-GARCH models. I-GARCH models are often appropriate in foreign exchange markets.

When  $\omega = 0$ , the I-GARCH model becomes an EWMA (exponentially weighted moving average) model.

### A-GARCH: Asymmetric GARCH

- Equity markets are typically more volatile in falling markets than rising markets.
- A-GARCH models were designed to fit such markets.

## E-GARCH: Exponential GARCH

The first A-GARCH model was the E-GARCH or Exponential GARCH model, of the following form:

$$\ln \sigma_t^2 = \omega + g(z_{t-1}) + \beta \ln \sigma_{t-1}^2$$

where  $g(*)$  is an asymmetric response function defined by:

$$g(z_t) = \lambda z_t + \phi(|z_t| - \sqrt{2\pi})$$

$z_t$  is the standard normal unexpected return  $\varepsilon_t/\sigma_t$

## N-GARCH: Non-linear GARCH

$$\begin{aligned} r_t &= r - .5 \sigma_t^2 + \sigma_t \xi_t \sigma_{2t} \\ &= \omega + \alpha \sigma_{2t-1} (\xi_{t-1} - \theta - \\ &\quad \lambda) 2 + \beta \sigma_{t-1} \xi_t = \varepsilon_t + \lambda \end{aligned}$$

N-GARCH models are typically not risk-neutral, although local risk-neutrality may exist. Models that are not risk-neutral are incomplete and generally do not have replicating portfolios everywhere available. Hence, hedging may not be possible. Typically, N-GARCH models are more complex than the preceding GARCH models in this presentation. For these reasons N-GARCH models are not currently used as extensively in finance as other GARCH models.

## High-Frequency GARCH Models

Model frequency is a function of the time interval associated with return observations. So hourly observation is more frequent than daily and daily is more frequent than weekly, and so on.

Most GARCH models in practice assume that unexpected portion of the actual return is conditionally normally distributed. Non-normal models are characterized by fat tailed distributions for the unexpected portion of the return. **t-GARCH** models develop the unexpected portion of the return according to a student-t distribution. Non-normal models may be developed by modifying the likelihood function used to estimate the model parameters.

High frequency models, especially intra-day models, are much more likely to be more volatile and fat tailed than lower frequency models.

# Multivariate GARCH

## Multivariate GARCH

The general multivariate GARCH model has the form:

$$y_t = c + \sum_{i=1}^r \Phi_i y_{t-i} + \sum_{l=1}^L \beta_l x_{t-l} + \sum_{s=1}^S \Theta_s \epsilon_{t-s}$$

$y_t$  -  $k \times 1$  matrix

$\Phi_i$  -  $k \times k$  matrix

$\beta_l$  -  $k \times m$  matrix

$x_{t-l}$  -  $m \times 1$  matrix

$\Theta_s$  -  $k \times k$  matrix

$$\epsilon_t = \Sigma_t^{1/2} z_t$$

$$z_t \sim iid(0, I_k)$$

$$\Sigma_t = \sum_t^{1/2} \sum_t^{1/2} \square$$

$$\sum_t^{1/2} = Cholesky\ factor$$

Here

$$var(y_t | I_{t-1}) = \sum_t^{1/2} var(z_t | I_{t-1}) \sum_t^{1/2} = \sum_t \square$$

## Multivariate GARCH Prediction

- Predictions from multivariate GARCH models can be generated in a similar fashion to predictions from univariate GARCH models
- For multivariate GARCH models, predictions can be generated for both the levels of the original multivariate time series and its conditional covariance matrix. Predictions of the levels are obtained just as for vector autoregressive (VAR) models. Compared with VAR models, the predictions of the conditional covariance matrix from multivariate GARCH models can be used to construct more reliable confidence intervals for predictions of the levels.

# Exponentially Weighted Moving Average (EWMA)

## EWMA

Volatility: “A statistical measure of the dispersion of returns for a given security or market index. Volatility can either be measured by using standard deviation or variance between returns from that same security or market index.

Commonly, the higher the volatility, the riskier the security.” (Investopedia)

Earlier we calculated simple volatility. EWMA is a more sophisticated measure of volatility of returns. Instead of giving equal weight to all of the squared returns in a series under study, squared returns are weighted to give more recent observations more importance.

EWMA is often used to estimate the next-day (or period) volatility (square root of the variance) of a returns time series and track the volatility as it changes.

The formula for the EWMA variance of a return at time t is:

## EWMA:

$$\sigma_t^2 = \lambda\sigma_{t-1}^2 + (1 - \lambda)r_{t-1}^2$$

For the beginning of a financial series:

Start:

$$\sigma_1^2 = 0 + (1-\lambda)r_0^2$$

Second:

$$\sigma_2^2 = \lambda\sigma_1^2 + (1-\lambda)r_1^2 = \lambda[(1-\lambda)r_0^2] + (1-\lambda)r_1^2$$

Third:

$$\begin{aligned}\sigma_3^2 &= \lambda\sigma_2^2 + (1-\lambda)r_2^2 = \lambda\{\lambda[(1-\lambda)r_0^2] + (1-\lambda)r_1^2\} + (1-\lambda)r_2^2 = \\ &\quad \lambda^2(1-\lambda)r_0^2 + \lambda(1-\lambda)r_1^2 + \lambda^0(1-\lambda)r_2^2\end{aligned}$$

T:

$$\sigma_t^2 = \left(\frac{1}{N}\right) \sum_{t=1}^N (1 - \lambda)\lambda^{t-1} r_{t-1}^2$$

where N is the number of observations

Where  $(1 - \lambda)\lambda^{t-1}$  represents the weight given at each observation with older observations receiving less weight. Lambda is often assumed to be 0.94. The variance in the return depends on the variance in the last period with weight lambda ( $\lambda$ ) plus a factor that updates information which is the square of last period's return weighted by  $1 - \lambda$

Microsoft Excel non-commercial use

date	Goog P	lag	ret	x					I	J	K	L	M
30-Jun-14	578.66	577.18	0.002561	6.56E-06	0	0.06	3.94E-07	2.66E-06					
27-Jun-14	577.18	581	-0.0066	4.35E-05	1	0.0564	2.45E-06	2.67E-06					
26-Jun-14	581	565.26	0.027465	0.000754	2	0.053016	4.00E-05	2.05E-06					
25-Jun-14	565.26	565.19	0.000124	1.53E-08	3	0.049835	7.62E-10	2.08E-06					
24-Jun-14	565.19	555.15	0.017924	0.000321	4	0.046845	1.50E-05	1.86E-06					
23-Jun-14	555.15	556.85	-0.00306	9.35E-06	5	0.044034	4.12E-07	1.89E-06					
20-Jun-14	556.85	554.24	0.004698	2.21E-05	6	0.041392	9.15E-07	1.90E-06					
19-Jun-14	554.24	544.86	0.017069	0.000291	7	0.038909	1.13E-05	1.73E-06					
18-Jun-14	544.86	544.2	0.001212	1.47E-06	8	0.036574	5.38E-08	1.76E-06					
17-Jun-14	544.2	549.26	-0.00926	8.57E-05	9	0.03438	2.95E-06	1.74E-06					
16-Jun-14	549.26	552.26	-0.00545	2.97E-05	10	0.032317	9.80E-07	1.75E-06					
13-Jun-14	552.26	557.3	-0.00909	8.25E-05	11	0.030379	2.51E-06	1.74E-06					
12-Jun-14	557.3	558	-0.00126	1.58E-06	12	0.028555	4.51E-08	1.77E-06					
11-Jun-14	558	560.51	-0.00449	2.01E-05	13	0.026842	5.40E-07	1.79E-06					
10-Jun-14	560.51	557.15	0.006013	3.62E-05	14	0.025231	9.13E-07	1.81E-06					
9-Jun-14	557.15	558.06	-0.00163	2.66E-06	15	0.023718	6.31E-08	1.85E-06					
8-Jun-14	558.06	546.4	0.021115	0.000446	16	0.022294	9.94E-06	1.67E-06					
5-Jun-14	546.4	541.5	0.009008	8.11E-05	17	0.020957	1.70E-06	1.67E-06					
4-Jun-14	541.5	550.99	-0.01737	0.000302	18	0.019699	5.95E-06	1.58E-06					
3-Jun-14	550.99	560.7	-0.01747	0.000305	19	0.019517	5.65E-06	1.49E-06					
2-Jun-14	560.7	560.8	-0.00018	3.18E-08	20	0.017406	5.54E-10	1.52E-06					

## Steps Taken in Excel for Google Returns:

**Step 1:** Column E is the squared returns for Google for the period. Column F is simply an index to be used later during exponentiation. In this example it goes from 0 to 62.

**Step 2:** The formula for Col G is  $(1-\lambda)\lambda^{t-1}$ . This creates the weights,  $(1-\lambda)\lambda^{t-1}$ , that EWMA uses. Lambda equals 0.94 so the first weight (greatest weight) equals  $(1-0.94)0.94^0=0.06$ . The last term is  $(1-0.94)0.94^{62}=0.0013$ .

**Step 3:** col H is the weighted squared returns or col G times col E or  $(1-\lambda)\lambda^{t-1}r_t^2$

<sup>1</sup>

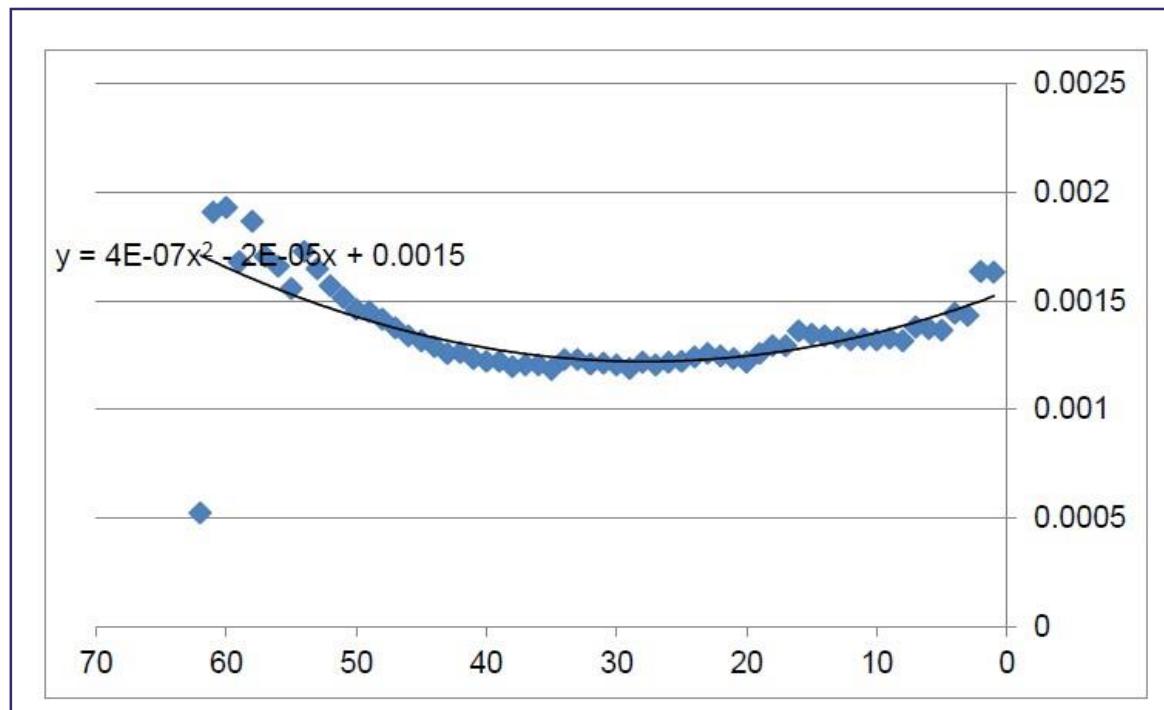
**Step 4:** Volatility squared: Excel has a great function that allows for a recursive operation using averaging. Thus, col I is the average of the cell with all the ones below it. In the diagram cell I5=average(h6:h67). Note: the data don't go past 63 but all the ones below that are blank and Excel is smart enough not to include blank cells in the averaging.

**Step 5:** Volatility is the last column in the picture below and is the square root.

The screenshot shows a Microsoft Excel window with the title bar "Microsoft Excel non-commercial use". The ribbon menu includes Home, Insert, Page Layout, Formulas, Data, Review, and View. The main area displays a table with the following columns:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	date	Goog P	lag	ret	sq ret	weights	wx sqrt	sigmas	Volatility				
2	30-Jun-14	578.66	577.18	0.002561	6.56E-06	0	0.06	3.94E-07	2.66E-06	0.001631			
3	27-Jun-14	577.18	581	-0.0066	4.35E-05	1	0.0564	2.45E-06	2.67E-06	0.001634			
4	26-Jun-14	581	565.26	0.027465	0.000754	2	0.053018	4.00E-05	2.05E-06	0.001432			
5	25-Jun-14	565.26	565.19	0.000124	1.53E-08	3	0.049835	7.62E-10	2.08E-06	0.001442			
6	24-Jun-14	565.19	555.15	0.017924	0.000321	4	0.046845	1.50E-05	1.86E-06	0.001364			
7	23-Jun-14	555.15	556.85	-0.00306	9.35E-06	5	0.044034	4.12E-07	1.86E-06	0.001371			
8	20-Jun-14	556.85	554.24	0.004698	2.21E-05	6	0.041392	9.16E-07	1.90E-06	0.001378			
9	19-Jun-14	554.24	544.86	0.017069	0.000291	7	0.038909	1.13E-05	1.73E-06	0.001315			
10	18-Jun-14	544.86	544.2	0.001212	1.47E-06	8	0.036574	5.38E-08	1.76E-06	0.001327	lambda=		
11	17-Jun-14	544.2	549.26	-0.00926	8.57E-05	9	0.03438	2.95E-06	1.74E-06	0.001319	0.94		
12	16-Jun-14	549.26	552.26	-0.00545	2.97E-05	10	0.032317	9.60E-07	1.75E-06	0.001323			
13	13-Jun-14	552.26	557.3	0.00908	8.25E-05	11	0.030378	2.51E-06	1.74E-06	0.001319			
14	12-Jun-14	557.3	558	-0.00126	1.58E-06	12	0.028555	4.51E-08	1.77E-06	0.00133			
15	11-Jun-14	558	560.51	-0.00449	2.01E-05	13	0.026842	5.40E-07	1.79E-06	0.001338			
16	10-Jun-14	560.51	557.15	0.006013	3.62E-05	14	0.025231	9.13E-07	1.81E-06	0.001345			
17	9-Jun-14	557.15	558.06	-0.00163	2.66E-06	15	0.023718	6.31E-08	1.85E-06	0.00136			
18	6-Jun-14	558.06	546.4	0.021115	0.000446	16	0.022294	9.94E-06	1.67E-06	0.001292			
19	5-Jun-14	546.4	541.5	0.009008	8.11E-05	17	0.020957	1.70E-06	1.67E-06	0.001292			
20	4-Jun-14	541.5	550.99	-0.01737	0.000302	18	0.019699	5.95E-06	1.58E-06	0.001257			
21	3-Jun-14	550.99	560.7	-0.01747	0.000305	19	0.018517	5.65E-06	1.48E-06	0.001217			
22	2-Jun-14	560.7	560.8	-0.00018	3.18E-08	20	0.017408	5.54E-10	1.52E-06	0.001233			

The graph of the volatilities is shown below. Note the volatility decrease and seems to increase in recent days (on the right).



## EWMA and Quantitative Finance

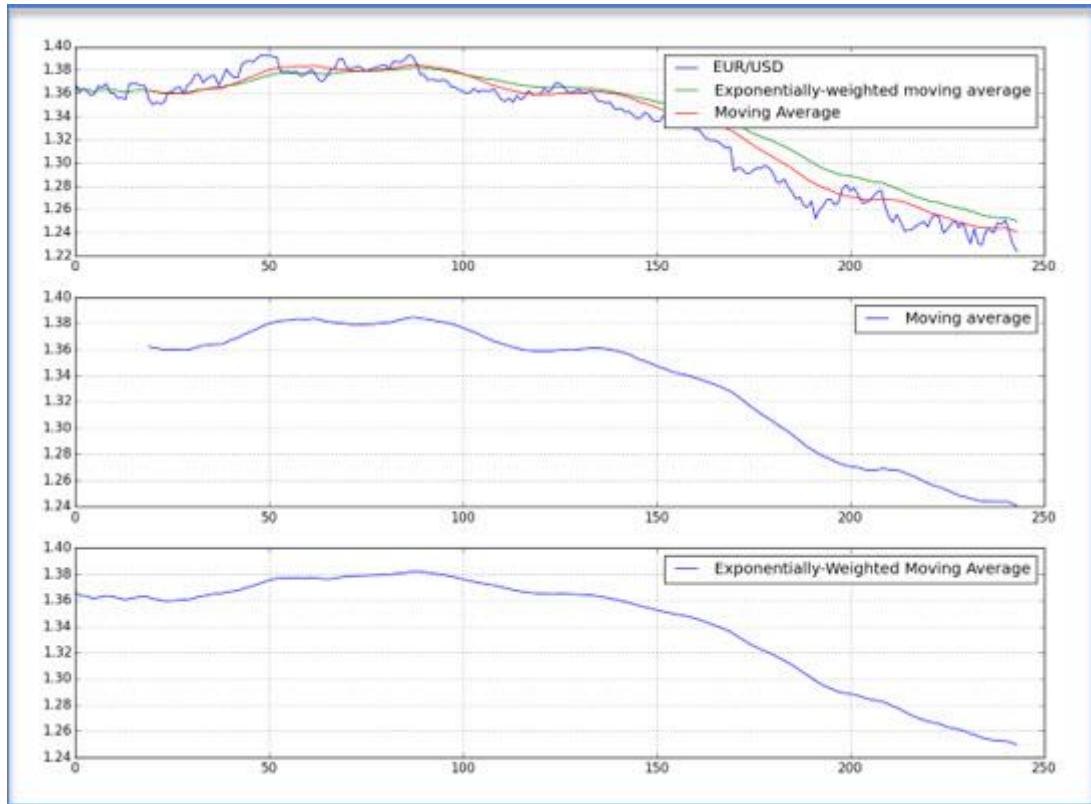
There are many types of moving averages that can be used in technical analysis, the most popular indicators being the simple and exponential moving average.

$$EMAn = EMAn-1 + (pn - EMAn-1)$$

We use `ewma` functions from Pandas in order to calculate the Exponentially-weighted moving average.

### Python Code:

```
ewma=pd.ewma(eurusd, 20)
plt.subplot(311)
plt.plot(eurusd, label="EUR/USD")
plt.plot(ewma, label="Exponentially-weighted moving
average")
plt.plot(sma, label="Moving Average")
plt.grid(True)
plt.legend()
plt.subplot(312)
plt.plot(sma, label="Moving average")
plt.legend()
plt.grid(True)
plt.subplot(313)
plt.plot(ewma, label="Exponentially-Weighted Moving
Average")
plt.legend()
plt.grid(True)
```



FX trading strategies using moving averages:

The price is rising and closes above the moving average – Buy signal

The price is falling and closes below the moving average – Sell signal

Fast moving average crosses above a slower moving average – Buy signal

Fast moving average crosses below a slower moving average – Sell signal

## Trading Rules and Concluding Thoughts

Below you will see several trading rules:

- FX trading is a probability game. You never know exactly what the evolution of the currency pair will be in the future. There are always risks you have to deal with.
- There are hundreds or even thousands of indicators that can be used in technical analysis.
- You can build your own indicators in Python if you believe it will help you in trading.
- Most of the time you will combine the information provided by different indicators for your trading decisions.
- You were provided some basic FX trading strategies. This does not mean that you can use them in FX trading and make profits with a 100% probability. You can win, you can lose - FX trading is a probability game.
- Most successful FX traders do not publish their tips and tricks they use in order to make profits.
- FX trading is not a job, it is a way of living. It requires a lot of experience, education and skill.

- Successful trading does not mean finding a magic formula, making millions in a few hours, and retiring. Successful trading means studying markets and new techniques for hours every day. Successful trading means learning from your mistakes and developing success in your future trading.

# Value-at-Risk Model

## Parametric (Analytical) Value-at-Risk

Value at risk (VaR) as the name suggests is a measure of risk of loss for a particular asset or portfolio. The probability level is one minus the probability of a VaR break (extremely risky scenario), for example if the confidence level is the extreme left-most 1% the specified confidence level is 100%-1% or 99% confidence level.

Assumptions:

- The returns from the asset/portfolio is normally distributed. This allows the use of the two params from the normal mean and standard deviation.
- We choose a level of confidence. If the confidence is 95% sure the worst case is not going to happen choose z-value = -1.645. If the confidence is 99% sure the worst case is not going to happen choose z-value = -2.33. The z value represents the number of standard deviations away from the mean.
- The returns are assumed to be serially independent so no prior return should influence the current return.

## Steps required to Get VaR

1. Find (log) returns from financial data (keep things as percentages)
2. Calculate mean (percentage)
3. Calculate standard deviation of means (percentage)
4. Choose a confidence level (we assume  $z=1.645$ )
5. Calculate the dollar loss associated with your investment as a money manager. Assume you invested \$10m in a security.

### Example:

Choose a 95% confidence level, meaning we wish to have 5% of the observations in the left-hand tail of the normal distribution. That means that the observations in that area are 1.645 standard deviations away from the mean (assume to be zero for short period-of-time return data).

The following are data for a security investment

- Amount of dollars invested: \$10 million
- Standard deviation: 1.99% or .0199

The VaR at the 95% confidence level is  $1.645 \times 0.0199$  or 0.032736. The portfolio has a market value of £10 million, so the VaR of the portfolio is  $0.032736 \times 10,000,000 = \$327,360$ .

## Relationship of Correlation and VaR

Covariance VaR changes linearly as the underlying model or factor variable instantaneous correlations change. Likewise, non-linear VaR will change non-linearly as the instantaneous correlations change. Dynamic hedging costs should properly reflect the VaR cost of cap.

# Credit Value Adjustments

## Credit Value Adjustments

“Credit Value-at-Risk (CVAR) is a measure of risk resulting from the possibility that a counterparty might not be able to honor its obligations” – Yves Hilpisch.

“Credit valuation adjustment (CVA) is the difference between the risk-free portfolio value and the true portfolio value that takes into account the possibility of a counterparty default. In other words, CVA is the market value of counterparty credit risk” – Wikipedia definition of CVA.

“CVA is derived from the CVaR” – Yves Hilpisch

The default of large corporations in the past 30 years proved that no investors are immune from credit risk. The 2007/2008 global liquidity showed that the default risk is even higher during periods of significant turmoil. Therefore, financial institutions invested heavily in developing new tools for credit risk analysis and forecasting.

Companies started to calculate the Counterparty Credit Risk (CCR) to incorporate into derivatives' prices. Therefore, CVA emerged as a useful tool in credit risk analysis.

### CVA in practice:

- CVA is the price one would pay to hedge the portfolio of derivative instruments against counterparty credit risk
- CVA was introduced in 2007/2008 in order to enhance fair value accounting
- CVA is largely used in the derivatives markets
- There is not a standardized methodology for CVA as financial companies use different versions of CVA depending on their needs and complexity required
- Complex CVA approaches require market risk factor simulations and different market scenarios
- Large financial institutions that have a high share of derivatives in their portfolio invest heavily in CVA analysis and even have a separate CVA trading desk
- Investment banks have a CVA desk that has the following tasks:
  - hedge for possible losses due to counterparty default
  - hedge to reduce the amount of capital required under the CVA calculation of Basel 3

## Example

We have one BlackBerry stock in our portfolio. The stock price on Friday 5th of December was 10.68 USD (from Yahoo Finance as showed below). We want to calculate CVaR and CVA for this portfolio. We use Python in our analysis. The Python was provided by Yves Hilpisch.

**BlackBerry Limited (BBRY)** - NasdaqGS ★ Watchlist  
**10.68 +0.21(2.01%)** Dec 5, 4:00PM EST  
 After Hours: 10.68 0.00 (0.00%) Dec 5, 6:39PM EST

Prev Close:	10.47	Day's Range:	10.42 - 10.75
Open:	10.47	52wk Range:	5.44 - 12.54
Bid:	N/A	Volume:	6,681,071
Ask:	10.80 x 100	Avg Vol (3m):	13,587,900
1y Target Est:	9.48	Market Cap:	5.55B
Beta:	0.14	P/E (ttm):	N/A
NextEarnings Date:	19-Dec-14	EPS (ttm):	-9.66
		Div & Yield:	N/A (N/A)

Beat the market  
Get the app

BlackBerry Limited  
BBRY Dec 5, 4:00pm EST 10.8  
10.7  
10.6  
10.5  
10.4  
10.3  
10.2  
10.1  
10.0  
9.9  
9.8  
9.7  
9.6  
9.5  
9.4  
9.3  
9.2  
9.1  
9.0  
8.9  
8.8  
8.7  
8.6  
8.5  
8.4  
8.3  
8.2  
8.1  
8.0  
7.9  
7.8  
7.7  
7.6  
7.5  
7.4  
7.3  
7.2  
7.1  
7.0  
6.9  
6.8  
6.7  
6.6  
6.5  
6.4  
6.3  
6.2  
6.1  
6.0  
5.9  
5.8  
5.7  
5.6  
5.5  
5.4  
5.3  
5.2  
5.1  
5.0  
4.9  
4.8  
4.7  
4.6  
4.5  
4.4  
4.3  
4.2  
4.1  
4.0  
3.9  
3.8  
3.7  
3.6  
3.5  
3.4  
3.3  
3.2  
3.1  
3.0  
2.9  
2.8  
2.7  
2.6  
2.5  
2.4  
2.3  
2.2  
2.1  
2.0  
1.9  
1.8  
1.7  
1.6  
1.5  
1.4  
1.3  
1.2  
1.1  
1.0  
0.9  
0.8  
0.7  
0.6  
0.5  
0.4  
0.3  
0.2  
0.1  
0.0  
-0.1  
-0.2  
-0.3  
-0.4  
-0.5  
-0.6  
-0.7  
-0.8  
-0.9  
-1.0  
-1.1  
-1.2  
-1.3  
-1.4  
-1.5  
-1.6  
-1.7  
-1.8  
-1.9  
-2.0  
-2.1  
-2.2  
-2.3  
-2.4  
-2.5  
-2.6  
-2.7  
-2.8  
-2.9  
-3.0  
-3.1  
-3.2  
-3.3  
-3.4  
-3.5  
-3.6  
-3.7  
-3.8  
-3.9  
-4.0  
-4.1  
-4.2  
-4.3  
-4.4  
-4.5  
-4.6  
-4.7  
-4.8  
-4.9  
-5.0  
-5.1  
-5.2  
-5.3  
-5.4  
-5.5  
-5.6  
-5.7  
-5.8  
-5.9  
-6.0  
-6.1  
-6.2  
-6.3  
-6.4  
-6.5  
-6.6  
-6.7  
-6.8  
-6.9  
-7.0  
-7.1  
-7.2  
-7.3  
-7.4  
-7.5  
-7.6  
-7.7  
-7.8  
-7.9  
-8.0  
-8.1  
-8.2  
-8.3  
-8.4  
-8.5  
-8.6  
-8.7  
-8.8  
-8.9  
-9.0  
-9.1  
-9.2  
-9.3  
-9.4  
-9.5  
-9.6  
-9.7  
-9.8  
-9.9  
-10.0  
-10.1  
-10.2  
-10.3  
-10.4  
-10.5  
-10.6  
-10.7  
-10.8  
-10.9  
-10.10  
-10.11  
-10.12  
-10.13  
-10.14  
-10.15  
-10.16  
-10.17  
-10.18  
-10.19  
-10.20  
-10.21  
-10.22  
-10.23  
-10.24  
-10.25  
-10.26  
-10.27  
-10.28  
-10.29  
-10.30  
-10.31  
-10.32  
-10.33  
-10.34  
-10.35  
-10.36  
-10.37  
-10.38  
-10.39  
-10.40  
-10.41  
-10.42  
-10.43  
-10.44  
-10.45  
-10.46  
-10.47  
-10.48  
-10.49  
-10.50  
-10.51  
-10.52  
-10.53  
-10.54  
-10.55  
-10.56  
-10.57  
-10.58  
-10.59  
-10.60  
-10.61  
-10.62  
-10.63  
-10.64  
-10.65  
-10.66  
-10.67  
-10.68  
-10.69  
-10.70  
-10.71  
-10.72  
-10.73  
-10.74  
-10.75  
-10.76  
-10.77  
-10.78  
-10.79  
-10.80  
-10.81  
-10.82  
-10.83  
-10.84  
-10.85  
-10.86  
-10.87  
-10.88  
-10.89  
-10.90  
-10.91  
-10.92  
-10.93  
-10.94  
-10.95  
-10.96  
-10.97  
-10.98  
-10.99  
-10.100  
-10.101  
-10.102  
-10.103  
-10.104  
-10.105  
-10.106  
-10.107  
-10.108  
-10.109  
-10.110  
-10.111  
-10.112  
-10.113  
-10.114  
-10.115  
-10.116  
-10.117  
-10.118  
-10.119  
-10.120  
-10.121  
-10.122  
-10.123  
-10.124  
-10.125  
-10.126  
-10.127  
-10.128  
-10.129  
-10.130  
-10.131  
-10.132  
-10.133  
-10.134  
-10.135  
-10.136  
-10.137  
-10.138  
-10.139  
-10.140  
-10.141  
-10.142  
-10.143  
-10.144  
-10.145  
-10.146  
-10.147  
-10.148  
-10.149  
-10.150  
-10.151  
-10.152  
-10.153  
-10.154  
-10.155  
-10.156  
-10.157  
-10.158  
-10.159  
-10.160  
-10.161  
-10.162  
-10.163  
-10.164  
-10.165  
-10.166  
-10.167  
-10.168  
-10.169  
-10.170  
-10.171  
-10.172  
-10.173  
-10.174  
-10.175  
-10.176  
-10.177  
-10.178  
-10.179  
-10.180  
-10.181  
-10.182  
-10.183  
-10.184  
-10.185  
-10.186  
-10.187  
-10.188  
-10.189  
-10.190  
-10.191  
-10.192  
-10.193  
-10.194  
-10.195  
-10.196  
-10.197  
-10.198  
-10.199  
-10.200  
-10.201  
-10.202  
-10.203  
-10.204  
-10.205  
-10.206  
-10.207  
-10.208  
-10.209  
-10.210  
-10.211  
-10.212  
-10.213  
-10.214  
-10.215  
-10.216  
-10.217  
-10.218  
-10.219  
-10.220  
-10.221  
-10.222  
-10.223  
-10.224  
-10.225  
-10.226  
-10.227  
-10.228  
-10.229  
-10.230  
-10.231  
-10.232  
-10.233  
-10.234  
-10.235  
-10.236  
-10.237  
-10.238  
-10.239  
-10.240  
-10.241  
-10.242  
-10.243  
-10.244  
-10.245  
-10.246  
-10.247  
-10.248  
-10.249  
-10.250  
-10.251  
-10.252  
-10.253  
-10.254  
-10.255  
-10.256  
-10.257  
-10.258  
-10.259  
-10.260  
-10.261  
-10.262  
-10.263  
-10.264  
-10.265  
-10.266  
-10.267  
-10.268  
-10.269  
-10.270  
-10.271  
-10.272  
-10.273  
-10.274  
-10.275  
-10.276  
-10.277  
-10.278  
-10.279  
-10.280  
-10.281  
-10.282  
-10.283  
-10.284  
-10.285  
-10.286  
-10.287  
-10.288  
-10.289  
-10.290  
-10.291  
-10.292  
-10.293  
-10.294  
-10.295  
-10.296  
-10.297  
-10.298  
-10.299  
-10.300  
-10.301  
-10.302  
-10.303  
-10.304  
-10.305  
-10.306  
-10.307  
-10.308  
-10.309  
-10.310  
-10.311  
-10.312  
-10.313  
-10.314  
-10.315  
-10.316  
-10.317  
-10.318  
-10.319  
-10.320  
-10.321  
-10.322  
-10.323  
-10.324  
-10.325  
-10.326  
-10.327  
-10.328  
-10.329  
-10.330  
-10.331  
-10.332  
-10.333  
-10.334  
-10.335  
-10.336  
-10.337  
-10.338  
-10.339  
-10.340  
-10.341  
-10.342  
-10.343  
-10.344  
-10.345  
-10.346  
-10.347  
-10.348  
-10.349  
-10.350  
-10.351  
-10.352  
-10.353  
-10.354  
-10.355  
-10.356  
-10.357  
-10.358  
-10.359  
-10.360  
-10.361  
-10.362  
-10.363  
-10.364  
-10.365  
-10.366  
-10.367  
-10.368  
-10.369  
-10.370  
-10.371  
-10.372  
-10.373  
-10.374  
-10.375  
-10.376  
-10.377  
-10.378  
-10.379  
-10.380  
-10.381  
-10.382  
-10.383  
-10.384  
-10.385  
-10.386  
-10.387  
-10.388  
-10.389  
-10.390  
-10.391  
-10.392  
-10.393  
-10.394  
-10.395  
-10.396  
-10.397  
-10.398  
-10.399  
-10.400  
-10.401  
-10.402  
-10.403  
-10.404  
-10.405  
-10.406  
-10.407  
-10.408  
-10.409  
-10.410  
-10.411  
-10.412  
-10.413  
-10.414  
-10.415  
-10.416  
-10.417  
-10.418  
-10.419  
-10.420  
-10.421  
-10.422  
-10.423  
-10.424  
-10.425  
-10.426  
-10.427  
-10.428  
-10.429  
-10.430  
-10.431  
-10.432  
-10.433  
-10.434  
-10.435  
-10.436  
-10.437  
-10.438  
-10.439  
-10.440  
-10.441  
-10.442  
-10.443  
-10.444  
-10.445  
-10.446  
-10.447  
-10.448  
-10.449  
-10.450  
-10.451  
-10.452  
-10.453  
-10.454  
-10.455  
-10.456  
-10.457  
-10.458  
-10.459  
-10.460  
-10.461  
-10.462  
-10.463  
-10.464  
-10.465  
-10.466  
-10.467  
-10.468  
-10.469  
-10.470  
-10.471  
-10.472  
-10.473  
-10.474  
-10.475  
-10.476  
-10.477  
-10.478  
-10.479  
-10.480  
-10.481  
-10.482  
-10.483  
-10.484  
-10.485  
-10.486  
-10.487  
-10.488  
-10.489  
-10.490  
-10.491  
-10.492  
-10.493  
-10.494  
-10.495  
-10.496  
-10.497  
-10.498  
-10.499  
-10.500  
-10.501  
-10.502  
-10.503  
-10.504  
-10.505  
-10.506  
-10.507  
-10.508  
-10.509  
-10.510  
-10.511  
-10.512  
-10.513  
-10.514  
-10.515  
-10.516  
-10.517  
-10.518  
-10.519  
-10.520  
-10.521  
-10.522  
-10.523  
-10.524  
-10.525  
-10.526  
-10.527  
-10.528  
-10.529  
-10.530  
-10.531  
-10.532  
-10.533  
-10.534  
-10.535  
-10.536  
-10.537  
-10.538  
-10.539  
-10.540  
-10.541  
-10.542  
-10.543  
-10.544  
-10.545  
-10.546  
-10.547  
-10.548  
-10.549  
-10.550  
-10.551  
-10.552  
-10.553  
-10.554  
-10.555  
-10.556  
-10.557  
-10.558  
-10.559  
-10.560  
-10.561  
-10.562  
-10.563  
-10.564  
-10.565  
-10.566  
-10.567  
-10.568  
-10.569  
-10.570  
-10.571  
-10.572  
-10.573  
-10.574  
-10.575  
-10.576  
-10.577  
-10.578  
-10.579  
-10.580  
-10.581  
-10.582  
-10.583  
-10.584  
-10.585  
-10.586  
-10.587  
-10.588  
-10.589  
-10.590  
-10.591  
-10.592  
-10.593  
-10.594  
-10.595  
-10.596  
-10.597  
-10.598  
-10.599  
-10.600  
-10.601  
-10.602  
-10.603  
-10.604  
-10.605  
-10.606  
-10.607  
-10.608  
-10.609  
-10.610  
-10.611  
-10.612  
-10.613  
-10.614  
-10.615  
-10.616  
-10.617  
-10.618  
-10.619  
-10.620  
-10.621  
-10.622  
-10.623  
-10.624  
-10.625  
-10.626  
-10.627  
-10.628  
-10.629  
-10.630  
-10.631  
-10.632  
-10.633  
-10.634  
-10.635  
-10.636  
-10.637  
-10.638  
-10.639  
-10.640  
-10.641  
-10.642  
-10.643  
-10.644  
-10.645  
-10.646  
-10.647  
-10.648  
-10.649  
-10.650  
-10.651  
-10.652  
-10.653  
-10.654  
-10.655  
-10.656  
-10.657  
-10.658  
-10.659  
-10.660  
-10.661  
-10.662  
-10.663  
-10.664  
-10.665  
-10.666  
-10.667  
-10.668  
-10.669  
-10.670  
-10.671  
-10.672  
-10.673  
-10.674  
-10.675  
-10.676  
-10.677  
-10.678  
-10.679  
-10.680  
-10.681  
-10.682  
-10.683  
-10.684  
-10.685  
-10.686  
-10.687  
-10.688  
-10.689  
-10.690  
-10.691  
-10.692  
-10.693  
-10.694  
-10.695  
-10.696  
-10.697  
-10.698  
-10.699  
-10.700  
-10.701  
-10.702  
-10.703  
-10.704  
-10.705  
-10.706  
-10.707  
-10.708  
-10.709  
-10.710  
-10.711  
-10.712  
-10.713  
-10.714  
-10.715  
-10.716  
-10.717  
-10.718  
-10.719  
-10.720  
-10.721  
-10.722  
-10.723  
-10.724  
-10.725  
-10.726  
-10.727  
-10.728  
-10.729  
-10.730  
-10.731  
-10.732  
-10.733  
-10.734  
-10.735  
-10.736  
-10.737  
-10.738  
-10.739  
-10.740  
-10.741  
-10.742  
-10.743  
-10.744  
-10.745  
-10.746  
-10.747  
-10.748  
-10.749  
-10.750  
-10.751  
-10.752  
-10.753  
-10.754  
-10.755  
-10.756  
-10.757  
-10.758  
-10.759  
-10.760  
-10.761  
-10.762  
-10.763  
-10.764  
-10.765  
-10.766  
-10.767  
-10.768  
-10.769  
-10.770  
-10.771  
-10.772  
-10.773  
-10.774  
-10.775  
-10.776  
-10.777  
-10.778  
-10.779  
-10.780  
-10.781  
-10.782  
-10.783  
-10.784  
-10.785  
-10.786  
-10.787  
-10.788  
-10.789  
-10.790  
-10.791  
-10.792  
-10.793  
-10.794  
-10.795  
-10.796  
-10.797  
-10.798  
-10.799  
-10.800  
-10.801  
-10.802  
-10.803  
-10.804  
-10.805  
-10.806  
-10.807  
-10.808  
-10.809  
-10.810  
-10.811  
-10.812  
-10.813  
-10.814  
-10.815  
-10.816  
-10.817  
-10.818  
-10.819  
-10.820  
-10.821  
-10.822  
-10.823  
-10.824  
-10.825  
-10.826  
-10.827  
-10.828  
-10.829  
-10.830  
-10.831  
-10.832  
-10.833  
-10.834  
-10.835  
-10.836  
-10.837  
-10.838  
-10.839  
-10.840  
-10.841  
-10.842  
-10.843  
-10.844  
-10.845  
-10.846  
-10.847  
-10.848  
-10.849  
-10.850  
-10.851  
-10.852  
-10.853  
-10.854  
-10.855  
-10.856  
-10.857  
-10.858  
-10.859  
-10.860  
-10.861  
-10.862  
-10.863  
-10.864  
-10.865  
-10.866  
-10.867  
-10.868  
-10.869  
-10.870  
-10.871  
-10.872  
-10.873  
-10.874  
-10.875  
-10.876  
-10.877  
-10.878  
-10.879  
-10.880  
-10.881  
-10.882  
-10.883  
-10.884  
-10.885  
-10.886  
-10.887  
-10.888  
-10.889  
-10.890  
-10.891  
-10.892  
-10.893  
-10.894  
-10.895  
-10.896  
-10.897  
-10.898  
-10.899  
-10.900  
-10.901  
-10.902  
-10.903  
-10.904  
-10.905  
-10.906  
-10.907  
-10.908  
-10.909  
-10.910  
-10.911  
-10.912  
-10.913  
-10.914  
-10.915  
-10.916  
-10.917  
-10.918  
-10.919  
-10.920  
-10.921  
-10.922  
-10.923  
-10.924  
-10.925  
-10.926  
-10.927  
-10.928  
-10.929  
-10.930  
-10.931  
-10.932  
-10.933  
-10.934  
-10.935  
-10.936  
-10.937  
-10.938  
-10.939  
-10.940  
-10.941  
-10.942  
-10.943  
-10.944  
-10.945  
-10.946  
-10.947  
-10.948  
-10.949  
-10.950  
-10.951  
-10.952  
-10.953  
-10.954  
-10.955  
-10.956  
-10.957  
-10.958  
-10.959  
-10.960  
-10.961  
-10.962  
-10.963  
-10.964  
-10.965  
-10.966  
-10.967  
-10.968  
-10.969  
-10.970  
-10.971  
-10.972  
-10.973  
-10.974  
-10.975  
-10.976  
-10.977  
-10.978  
-10.979  
-10.980  
-10.981  
-10.982  
-10.983  
-10.984  
-10.985  
-10.986  
-10.987  
-10.988  
-10.989  
-10.990  
-10.991  
-10.992  
-10.993  
-10.994  
-10.995  
-10.996  
-10.997  
-10.998  
-10.999  
-10.9999  
-10.99999  
-10.999999  
-10.9999999  
-10.99999999  
-10.999999999  
-10.9999999999  
-10.99999999999  
-10.999999999999  
-10.9999999999999  
-10.99999999999999  
-10.999999999999999  
-10.9999999999999999  
-10.99999999999999999  
-10.999999999999999999  
-10.9999999999999999999  
-10.99999999999999999999  
-10.999999999999999999999  
-10.9999999999999999999999  
-10.9999999999999999

## Import Python modules

```
>>> import numpy as np
>>> import numpy.random as npr
>>> from pylab import *
>>> import matplotlib.pyplot as plt
```

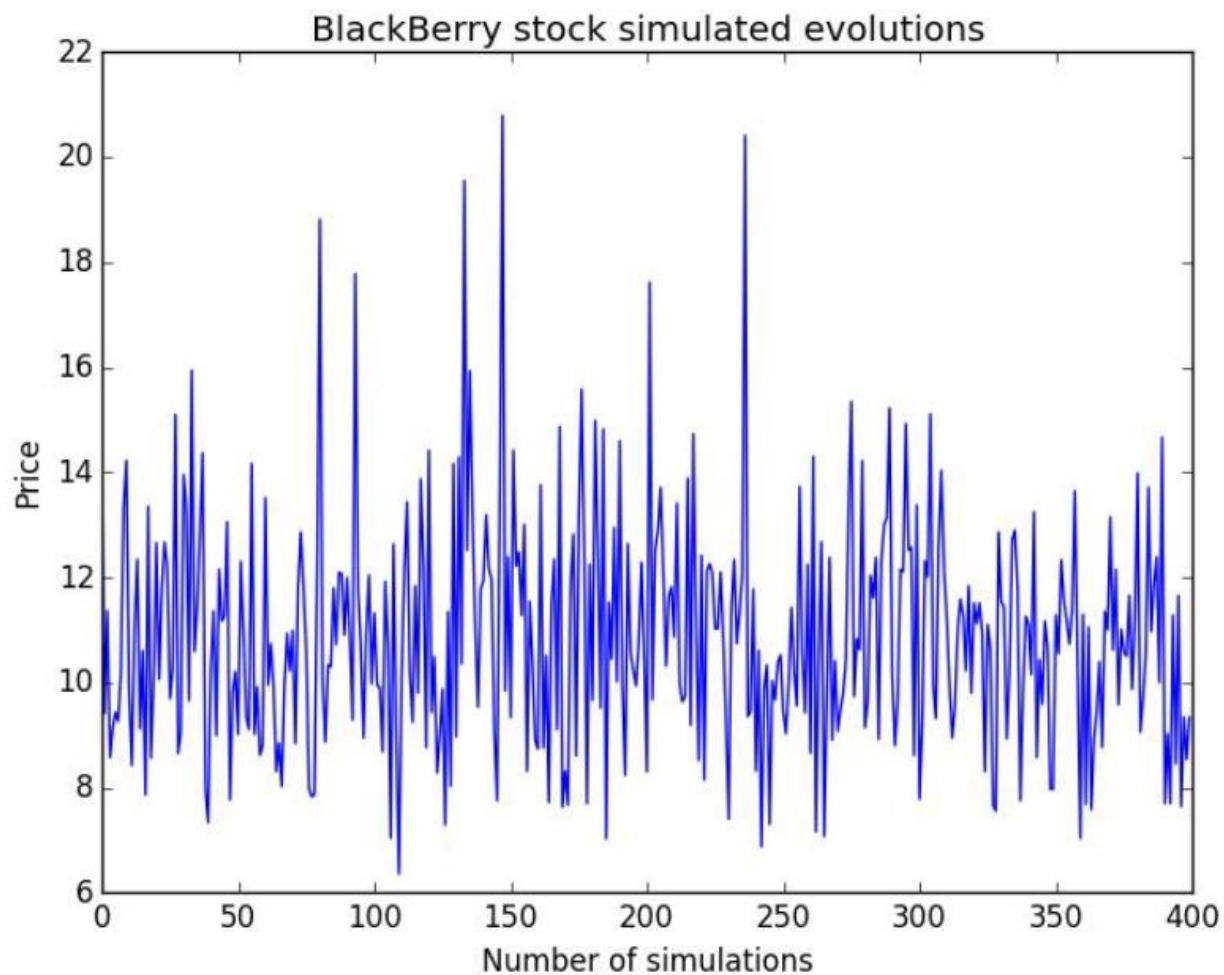
Consider the Black-Scholes-Merton model with the following parameters:

Stock price evolution is considered to follow a Geometric Brownian Motion

```
>>> S0 = 10.68
>>> r = 0.01
>>> sigma = 0.2
>>> T = 1.
>>> I=400
>>> ST = S0 * np.exp((r - 0.5 * sigma ** 2) * T + sigma * np.sqrt(T) * npr.standard_normal(I))
```

We simulate BlackBerry stock price movements.

```
>>> ST
array([ 14.1387732 ,  9.41168492,  11.36166677,  8.57137813,
       9.13998706,  9.43323287,  9.26017231,  10.21358981,
      13.34773552,  14.22240156,  9.83174466,  8.41898195,
     11.04158642,  12.33915473,  9.11802004,  10.60138712,
      7.85924642,  13.35095331,  8.55966359,  10.0508237 ,
     12.65583934,  10.06376275,  11.76174893,  12.66927371,
     12.19785931,  9.69180567,  10.25353543,  15.09600627,
      8.64751427,  9.08014664,  13.95235702,  13.33678755,
      9.65563377,  15.9380399 ,  10.59028498,  11.42548846,
     12.94248259,  14.36793257,  7.98707786,  7.33468041,
```



```

>>> L=0.05 #fixed (average) loss level
>>> p=0.01 #probability for default
>>>

```

Generate default scenarios using Poisson distribution:

```

>>> D = np.random.poisson(p * T, I)
>>> D = np.where(D > 1, 1, D)

```

Poisson distribution is used in modelling rare events, in our case if the counterparty goes bankrupt.

Risk-neutral value of the future index level is equal to the current value of the asset today if there is no default.

Risk-neutral value:

```
>>> np.exp(-r * T) * 1 / I * np.sum(ST)
10.780069055602915
```

CVaR

```
>>> CVaR = np.exp(-r * T) * 1 / I * np.sum(L * D * ST)
>>> CVaR
0.0042798199945273648
```

Present value of the asset adjusted for the credit risk:

```
>>> S0_CVA = np.exp(-r * T) * 1 / I * np.sum((1 - L * D) * ST)
>>> S0_CVA
10.775789235608386
```

CVaR can also be calculated as follows:

CVaR = Risk free value – risk-adjusted value

Another way for calculating the present value of the asset:

```
>>> S0_adj = S0 - CVaR
>>> S0_adj
10.675720180005472
```

Calculate the number of possible losses:

```
>>> np.count_nonzero(L * D * ST)
3
```

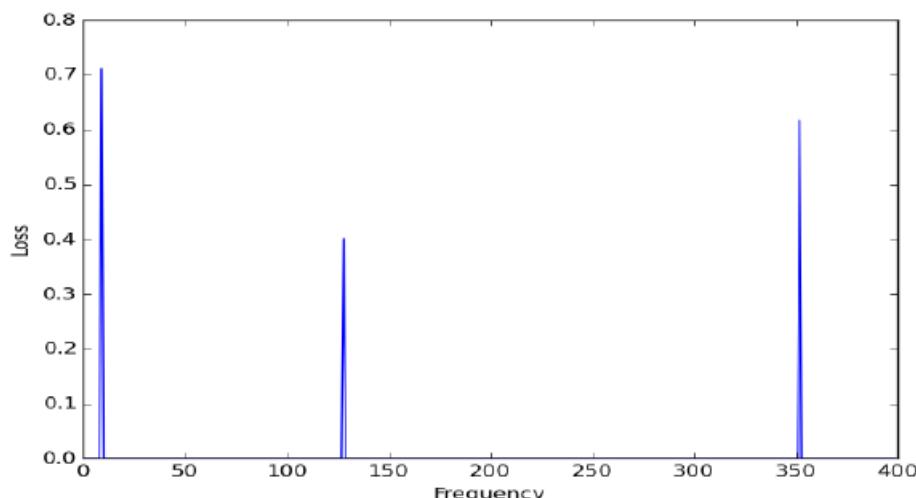
We observe only 3 losses due to credit risk given a number of 400 simulations and 1% assumed default probability

## The three possible losses:

The three losses incurred by the financial institution are: 0.71112008, 0.40135781 and 0.61665528

The fixed average loss level is 0.50

Below is a graph of losses due to risk-neutrally expected default (stock). According to the graph there are 3 losses in 400 simulations and a 1% probability of default.



# The Linear Probability Model

## The Linear Probability Model

The Linear Probability Model (LPM) is an easier to interpret alternative to Logit or Probit. Suppose the dependent variable can only assume values of 0 and 1. What if we still want to use a multiple linear regression model.

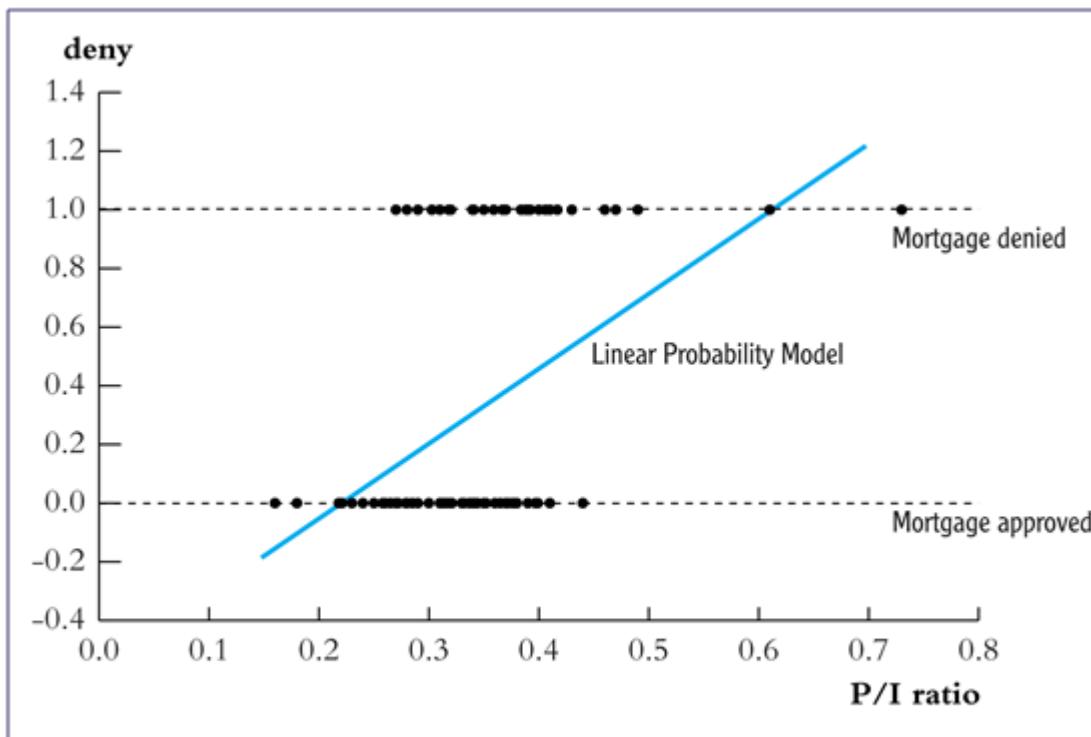
$$Y_i = b_0 + b_1 X_{1i} + b_2 X_{2i} + u_i$$

$Y_1$  can take values of 0 and 1.  
 $b_j$  - coefficient  
 $(Y_i | X) = b_0 + b_1 X_{1i} + b_2 X_{2i} = P(Y_i = 1)$  Response Probability

Predicted probability for  $b_0 + b_1 X_{1i} + b_2 X_{2i}$  can be outside of the range 0, 1.

*Example:* linear probability model, Home Mortgage Disclosure Act (HMDA) data

Mortgage denial v. ratio of debt payments to income (P/I ratio) in a subset of the HMDA data set ( $n = 127$ )

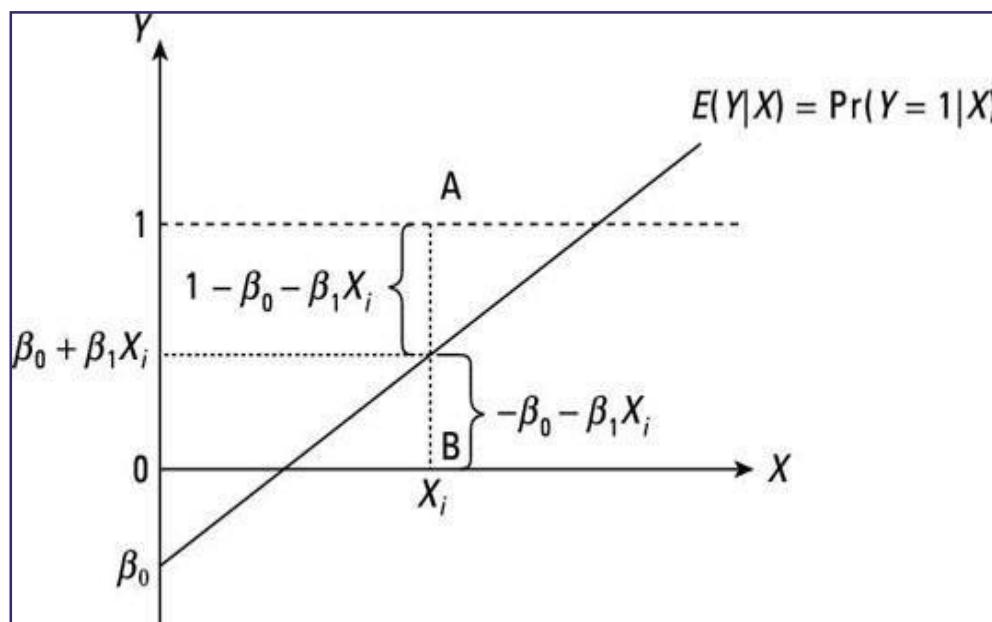


Source: [people.ku.edu/~p112m883/pdf/Econ526/Ch11Part1\\_slides.docx](http://people.ku.edu/~p112m883/pdf/Econ526/Ch11Part1_slides.docx)

**Example:**

$$Y = -0.40 + 0.09X_1 + 0.14X_2$$

One-unit increase of  $X_2$  increases the probability of occurrence of  $Y$  by 0.14.



Advantages of the LPM are:

- It's computationally simpler.
- It's easier to interpret the "marginal effects".
- "It avoids the risk of mis-specification of the "link function".
- There are complications with Logit or Probit if you have *endogenous* dummy regressors.
- The estimated marginal effects from the LPM, Logit and Probit models are usually very similar, especially if you have a large sample size.

Disadvantages of the LPM are:

LPM yields biased and inconsistent estimates in most situations. See Horrace and Oaxaca (2006), and Amemiya (1977)

# LOGIT Model

## Logit

Logit is used when estimating a binary (dummy) dependent variable.

The functional form is

$$P = \Pr(y=1|x) = \frac{e^{\alpha + \beta x}}{1 + e^{\alpha + \beta x}} \quad (\text{which assume one independent variable } x)$$

The dependent variable ( $y$ ) can take two values of 1 or 0.

$y=1$  – success

$y=0$  - failure

The probability of a failure is:

$$\begin{aligned} 1-P &= \Pr(y=0|x) = 1/(1+e^{\alpha+\beta x}) \\ \text{Odds} &= P/(1-P) = e^{\alpha+\beta x} \end{aligned}$$

The logit uses the log of the odds or

$$\ln(\text{odds}) = \ln(P/(1-P)) = \alpha + \beta x$$

The odds ratio is the probability of the event divided by the probability of the non-event.

## Logistic Regression

Logistic Regression is a statistical technique capable of predicting a binary outcome. This econometric model is an excellent algorithm for classification. Although in some cases, the logistic regression is outperformed by algorithms like SVM and Random Forests it is still used on a large scale.

Let  $y$  be a binary outcome variable

$y=0$  – indicates failure

$y=1$  – indicates success

$p$  – the probability of  $y$  to be 1

$p = \text{prob}(y=1)$

$x_1, \dots, x_k$  - a set of predictor variables

The logistic regression of  $y$  on  $x_1, x_2, \dots, x_k$  estimates parameter values for  $\beta_0, \beta_1, \dots, \beta_k$  via maximum likelihood method.

The equation is the following:

$$\text{logit}(p) = \log(p/(1-p)) = \beta_0 + \beta_1 * x_1 + \dots + \beta_k * x_k$$

In terms of probabilities, the equation above is translated into

$$p = \exp(\beta_0 + \beta_1 * x_1 + \dots + \beta_k * x_k) / (1 + \exp(\beta_0 + \beta_1 * x_1 + \dots + \beta_k * x_k)).$$

The logistic regression is used for:

- Predicting a financial crisis
- Predicting bond defaults
- Predicting a bond rating

# Probit Model

## Probit Model

Probit calculates the maximum likelihood estimates of regression parameters for the same type of binary dependent variable discussed in logit.

The probit model is defined as

$$\Pr(y=1|X) = F(\beta_1 + \beta_2X_1 + \beta_3X_2 + \dots)$$

where  $F$  is the standard cumulative normal probability distribution. The  $F$  calculator can be found online once we have found a value for  $(\beta_1 + \beta_2X_1 + \beta_3X_2 + \dots)$ . It gives us the probability.

Where the betas are the probit estimates and the  $X$ 's are the independent(explanatory) variables.

## Probit Example

In this probit example we study whether being a CEO (assumes value =1 if CEO, zero otherwise) is related to score on a common aptitude test (range of possible values is 0 to 100) and whether the person is male (male=1, zero if female). Data:

ceo	apptitude	male
1	98	1
1	87	1
1	89	1
1	99	0
1	92	0

All the code above also applies here except when we specify the model:

```
Probit_mod = sm.Probit(ceo, X)
probit_res = Probit_mod.fit()
print probit_res.summary()
```

Probit Regression Results						
<hr/>						
Dep. Variable:	ceo	No. Observations:	70			
Model:	Probit	Df Residuals:	67			
Method:	MLE	Df Model:	2			
Date:		Pseudo R-squ.:	0.6003			
Time:		Log-Likelihood:	-19.291			
converged:	True	LL-Null:	-48.263			
		LLR p-value:	2.615e-13			
<hr/>						
	coef	std err	z	P> z	[95.0% Conf. Int.]	
<hr/>						
const	-7.5025	1.500	-5.000	0.000	-10.443	-4.562
male	2.4131	0.564	4.280	0.000	1.308	3.518
apptitude	0.0791	0.017	4.751	0.000	0.046	0.112
<hr/>						

The predicted probability of the dependent variable can be calculated using the probit coefficients various values (usually the mean) of the independent variables:

$$\text{Predicted } P = F(\beta_1 + \beta_2 X_1 + \beta_3 X_2 + \dots)$$

- The betas are from the probit regression.
- The betas are inputted into the F calculator.

An online calculator to find the F value for the *cumulative distribution function of the standard normal* can be found at:

<http://www.danielsoper.com/statcalc3/calc.aspx?id=55>

# Censored Data Models

## Sample Selection Bias, Tobit, Censored Regression

Sample selection bias occurs when the sample chosen is nonrandom (biased).

**Survivorship bias** involves choosing only those company stocks for which no gaps occur in the data. A stock may have stopped trading because it was delisted or temporarily stopped trading. Researchers understandably find this inconvenient as it is difficult

**Problem:** You are trying to get an estimate of drug use among high-school age teenagers. You sample several high schools in the area. Is this random or does a bias exist? Explain.

**Problem:** Many years ago, in the U.S., a survey was done in the Great Depression asking respondents for whom they will vote. The survey *phoned* households asking their candidate preference. The respondents overwhelmingly preferred the Republican (conservative) candidate. The Democrat candidate (Roosevelt) won the election in a landslide. Where did sample selection bias come into play?

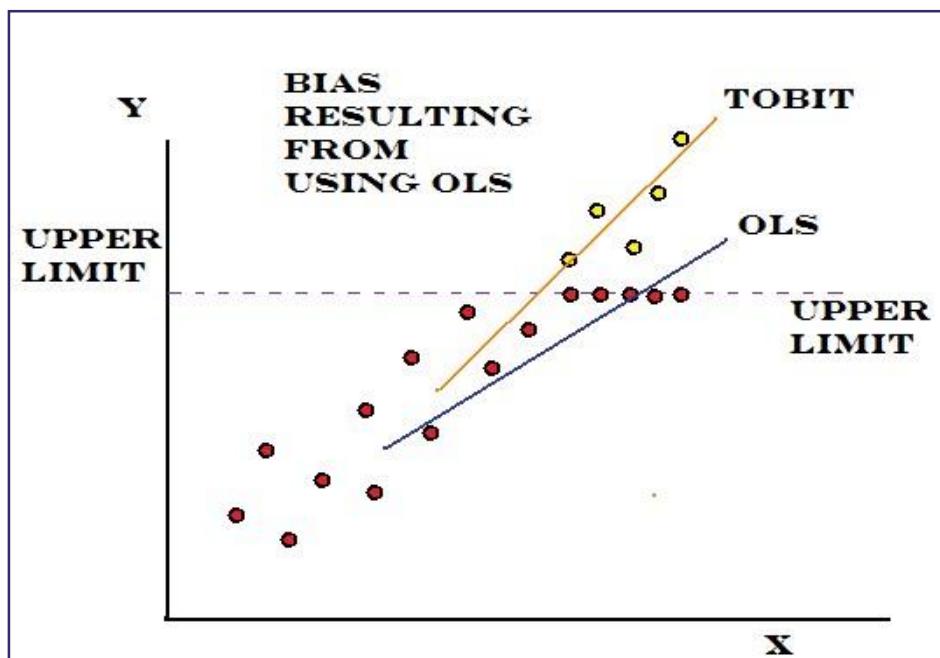
## Tobit or Censored Regression

- Censoring occurs when the dependent variable (or response variable),  $y$ , is unknown, other than being smaller than some (noise) *threshold* value (left-censored) or greater than some *saturation* limit (right-censored). This is called interval-censored.
- Sometimes only right censoring occurs; sometimes only left censoring occurs.
- Truncated regression models are used for data where whole observations are missing so that the values for the dependent and the independent variables are unknown.

**Problem 1:** Households are sometimes survey asking demographic factors such as age, education, etc. Labor surveys may ask how many hours were worked during the survey week. If number of hours worked is the dependent variable where might censoring occur?

**Problem 2:** You want to determine what the distribution of male population height looks like by using sample of soldiers heights. Note: many armies impose a minimum height requirement for soldiers. Is this censoring or truncation?

Using OLS as an estimation is ill-advised as bias results as demonstrated in the graph:



In the graph the dependent variable is (upper) bounded at 'upper limit'. We get the OLS regression line (blue) using OLS. Tobit takes into account the distribution of Y and produces the 'Tobit' line which is consistent.

## Structure

Censored regression model is a generalization of the standard Tobit model. The dependent variable can be either left-censored, right-censored, or both left-censored and right-censored as related to a lower and/or upper limit of the dependent variable  $y$ :

$$y = x'\beta + \varepsilon$$

$$y = a \quad \text{if } y^* \leq a \quad (\text{we only observe } y=a \text{ because of censoring})$$

$$y = y^* \quad \text{if } a < y^* < b$$

$$y = b \quad \text{if } y^* \geq b \quad (\text{we only observe } y=b \text{ because of censoring})$$

$a$  is the lower limit and  $b$  is the upper limit of the dependent variable.

## Estimation

Censored regression is estimated by using MLE (maximum likelihood). Assume the disturbance term above is white noise,  $(0, \sigma)$ . The likelihood function for N observations is:

$$\text{Log } L = \sum_{i=1,N} [I_{ai} \log \xi(\{a - x_i' \beta\}/\sigma) + I_{bi} \log \xi(\{x_i' \beta - b\}/\sigma) + (1 - I_{ai} - I_{bi}) \{\log \Phi([y_i - x_i' \beta]/\sigma) - \log(\sigma)\}]$$

Where

$\Phi$ : pdf of the standard normal distribution

$\xi$ : cdf of the standard normal distribution

$I$  is an indicator function such that

$I_{ai} = 1$  if  $y_i = a$

$I_{ai} = 0$  if  $y_i > a$

$I_{bi} = 1$  if  $y_i$

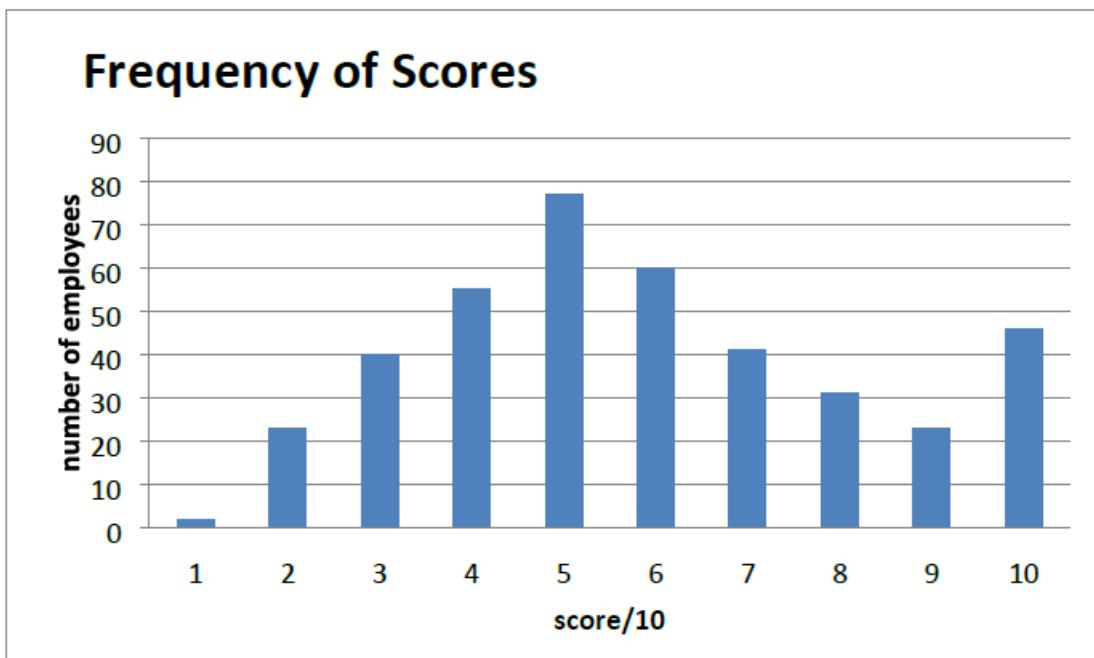
$= b$      $I_{bi} = 0$

if  $y_i < b$

**Example:**

A company is trying to determine factors influencing the supervisors' ratings (Y). The predictors are 1. The output of the employee measured in hundreds and 2. The gender (male=1) is a dummy.

The frequency of scores is given by the frequency distribution below. The scores are truncated at 100. A number of employees (specifically 46) have scores at 100 (as given by the managerial staff).



To generate a censored regression model (tobit model) list the outcome variable and then specify the lower limit and/or upper limit of the outcome variable. A tobit model can be used to predict an outcome that is censored from above, from below, or both. In this case, there is no lower limit while the upper limit is specified as 100. The model generated is a log-likelihood model.

Dependent variable=supervisors rating scale (Range from 0 to 100).

Tobit Regression			No. of Observations=398	
Log Likelihood=-1854.56			LR chi2(2) =70.89	
			Prob > chi2 =0.000	
			Pseudo R2 =0.0350	
Dependent=Y	Coefficient	Std. Error	t	P> t
Output (100s)	12.874	2.619	4.915	0.000
Gender (Male=1)	13.689	3.672	3.728	0.000
Constant	19.361	4.946	3.914	0.000
0 left-censored observations				
352 uncensored observations				
46 right-censored observations at Y>=100				

## Notes

1. **Pseudo R2** - This is McFadden's pseudo R-squared. Tobit regression does not have an equivalent to the R-squared that is found in OLS regression; There are a wide variety of pseudo-R-square statistics. This statistic does not have the same interpretation as in OLS regression.
2. **Coefficient interpretation** - If output per employee increases by 100 units the supervisor score for the employee increases by about 12.87. Thus a score would increase from, say, 50 to 62.87 if the employee's output increased by 100 units.

In contrast, if the employee is a female the score decreases by 13.69 out of 100.

## Prediction from the Model:

The Tobit allows for prediction above or below the limits:

For example. What would be the (predicted supervisor's) score for a male producing 600 units per period? The answer is

$$\text{Score} = 19.361 + 12.874(6 \text{ in hundred units}) + 13.689 \text{ (times 1 for male)}$$

$$= 19.361 + 77.244 + 13.689 = 110.294\%$$

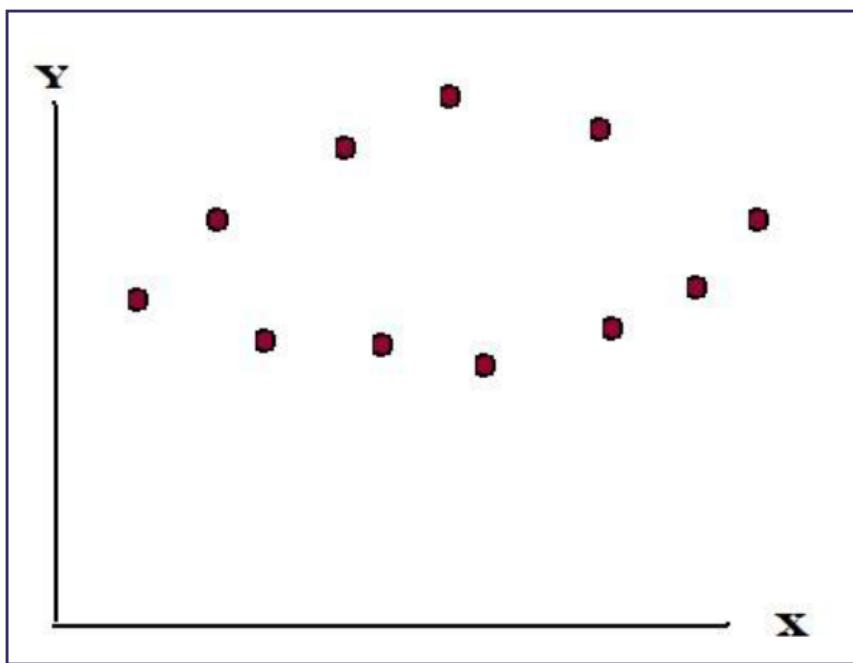
**Note** as well that a female has to produce more than 100 units more per period to get the same score as the male counterpart.

Tobit can be accessed in Python using RPY2

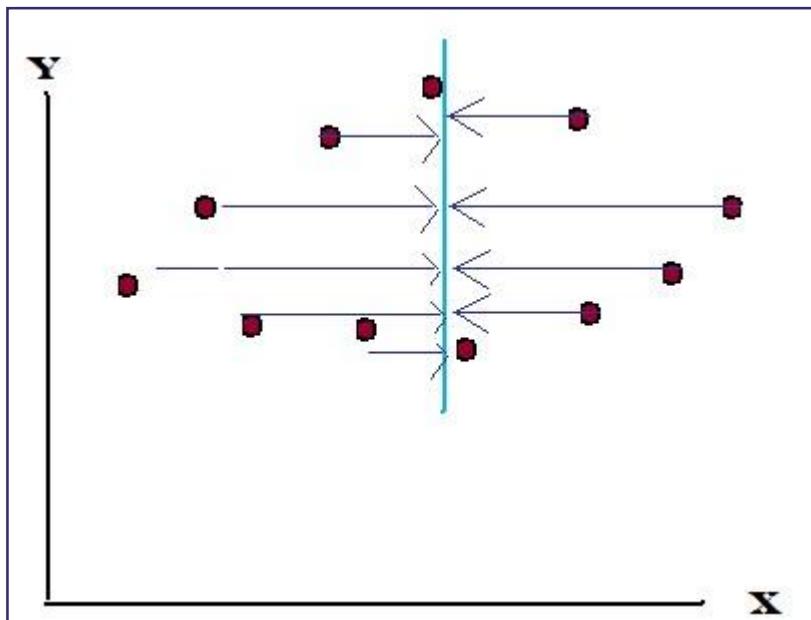
# Principal Components Analysis

## Principal Components Analysis

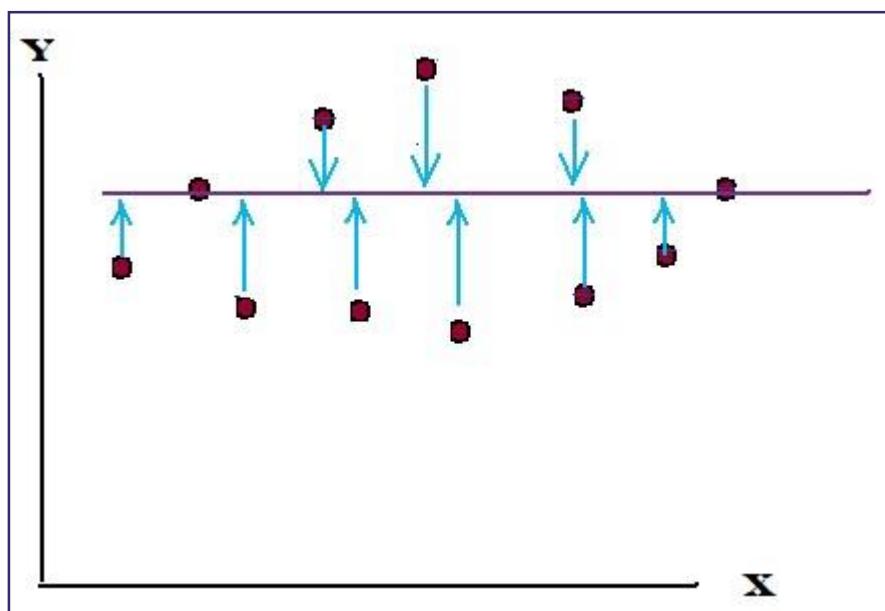
- PCs are directions where there is most variance in the data.
- PCA reduce the dimensionality of a data analysis that typically involve lots of variables
- Example: A fund manager has 100 stocks in her portfolio. The correlation matrix between the stocks would have a dimension of  $100 \times 100 = 10,000$  values.
- PCA reduces the number of variables into components say 5 so that the fund manager only has to analyze  $5 \times 5 = 25$  correlations.
- It is useful to measure data using PC rather than an x-y axis.
- Consider the set of data in x-y plane:



We want to find the direction where there is most variance. Fit a straight line where the data is most spread out when projected onto it. A vertical straight line with the points orthogonally projected on to it will look like this:



If instead the line had been run horizontally through the data the variance would have been greater as follows. This captures greater variance than the vertical line:



The regression problem has  $k$  explanatory variables ( $k$  is often large):

$X$ s:  $x_1, x_2, \dots, x_k$

$$\text{Cov}(x_i) = \sigma R_{ij} = R$$

Or in matrix form

$$\begin{vmatrix} & & \\ | & & | \\ | R_{11} & R_{12} & R_{13} & \dots & | \\ | R_{21} & R_{22} & R_{23} & \dots & | & R \text{ (cov matrix)} \\ | R_{31} & R_{32} & R_{33} & & & \\ \dots & | & | \\ \dots & & & & & \\ | i,j=1,2,\dots,k & & & & & \end{vmatrix}$$

The software calculates the eigenvectors

$z_1, z_2, \dots, z_k$

and corresponding eigenvalues of

$u_1, u_2, \dots,$   
 $u_k$  and

$$z_j^T z_j = 1$$

Then factors (principal components) are found:

$$F_1 = \sum_{i=1}^k a_{1i} x_i \quad (\text{and } a_1^T a_1 = 1)$$

- $F_1$  contains the largest possible variance.
- $F_2$  contains the second largest possible variance and is orthogonal to  $F_1$ :

$$\text{Var}(F_1) = \sigma^2 a_1^T R_{a1}$$

The Lagrangian from constrained maximization is

$$L = a_1^T R_{a1} + \lambda(a_1^T a_1 - 1)$$

Factor F1 is solved as follows:

$$R_{a1} = -\lambda a_1 \quad (a_1 \text{ is an eigenvector of } R) \\ a_1 = z_1$$

or another way to think about that

$$F_1 = z_1^T X = \sum_{i=1}^k z_{1i} x_i$$

The second factor is found as follows:

$$F_2 = a_2^T X = \sum_{i=1}^k a_{2i} x_i$$

The orthogonality condition is:

$$\text{Correlation}(F_1, F_2) = 0 \text{ Or } a_2^T z_1 = 0$$

With Lagrangian:

$$L = a_2^T R_{a2} + \lambda_1(a_2^T a_2 - 1) + \lambda_2(a_2^T z_1) \text{ And so on}$$

## Selection of the Number of Factors (PCs):

1. Eigenvalue rule
2. Scree plot
3. Percentage of variance method: A third criterion in solving the number of factors (PCs) problem involves retaining a component if it accounts for a specified percentage (or proportion) of variance in the data set. For example, you may decide to retain any component that accounts for at least 5% or 10% of the total variance. This proportion can be calculated with a simple

Formula:

- Proportion = Eigenvalue for the component of interest
- Total eigenvalues of the correlation matrix
- In principal component analysis, the “total eigenvalues of the correlation matrix” is equal to the total number of variables being analyzed (because each variable contributes one unit of variance to the analysis).
- Simple code example of PCA  
(<http://scikitlearn.org/stable/modules/generated/sklearn.decomposition.PCA.html>):

```
>>> import numpy as np
>>> from sklearn.decomposition import PCA
>>> X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1],
[3, 2]])
>>> pca = PCA(n_components=2)
>>> pca.fit(X)
PCA(copy=True, n_components=2, whiten=False)
>>> print(pca.explained_variance_ratio_)
[ 0.99244...  0.00755...]
```

## Methods

1. pca.components - Returns the PC or factors
2. pca.explained\_variance - returns the explained variance for each factor/principal component
3. Note the total variance can be found by dividing the pca.explained\_variance\_ (above) by the pca.explained\_variance\_ratio\_. That will be the same value for each factor.
4. Pca.components\_returns the eigenvectors of the covariance matrix

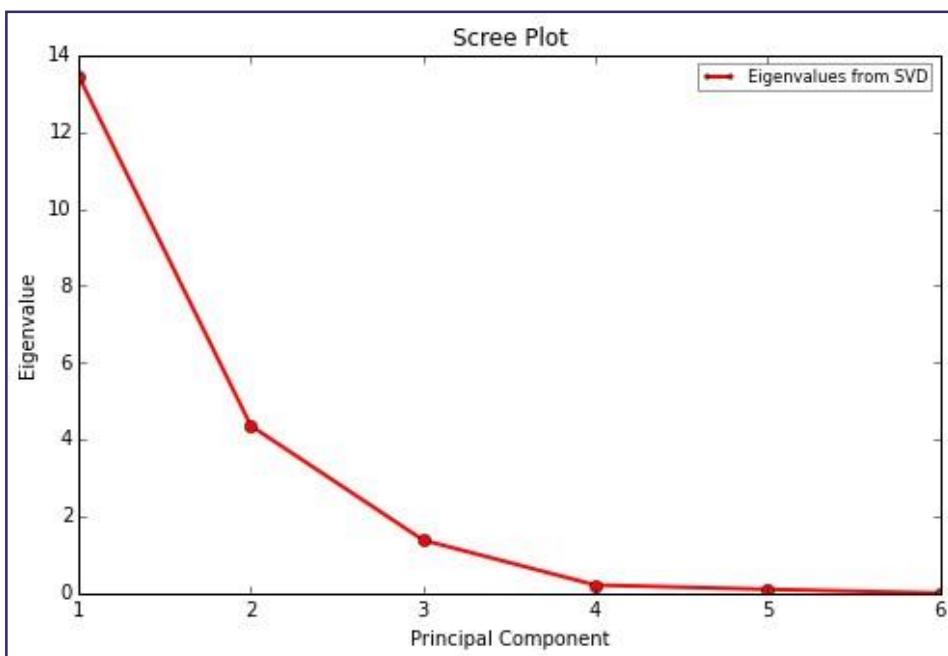
<b>fit(X[, y])</b>	<b>Fit the model with X.</b>
fit_transform(X[, y])	Fit the model with X and apply the dimensionality reduction on X.
get_covariance()	Compute data covariance with the generative model.
get_params([deep])	Get parameters for this estimator.
get_precision()	Compute data precision matrix with the generative model.
inverse_transform(X)	Transform data back to its original space, i.e.,
score(X[, y])	Return the average log-likelihood of all samples
score_samples(X)	Return the log-likelihood of each sample
set_params(**params)	Set the parameters of this estimator.
transform(X)	Apply the dimensionality reduction on X.

## Scree Plot in Python

```

import matplotlib
import matplotlib.pyplot as plt
#Make a random array and then make it positive-definite
num_vars = 6 num_obs = 9
A = np.random.randn(num_obs, num_vars)
A = np.asmatrix(A.T) * np.asmatrix(A) U, S, V =
np.linalg.svd(A) eigvals = S**2 / np.cumsum(S)[-1] fig
= plt.figure(figsize=(8,5)) sing_vals =
np.arange(num_vars) + 1 plt.plot(sing_vals, eigvals,
'ro-', linewidth=2) plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Eigenvalue')
#I don't like the default legend so I typically make mine like below, e.g.
#with smaller fonts and a bit transparent so I do not cover up data, and
make
#it moveable by the viewer in case upper-right is a bad place for it
leg = plt.legend(['Eigenvalues from SVD'], loc='best', borderpad=0.3,
shadow=False,
prop=matplotlib.font_manager.FontProperties(size='small'),
markerscale=0.4) leg.get_frame().set_alpha(0.4)
leg.draggable(state=True) plt.show()

```



## To Get Eigenvectors (Evec Below) and Eigenvalues (Eval) from Randomly Generated Data

```

import numpy as NP import
numpy.linalg as LA
# a simulated data set with 8 data points, each point having five features
data = NP.random.randint(0, 10, 40).reshape(8, 5) data Out[22]:
array([[9, 0, 4, 3, 2],
[8, 5, 8, 9, 8],
[3, 7, 9, 2, 0],
[0, 4, 7, 2, 3],
[4, 3, 6, 9, 8],
[5, 4, 3, 8, 2],
[3, 0, 7, 3, 4],
[3, 9, 2, 0, 6]])
) data -= NP.mean(data, axis=0 C:\Users\gib\Anaconda\lib\site-
packages\spyderlib\widgets\externalshell\start_ipython_kernel.py:1:
DeprecationWarning: Implicitly casting between incompatible kinds. In a future numpy
release, this will raise an error. Use casting="unsafe" if this is intentional.
# -*- coding: utf-8 -*-
data Out[24]: array([[ 4, -4, -1, -1, -
2], [ 3, 1, 2, 4, 3],
[-1, 3, 3, -2, -4],
[-4, 0, 1, -2, -1],
[ 0, -1, 0, 4, 3],
[ 0, 0, -2, 3, -2],
[-1, -4, 1, -1, 0],
[-1, 5, -3, -4, 1]])

C = NP.corrcoef(data, rowvar=0)

eval, evec = LA.eig(C)
evec Out[27]: array([[ 0.52520478, -0.31059187,  0.68210861,  0.39946855,
0.05334178],
[-0.37106069,  0.1347789 ,  0.62536826, -0.40322628,  0.53895429],
[ 0.02898133, -0.26714456,  0.20169779, -0.67757769,  0.65421735],
[ 0.62200702,  0.73256126,  0.0173553 , -0.27532167,
0.01892213],
[ 0.44580406, -0.52664517, -0.3204047 , -0.37804747,
0.52756473]])
eval
Out[28]: array([ 1.87229985,  0.37248017,  0.61506833,
0.99118941,  1.14896224])

```

# Weighted Least Squares Regression

## Weighted Least Squares

WLS assumes that the weights are known. In reality estimated weights must be used.

OLS minimizes:

$$\sum_{i=1,N} (y_i - \beta x_i)^2 \quad (\text{for the simple one-variable regressor})$$

WLS minimizes:

$$\sum_{i=1,N} w_i (y_i - \beta x_i)^2$$

Where  $w$  are weights applied to each observation. Note if  $w=1$  for all observations then WLS reverts to OLS trivially.

Example of WLS in Python

## Y Data

5.81222431, 5.11955505, 5.96022141, 5.75432002,  
5.9320481, 5.52817376, 6.59144949, 6.48350375,  
5.66491191, 6.53488439, 6.99983163, 7.64570321,  
7.15996667, 7.93865443, 7.68370334, 8.19712924,  
8.05447631, 8.50935162, 8.97081677, 9.22425405,  
8.65247979, 9.44221605, 9.58911503, 9.19352861,  
10.13100799, 9.18825973, 10.70621965, 9.93500219,  
10.09973185, 10.75300998, 8.14396165, 9.49105327,  
10.98332921, 11.6505849, 12.08000528, 10.7043178,  
11.44004377, 11.47880595, 9.94124697, 12.55872148,  
13.06078663, 13.74327899, 10.20802826, 12.7754386,  
14.24653384, 11.94305543, 13.69408422, 14.06124278,  
17.36775606, 12.53442022

## X Data

0., 0.40816327, 0.81632653, 1.2244898,  
1.63265306, 2.04081633, 2.44897959, 2.85714286,  
3.26530612, 3.67346939, 4.08163265, 4.48979592,  
4.89795918, 5.30612245, 5.71428571, 6.12244898,  
6.53061224, 6.93877551, 7.34693878, 7.75510204,  
8.16326531, 8.57142857, 8.97959184, 9.3877551,  
9.79591837, 10.20408163, 10.6122449, 11.02040816,  
11.42857143, 11.83673469, 12.24489796, 12.65306122,  
13.06122449, 13.46938776, 13.87755102, 14.28571429,  
14.69387755, 15.10204082, 15.51020408, 15.91836735,  
16.32653061, 16.73469388, 17.14285714, 17.55102041,  
17.95918367, 18.36734694, 18.7755102, 19.18367347,  
19.59183673, 20.

With a possible weight given by

```
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 3., 3., 3., 3., 3., 3., 3.,
3., 3., 3., 3., 3., 3., 3., 3., 3.
```

## Simple Table

Out[3]: statsmodels.iolib.table.SimpleTable

```
import numpy as np
from scipy import stats
import statsmodels.api as sm
import matplotlib.pyplot as plt
from statsmodels.sandbox.regression.predstd import
wls_prediction_std
from statsmodels.iolib.table import (SimpleTable,
default_txt_fmt)
X = np.c_[x, np.ones(nsamp)]
beta = [0.5, -0.01, 5.]
sig = 0.5
w = np.ones(nsamp)
w[nsamp * 6 / 10:] = 3
y_true = np.dot(X, beta)
e = np.random.normal(size=nsamp)
y = y_true + sig * w * e
X = X[:, [0, 2]]
mod_wls = sm.WLS(y, X, weights=1. / w)
res_wls = mod_wls.fit()
print res_wls.summary()
```

## WLS Regression Results

Dep. Variable:	y	R-squared:	0.910			
Model:	WLS	Adj. R-squared:	0.909			
Method:	Least Squares	F-statistic:	487.9			
Date:	Thu, 23 Oct 2014	Prob (F-statistic):	8.52e-27			
Time:	14:19:07	Log-Likelihood:	-46.062			
No. Observations:	50	AIC:	96.12			
Df Residuals:	48	BIC:	99.95			
Df Model:	1					
=====						
	coef	std err	t	P> t	[95.0% Conf. Int.]	
x1	0.4379	0.020	22.088	0.000	0.398	0.478
const	5.2726	0.185	28.488	0.000	4.900	5.645
=====						
Omnibus:	5.040	Durbin-Watson:		2.242		
Prob(Omnibus):	0.080	Jarque-Bera (JB):		6.431		
Skew:	0.024	Prob(JB):		0.0401		
Kurtosis:	4.756	Cond. No.		17.0		
=====						

Suppose instead the variance is thought to vary with x as follows:

```
ww= sigma(x) = 1+ x^2 / 2 Then the weights are 1/sigma as in the
following
>>> ww=1+ (x**2)/2
mod_wls = sm.WLS(y, X, weights=1. / ww)
res_wls = mod_wls.fit()
print res_wls.summary()
```

## WLS Regression Results

===== Dep. Variable: y R-squared: 0.897 Model: WLS Adj. R-squared: 0.895 Method: Least Squares F-statistic: 416.6 Date: Thu, 23 Oct 2014 Prob (F-statistic): 2.64e-25 Time: 16:14:05 Log-Likelihood: 22.720 No. Observations: 50 AIC: -41.44 Df Residuals: 48 BIC: -37.62 Df Model: 1 =====						
	coef	std err	t	P> t	[95.0% Conf. Int.]	
x1	0.4169	0.020	20.412	0.000	0.376	0.458
const	5.3397	0.081	66.311	0.000	5.178	5.502
<hr/> Omnibus: 7.091 Durbin-Watson: 2.264 Prob(Omnibus): 0.029 Jarque-Bera (JB): 9.342 Skew: -0.374 Prob(JB): 0.00936 Kurtosis: 4.981 Cond. No. 4.94 <hr/>						

# Nonparametric Estimation

## Kernel Density Estimation (KDE)

- Also called Parzen–Rosenblatt window method
- KDE estimates the probability density function (pdf) of a random variable
- The iid sample ( $x_1, x_2, \dots, x_n$ ) is drawn from an unknown distribution
- Kernel density estimation smooths data from the sample to make inferences about the population

The KDE for the unknown pdf,  $f$ , is given by:

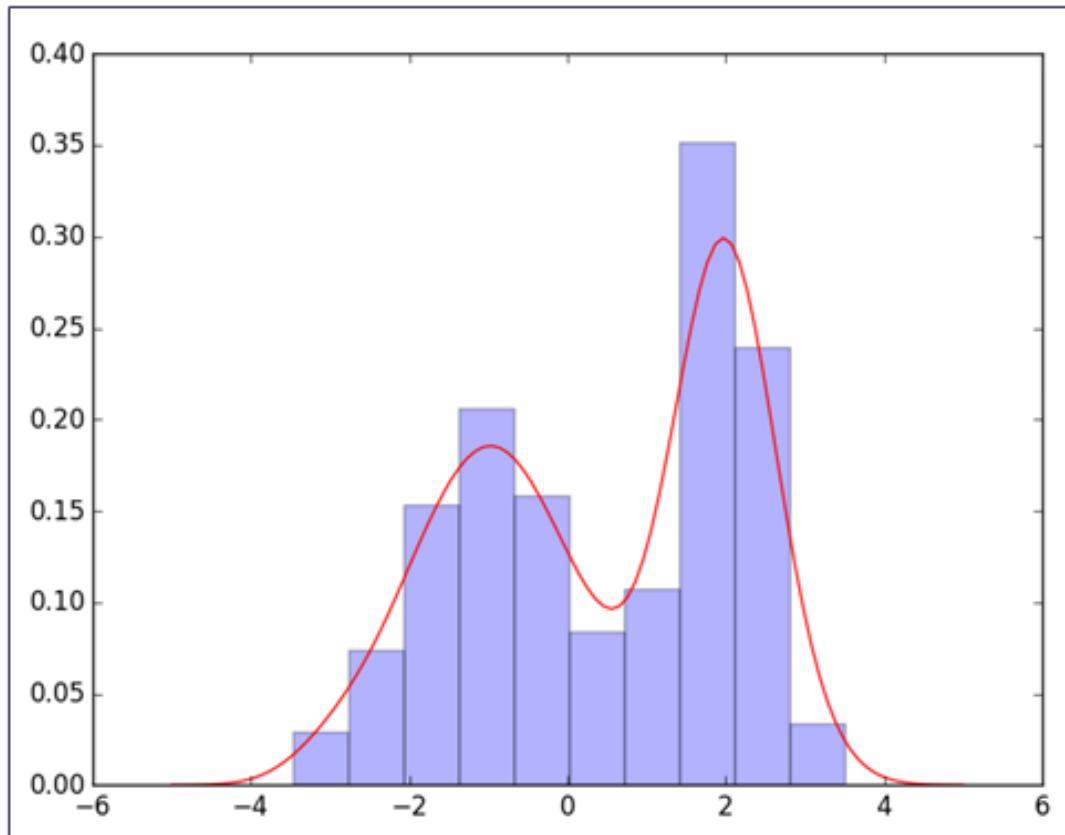
- $$f^*(x) = (1/Nh) \sum_{i=1}^N K([x-x_i]/h)$$
- $K(\cdot)$  is the kernel which is a positive function
- The normal kernel is frequently used
- $K(x) = \phi(x)$ , where  $\phi$  is the standard normal density function
- Bandwidth:  $h$  is called the bandwidth and is a smoothing parameter
- Residual =  $y_t - y_t^*$  where  $y_t$  is the observed (actual) response at time  $t$  and  $y_t^*$  is the estimated or predicted response at time  $t$ .

## Kernel Density Estimation with SciPy

### Python Code:

```
from scipy.stats.kde import gaussian_kde
from scipy.stats import norm
from numpy import linspace,hstack
from pylab import plot,show,hist
# creating data with two peaks
sampD1 = norm.rvs(loc=-1.0,scale=1,size=300)
sampD2 = norm.rvs(loc=2.0,scale=0.5,size=300)
samp = hstack([sampD1,sampD2])
#obtaining the pdf
my_pdf = gaussian_kde(samp)
#plotting the result
x = linspace(-5,5,100)
plot(x,my_pdf(x),'r') # distribution function
hist(samp,normed=1,alpha=.3) # histogram
(array([ 0.02872496,  0.07420615,  0.1531998 ,  0.20586223,
0.15798729,  0.08378114,  0.10771861,  0.35188079,  0.23937468,
0.03351246]), array([-3.46184678, -2.76558825, -2.06932972, -
1.37307119, -0.67681265,  0.01944588,  0.71570441,  1.41196294,
2.10822148,  2.80448001,  3.50073854]))
>>> show()
```

Source: <http://glowingpython.blogspot.ro/2012/08/kernel-density-estimation-with-scipy.html>



## Nonparametric Tests in Python

### Python Code - A Wilcoxon Signed-rank Test for x and y:

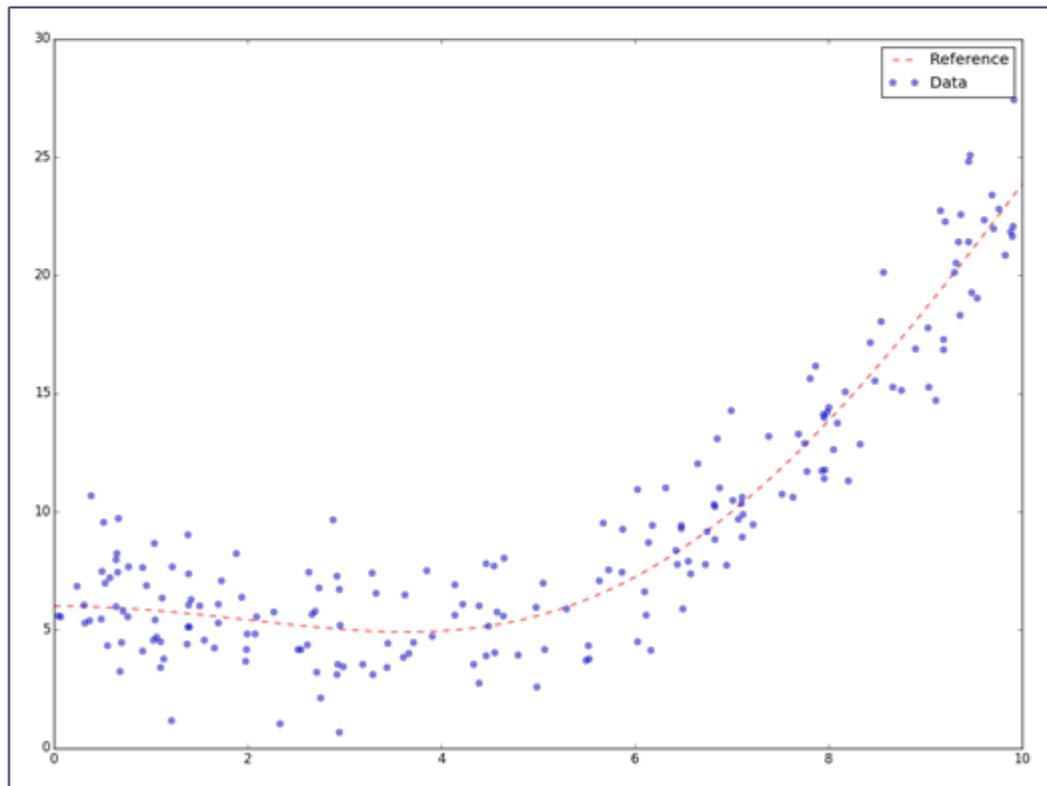
```
>>> import scipy.stats  
>>> x=[0,1,3,2,4,1,3,2,5,1,-1,2,-2,3,1,0,-1,1,2,-1,0,2,3,1,-  
1,1,3,2,4,1,3,2,5,1,-1,2,-2,3,1,0,-1,1,2,-1,0,2,3,1,1]  
>>> y=[1,0,3,2,2,1,3,1,5,1,0,2,-2,3,1,2,-1,1,2,-1,-1,2,3,1,-  
1,2,3,2,4,1.5,3,2,5,0,-1,1,-2,3,-1,0,-1,0,2,-1,1,2,1,1,-1]  
scipy.stats.wilcoxon(x, y)  
Out[7]: (40.5, 0.25063877239660326)
```

Critical values (based on the number of observations, n, and the significance level) of the Wilcoxon signed-rank test can be found at:

[http://facultyweb.berry.edu/vbissonnette/tables/wilcox\\_t.pdf](http://facultyweb.berry.edu/vbissonnette/tables/wilcox_t.pdf)

## Nonparametric Regression

```
Python code:  
import numpy as np  
def f(x):  
...    return 3*np.cos(x/2) + x**2/5 + 3  
xs = np.random.rand(200) * 10  
ys = f(xs) + 2*np.random.randn(*xs.shape)  
import matplotlib.pyplot as plt  
grid = np.r_[0:10:512j]  
plt.plot(grid, f(grid), 'r--', label='Reference')  
plt.plot(xs, ys, 'o', alpha=0.5, label='Data')  
plt.legend(loc='best')
```



## Two-Stage Least Squares

### Instrumental Variables and Two-Stage Least Squares Estimation

Suppose we want to study the relation between a firm's CEO's compensation ( $y$ ) and a firm's board characteristics ( $x$ ). Usually, a linear regression model is used, relating  $y$  and  $x$ , with additional "control variables" ( $W$ ) controlling for other features that make one CEO's compensation different from another. The term  $\epsilon$  represents the effects of individual variation that have not been controlled for with  $W$  or  $x$ . The model is:

$$y = x\beta + W\gamma + \epsilon$$

If the firm's board is selected by the CEO, we have a problem:  $y$  and  $x$  are both endogenous –i.e., influenced by the unobserved CEO's skills. Then,  $\text{Cov}(x, \epsilon) \neq 0$

We suppose that  $y_{2i}$  is also an endogenous variable and  $\text{cov}(y_{2i}, u) \neq 0$ . There may be several reasons for this nonzero covariance that we examine shortly.

Suppose we have a population model (called the *structural equation*) that looks as follows:

$$y_{1i} = \alpha_1 y_{2i} + \beta_0 + \beta_1 x_{1i} + u$$

We suppose that  $y_{2i}$  is also an endogenous variable and  $\text{cov}(y_{2i}, u) \neq 0$ . There may be several reasons for this nonzero covariance. These include:

1. Omitted variables that are correlated with  $y_1$  and  $y_2$ .

2.  $y_2$  is measured with errors.
3.  $y_1$  and  $y_2$  are simultaneously determined as supply and demand equations from economics are simultaneously determined.

## Bias

The OLS multivariate estimator  $\beta^* = (X' X)^{-1} X' Y$  is unbiased because  $\text{cov}(X, u) = 0$ . In the present case the  $\text{cov}(X, u) \neq 0$  and

$$E(b_{OLS}) = \beta + \text{Cov}(X, u) / \text{Var}(X) = \beta + \text{bias}$$

$$\text{plim}(\beta^*_{OLS}) = \beta + \text{Cov}(X, u) / \text{Var}(X) = \beta + \text{bias}$$

Two-stage least squares is designed to get estimators that are consistent (i.e.,  $E(\beta^*) \rightarrow \beta$  as  $N \rightarrow \infty$ ) with accurate estimators when the sample size is large.

We want instrumental variable(s),  $z$ , are variables that are uncorrelated with  $u$  or  $\text{cov}(z, u) = 0$ , but  $y_2$  and  $z$  are correlated once the other independent variables (in our case  $x$ ) are controlled for  $\text{cov}(y_2, z) \neq 0$

Both instrumental variables (IV) and two-stage least squares (2SLS) rely on a reduced form equation. The reduced form equation regresses the second endogenous variable against all instrumental variables ( $z_1, z_2, \dots$ ) in the case of more than one) and the exogenous variables from the population model above. The reduced form equation looks as follows:

$$y_{2i} = \delta_0 + \delta_1 z_i + \delta_2 x_{1i} + v_i \quad (\text{reduced form equation})$$

This can be extended to contain more exogenous and IV variables. Note a test for the requirement that  $\text{cov}(y_2, z) \neq 0$  can be performed using the t test from OLS for  $\delta_1$ .

Steps in IV and 2SLS:

1. Use OLS to estimate the reduced form equation:

$$y_{2i} = \delta^*_0 + \delta^*_1 z_i + \delta^*_2 x_{ii} + v_i$$

The stars \* indicate that these are the estimated parameters from OLS

2. Use these estimated parameters to produce the proxy variable for  $y_2$  (predicted value of  $y_2$ ):

$$y^*_{2i} = \delta^*_0 + \delta^*_1 z_i + \delta^*_2 x_{ii} \quad (\text{proxy variable used instead of } y_2)$$

3. Use the proxy in the original equation:

$$y_{ii} = \alpha_1 y^*_{2i} + \beta_0 + \beta_1 x_{ii} + u$$

In this case we had one endogenous variable (on the right-hand side) and one IV,  $z$ . We say the endogenous variable is *just identified*. When there are more IVs than endogenous variables we say it is *over identified*.

Many exogenous and instrumental variables (one RHS endogenous explanatory):

More generally we might have k x variables and m instruments

$$y_{1i} = \alpha_1 y_{2i} + \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \dots + \beta_k x_{ki} + u$$

Then the reduced-form equation is:

$$y_{2i} = \delta_0 + \delta_1 z_{1i} + \delta_2 z_{2i} + \dots + \delta_k z_{ki} + \phi_1 x_{1i} + \phi_2 x_{2i} + \dots + \phi_m x_{mi} + v_i$$

#### More than one RHS endogenous explanatory variable

Suppose that there is more than one RHS variable. Specifically, assume we have two  $y_2$  and  $y_3$ .

Now

1. Regress  $y_2$  against all the exogenous and instrumental variables and get the predicted values

$$y_{2i} = \delta_0 + \delta_1 z_{1i} + \delta_2 z_{2i} + \dots + \delta_k z_{ki} + \phi_1 x_{1i} + \phi_2 x_{2i} + \dots + \phi_m x_{mi} + v_i$$

and get the predicted values:  $y^*_{2i}$ . To be used in the original equation

2. Do the same for the other RHS endogenous variable

$$y_{3i} = \delta_0 + \delta_1 z_{1i} + \delta_2 z_{2i} + \dots + \delta_k z_{ki} + \phi_1 x_{1i} + \phi_2 x_{2i} + \dots + \phi_m x_{mi} + w_i$$

Again, use predicted values in the original estimating equation.

## Example from Finance: CEO Compensation

Agency theory asserts that there may be a motive for CEOs to take more risks in order to increase compensation. The argument is that risk is a function of compensation. Suppose risk is measured as the percentage financial leverage ratio=operating income/ net income (equals 100 if the firm has no debt). The researcher speculates that

$$\text{Risk} = \alpha_0 + \alpha_1 (\text{Comp}) + \alpha_2 (\text{LTA}) + u$$

Where

Comp: CEO compensation (in thousands)

LTA: log of total assets

But also believes an agency problem exists and Comp is a function of:

$$\text{Comp} = \beta_0 + \beta_1 (\text{risk}) + \beta_2 (\text{Years}) + v$$

Where Years is years on industry experience

Below is the calculation of the risk function using 2SLS for data stored at

d72.csv:

```
>>> import pandas as pd
>>> import numpy as np
>>> import statsmodels.formula.api as smf
>>> import csv
>>> url = 'c:/users/gib/documents/d72.csv'
>>> dat = pd.read_csv(url)
>>> risk=dat['risk']
>>> years=dat['years']
>>> comp=dat['comp']
>>> LTA=dat['LTA']
```

Reduced form equation:

```
n [53]: results = smf.ols('comp ~ LTA + years', data=dat).fit()
```

```
In [54]: print results.summary()
```

## OLS Regression Results

Dep. Variable:	comp	R-squared:	0.752	
Model:	OLS	Adj. R-squared:	0.747	
Method:	Least Squares	F-statistic:	147.5	
Date:	Sun, 07 Sep 2014	Prob (F-statistic):	3.88e-30	
Time:	19:55:36	Log-Likelihood:	-646.92	
No. Observations:	100	AIC:	1300.	
Df Residuals:	97	BIC:	1308.	
Df Model:	2			
coef	std err	t	P> t	[95.0% Conf. Int.]
Intercept	129.4880	38.117	3.397	0.001 53.837 205.139
LTA	58.2993	5.970	9.765	0.000 46.450 70.149
years	19.5025	3.312	5.888	0.000 12.929 26.076
Omnibus:	23.415	Durbin-Watson:	1.171	
Prob(Omnibus):	0.000	Jarque-Bera (JB):	44.448	
Skew:	0.928	Prob(JB):	2.23e-10	
Kurtosis:	5.688	Cond. No.	33.1	

The next step is calculated the predicted values for comp:

In [62]:  $a1=58*LTA$

In [63]:  $a2=19.5*years$

In [64]:  $a3=a1+a2$

In [65]:  $a4=129.5+a3$  (or just simply type:  $a4=[129.5 + (58*LTA) + (19.5*years)]$  =predicted comp )

Finally run the second stage using OLS:

In [66]: `results = smf.ols('risk ~ a4 + LTA', data=dat).fit()`

In [67]: `print results.summary()`

### OLS Regression Results

Dep. Variable:	risk	R-squared:	0.780		
Model:	OLS	Adj. R-squared:	0.776		
Method:	Least Squares	F-statistic:	172.2		
Date:	Sun, 07 Sep 2014	Prob (F-statistic):	1.22e-32		
Time:	20:34:19	Log-Likelihood:	-366.52		
No. Observations:	100	AIC:	739.0		
Df Residuals:	97	BIC:	746.8		
Df Model:	2				
=====					
	coef	std err	t	P> t	[95.0% Conf. Int.]
=====					
Intercept	100	.4622	2.958	33.963	0.000 94.591 106.333
a4	-0.0606	0.010	-5.886	0.000	-0.081 -0.040
LTA	-0.4518	0.858	-0.527	0.600	-2.155 1.251
=====					
Omnibus:		23.011			Durbin-Watson: 1.197
Prob(Omnibus):		0.000			Jarque-Bera (JB): 42.125
Skew:		-0.929			Prob(JB): 7.12e-10
Kurtosis:		5.580			Cond. No. 2.39e+03
=====					

Conclusion if compensation increases by 100k risk variable increases by 6% (debt increases) on the scale.

**Note:** Interpret the t stats carefully since the standard errors are based on the wrong residual error =  $y_1 - \beta_0 - \beta_1 y_2 - \beta_2 x_1$  (predicted values of comp) when it should be error =  $y_1 - \beta_0 - \beta_1 y_2 - \beta_2 x_1$  (actual comp values)

## The Wu-Hausman Test for Endogeneity

(Hausman specification test or Durbin–Wu–Hausman test) is a statistical hypothesis test for endogeneity comparing the 2SLS and the OLS estimators.

1. Estimate the first stage (reduced form)

$$y^*2 = \delta^*0 + \delta^*1z + \delta^*2 x1 + v$$

This would be comp against LTA and years in the example above.

Calculate the residuals by subtracting the predicted value (pred) from the actual LHS value (comp):  $v^* = y2 - [\delta^*0 + \delta^*1 z + \delta^*2 x1]$

```
>>> pred=129.5 +(58*LTA) + (19.5*years)
>>> resid=comp - pred
```

2. The final stage is to regress risk against comp, LTA and

$$y1 = \beta_0 + \beta_1 y2 + \beta_2 x1 + \beta_3 v^* + u$$

**Note:** regress  $y_2$  not the predicted value of  $y_2$

```
results = smf.ols('risk ~ comp + LTA + resid', data=dat).fit()
print results.summary()
```

## OLS Regression Results

Dep. Variable:		risk	R-squared:		0.998
Model:		OLS	Adj. R-squared:		0.998
Method:		Least Squares	F-statistic:		1.383e+04
Date:		Tue, 09 Sep 2014	Prob (F-statistic):		2.15e-126
Time:		10:41:41	Log-Likelihood:		-138.70
No. Observations:		100	AIC:		285.4
Df Residuals:		96	BIC:		295.8
Df Model:		3			
		coef	std err	t	P> t  [95.0% Conf. Int.]
Intercept	100.4605	0.305	329.694	0.000	99.856 101.065
comp	-0.0605	0.001	-57.134	0.000	-0.063 -0.058
LTA	-0.4342	0.088	-4.913	0.000	-0.610 -0.259
resid	0.0003	0.001	0.244	0.808	-0.002 0.003
Omnibus:		206.575	Durbin-Watson:		1.088
Prob(Omnibus):		0.000	Jarque-Bera (JB):		31499.439
Skew:		-9.069	Prob(JB):		0.00
Kurtosis:		88.035	Cond. No.		2.45e+03

# Ridge Regression

## Ridge Regression

Ridge regression penalizes the size of the regression coefficients.

The RR estimates,  $\beta^*$ , finds the values for  $\beta$  that minimizes

$$\sum_{i=1}^n (y_i - x_i^T \beta)^2 + \lambda \sum_{j=1}^k \beta_j^2$$

The solution in matrix form is

$$\beta^* = (X^T X + \lambda I)^{-1} X^T y$$

$I$  is the identity (unit) matrix. If  $\lambda = 0$  this reduces to the ols estimate for  $\beta$ .

The variance of the estimates is:

$$\text{Var}(\beta^*) = \sigma^2 (X^T X + \lambda I)^{-1} X^T X (X^T X + \lambda I)^{-1}$$

The total variance is a monotone decreasing sequence with respect to  $\lambda$ .

The bias of the regression estimates is

$$\text{Bias}(\beta^*) = -\lambda (X^T X + \lambda I)^{-1} \beta$$

Which is zero if  $\lambda = 0$ .

## Python Code:

```

import numpy as np
import pandas as pd
from sklearn import linear_model
clf = linear_model.LinearRegression()
X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])
y = np.array([1, 1, 2, 2])
clf.fit(X, y)          # for OLS
clf.coef_      # returns OLS coefficients : array([
5.55111512e-17,    5.00000000e-01])
clf.intercept_   # returns the intercept value of the
fitted model 1.5

```

## Partial Least Squares

- PLS (alternative to ordinary least squares (OLS) regression) is used when multicollinearity (when predictor variables are highly correlated) or when the number of predictors exceeds the number of cases (see PCA).
- finds a linear regression model by projecting the predicted variables and the observable variables into a new space. Because  $y$  and  $x$  data are projected to new spaces, it is sometimes called a bilinear factor model.
- PLS combines features of principal components analysis and multiple regression. It first extracts a set of latent factors that explain as much of the covariance as possible between the independent and dependent variables. Then a regression step predicts values of the dependent variables using the decomposition of the independent variables.

PLS steps:

1. Model we have is  $Y = X\beta + e$  (  $e$  is error term)
2. Compute a factor score matrix :  $T = XW$  ( where  $W$  are the weights)
3. Regress  $Y = T Q + e$  ( $Q$  is the matrix of regression coefficients [loadings as discussed in Linalg] for  $T$ )
4. This is equivalent to  $Y = XWQ + e$

## Running PLS in Python

There are three survey variables (scale of 0 to 10) called  $x_1$   $x_2$   $x_3$  being used to predict respondent's income,  $y$ . The  $x$  variables are thought to be multicollinear. PLS uses `dat d5.PCA4` to predict regression coefficients for  $x_1$   $x_2$  and  $x_3$ :

```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
import pylab as pl
from sklearn.pls import PLSRegression
url = 'c:/users/gib/documents/d5.PCA4.csv'
dat = pd.read_csv(url)
X=dat[['x1','x2','x3']]
Y = dat['y']
pls2 = PLSRegression(n_components=1)      # uses only the first
factor or principal component
pls2.fit(X, Y)
print np.round(pls2.coefs, 1)      # returns the beta
coefficients for x1 x2 x3
pls2.predict(X)                  # returns predicted
values for Y
print np.round(pls2.coefs, 1)  #( for the d5.PCA4 data)
[[ 1.9]
[ 1.8]
[ 1.7]]
pls2.predict(X)      # predicted values of y
Out[84]:
array([[ 17.72703409],
       [ 30.02843662],
       [ 34.27239941],
       [ 35.65163526],
       [ 24.75555554],
       [ 34.14792457],
       [ 20.71134849],
       [ 25.05965915],
       [ 39.03844115],
       [ 20.88799726],
       [ 27.84123781],
       [ 30.75813869],
       [ 24.37617102],
       [ 37.81274711],
       [ 23.15047698],
       [ 41.07507815],
       [ 39.09061508],
       [ 37.7374662 ],
       [ 35.60244132],
       [ 23.07519608]])
```

*Prediction function:* The predict function in PLS can be used to predict Y based on hypothetical values for x1 x2 and x3:g=[0,0,0]

```
g=[0,0,0]
pls2.predict(g)
Out[99]: array([ 3.26749971])
h=[3,5,3]
pls2.predict(h)
Out[101]: array([ 23.07519608])
i=[10,10,10]
pls2.predict(i)
Out[103]: array([ 57.24105896])
```

# Estimation with Maximum Likelihood

## Theory of MLE

We have a random sample ( $x_1, x_2, \dots, x_N$ ) from a pdf with parameter  $\beta$ . We want to find an estimate of  $\beta$  that maximizes the likelihood of observing the random sample. The pdf is  $f(x_i, \beta)$ . The joint probability density function of the random sample is the likelihood function:

$$L(\beta) = f(x_1, \beta) * f(x_2, \beta) * \dots * f(x_N, \beta) = \prod_{i=1}^N f(x_i, \beta)$$

In practice we take the monotonic log transformation of  $L$ :

$$\text{Max } \log(L)$$

## Binomial MLE

$$f(x) = \left( \frac{n!}{x!(n-x)!} \right) p^x (1-p)^{n-x}$$

$$L(p) = \prod_{i=1}^n f(x_i) = \prod_{i=1}^n \left( \frac{n!}{x_i!(n-x_i)!} \right) p^{x_i} (1-p)^{n-x_i}$$

$$L(p) = \left( \prod_{i=1}^n \left( \frac{n!}{x_i!(n-x_i)!} \right) \right) p^{\sum_{i=1}^n x_i} (1-p)^{n-\sum_{i=1}^n x_i}$$

$$\ln L(p) = \sum_{i=1}^n x_i \ln(p) + \left( n - \sum_{i=1}^n x_i \right) \ln(1-p)$$

$$\frac{d \ln L(p)}{dp} = \frac{1}{p} \sum_{i=1}^n x_i + \frac{1}{1-p} \left( n - \sum_{i=1}^n x_i \right) = 0$$

$$(1 - \hat{p}) \sum_{i=1}^n x_i + p \left( n - \sum_{i=1}^n x_i \right) = 0$$

$$\hat{p} = \frac{\sum_{i=1}^n x_i}{n} = \frac{k}{n}$$

## Bernoulli: How to Generate a Random Sample in Python Using the Bernoulli PDF

```

import scipy.stats
#to generate data using the Bernoulli distribution is using
Bernoulli;
b=scipy.stats.bernoulli(.5)      # heads/tails or male /female
probability=.5

xs = b.rvs(100)                  # draw a sample of size= 100

xs
Out[36]:
array([1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1,
0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1,
1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0,
0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0])
#To get the mean of this sample type:
pest = np.mean(xs)

Out[39]:
0.54

```

## Poisson Distribution Data Creation and MLE

```

import numpy as np
from scipy.stats import poisson

c=scipy.stats.poisson(10)          #lambda equals 10 from
the poisson

xs = c.rvs(100)                   # generates a 100 sample

xs
Out[24]:
array([ 6, 12,  8,  8, 16,  6,  8, 12, 12, 12, 13, 15, 16, 16,
18,  8, 10,
       12, 20, 10, 13, 17,  6, 14, 12, 12,  8,  7,  8, 10, 15,
16, 15, 10,
       8, 12, 10, 12, 12, 14,  7,  6,  9, 14, 11,  9, 14,  8,
9, 11, 14,
....])

```

## MLE Poisson Distribution

Observations:  $X_1, X_2, X_3, \dots, X_n$

$$f(x) = \frac{\lambda^x e^{-\lambda}}{x!} \quad x = 0, 1, 2, \dots$$

$$L(\lambda) = \prod_{i=1}^n \frac{\lambda^{x_i} e^{-\lambda}}{x_i!} = e^{-n\lambda} \frac{\lambda^{\sum_1^n x_i}}{\prod_{i=1}^n x_i}$$

$$\ln L(\lambda) = -n\lambda + \sum_1^n x_i \ln(\lambda) - \ln \left( \prod_{i=1}^n x_i \right)$$

$$\frac{d \ln L(\lambda)}{d\lambda} = -n + \sum_1^n x_i \frac{1}{\lambda}$$

$$\hat{\lambda} = \frac{\sum_{i=1}^n x_i}{n}$$

If the xs data were given and we wanted an MLE for the xs data for lambda it would be:

```
lambda=np.mean(xs)
lambda
Out[26]: 10.539999999999999
```

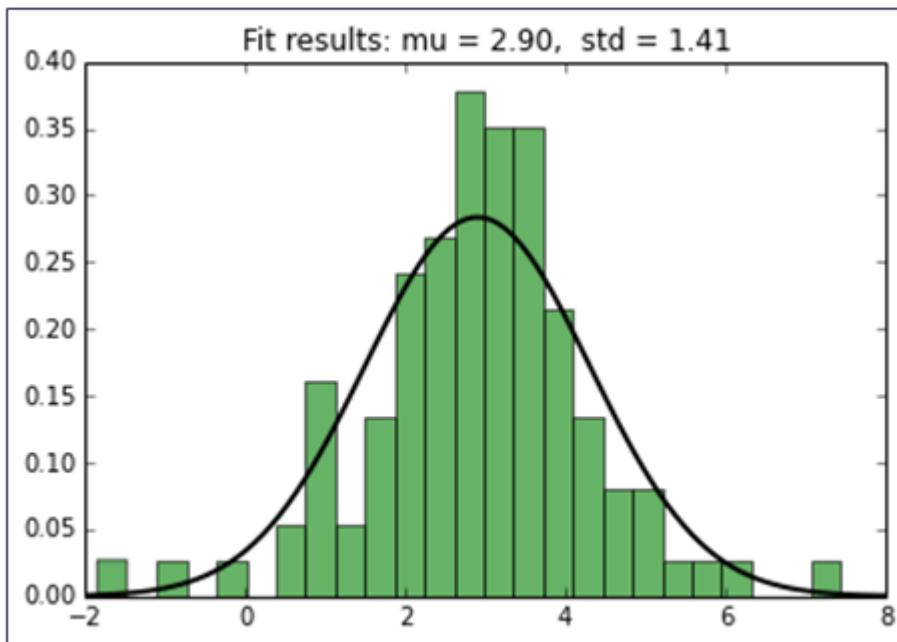
## How to Generate a Random Sample in Python Using the Normal PDF

```
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt
c=scipy.stats.norm(3,1.5)      # normal distribution from scipy
xs = c.rvs(100)
xs
Out[44]:
array([ 1.88877934,  3.10936086,  3.60462894,  5.20789405,
3.46107633,
       2.08316199,  2.41257028,  3.20996716,  3.14019124,
5.1893839 ,
       5.09302939,  2.46159611,  2.17703681, -0.83558191,
2.17661938,...]
Fit the data using morm.fit
# Fit a normal distribution to the data:
```

## Graph

Of course, we knew this data was normally distributed. If we didn't we would have plotted the data first as follows:

```
# Plot the histogram.  
plt.hist(data, bins=25, normed=True, alpha=0.6, color='g')  
  
# Plot the PDF.  
xmin, xmax = plt.xlim()  
x = np.linspace(xmin, xmax, 100)  
p = norm.pdf(x, mu, std)  
plt.plot(x, p, 'k', linewidth=2)  
title = "Fit results: mu = %.2f, std = %.2f" % (mu, std)  
plt.title(title)  
  
plt.show()
```



## Exponential Distribution MLE

Let  $X_1, X_2, X_3, \dots, X_n$  be a random sample from the exponential distribution with p.d.f.

$$f(x; \theta) = \frac{1}{\theta} e^{-\frac{x}{\theta}} \quad 0 < x < \infty, \theta \in \Omega = \{\theta | 0 < \theta < \infty\}$$

The likelihood function is given by:

$$L(\theta) = L(\theta; x_1, x_2, \dots, x_n) = \left( \frac{1}{\theta} e^{-\frac{x_1}{\theta}} \right) \left( \frac{1}{\theta} e^{-\frac{x_2}{\theta}} \right) \dots \left( \frac{1}{\theta} e^{-\frac{x_n}{\theta}} \right) = \frac{1}{\theta^n} \exp \left( -\frac{\sum_1^n x_i}{\theta} \right)$$

Taking log, we get,

$$\ln L(\theta) = -n \ln(\theta) - \frac{1}{\theta} \sum_1^n x_i, \quad 0 < \theta < \infty$$

Differentiating the above expression, and equating to zero, we get

$$\frac{d[\ln L(\theta)]}{d\theta} = \frac{-n}{\theta} + \frac{1}{\theta^2} \sum_1^n x_i = 0$$

The solution of equation for  $\theta$  is:

$$\theta = \frac{\sum_1^n x_i}{n}$$

Thus, the maximum likelihood estimator of  $\Theta$  is

$$\Theta = \frac{\sum_1^n X_i}{n}$$

## Geometric Distribution MLE

Let  $X_1, X_2, X_3, \dots, X_n$  be a random sample from the geometric distribution with p.d.f.

$$f(x; p) = (1 - p)^{x-1} p, x = 1, 2, 3, \dots$$

The likelihood function is given by:

$$L(p) = (1 - p)^{x_1-1} p (1 - p)^{x_2-1} p \dots (1 - p)^{x_n-1} p = p^n (1 - p)^{\sum_1^n x_i - n}$$

Taking log,

$$\ln L(p) = n \ln p + \left( \sum_1^n x_i - n \right) \ln(1 - p)$$

Differentiating and equating to zero, we get,

$$\frac{d[\ln L(p)]}{dp} = \frac{n}{p} - \frac{(\sum_1^n x_i - n)}{(1 - p)} = 0$$

Therefore,

$$p = \frac{n}{(\sum_1^n x_i)}$$

So, the maximum likelihood estimator of P is:

$$P = \frac{n}{(\sum_1^n X_i)} = \frac{1}{X}$$

This agrees with the intuition because, in  $n$  observations of a geometric random variable, there are  $n$  successes in the  $\sum_1^n X_i$  trials. Thus the estimate of  $p$  is the number of successes divided by the total number of trials.

Types of PDF's that can be Imported into Python are as Follows:

```
| from scipy.stats import norm  
| from scipy.stats import Bernoulli
```

Suppose we had the Data and Suspected it was Normal, it can be Fitted as Follows:

```
| from scipy.stats import norm  
| mu, std = norm.fit(xs)  
| mu  
| Out[48]:  
| 2.9  
  
| std  
| Out[49]:  
| 1.4
```

## Python Example:

```

from __future__ import division
from scipy.stats import bernoulli
import numpy as np

p_true=1/2 # this is the value we will try to estimate from the observed
            data
fp=bernoulli(p_true)

def sample(n=10):
    'simulate coin flipping'
    return fp.rvs(n)# flip it n times

xs = sample(100) # generate some samples

import sympy
from sympy.abc import x, z
p=sympy.symbols('p',positive=True)

L=p**x*(1-p)**(1-x)
J=np.prod([L.subs(x,i) for i in xs]) # objective function to maximize

logJ=sympy.expand_log(sympy.log(J))
sol=sympy.solve(sympy.diff(logJ,p),p)[0]

x=linspace(0,1,100)
plot(x,map(sympy.lambdify(p,logJ,'numpy'),x),sol,logJ.subs(p,sol),'o',
      p_true,logJ.subs(p,p_true),'s')
xlabel('$p$',fontsize=18)
ylabel('Likelihood',fontsize=18)
title('Estimate not equal to true value',fontsize=18)

```

# Modeling Stock Returns

## Introduction

Quantitative analysts have been trying for decades to find an efficient way of forecasting stock returns. This topic is presented in thousands of working papers, articles, books and websites.

Can we really predict stock returns?

Sometimes we can, sometimes we cannot.

If stock returns follow a smooth process, we can forecast the evolution accurately. If returns tend be a random process, then it is sometimes impossible to predict and we have to rely on our “feel of the data” (expert judgement). Certain models offer us some answers regarding modeling stock returns.

## Prices, Dividends and Returns with Constant Discount Rates

$$R_{t+1} = \frac{P_{t+1} + D_{t+1}}{P_t} \quad (1)$$

$R_{t+1}$  – net simple return on a stock at time  $t+1$

$P_{t+1}$  – stock price at time  $t+1$

$P_t$  – stock price at time  $t$

$D_{t+1}$  – dividend at time  $t+1$

$E_t R_{t+1}$  – the expected return on the stock

The expected return is assumed constant:

$$E_t R_{t+1} = R \quad (2)$$

Then

$$P_t = E_t \left[ \frac{P_{t+1} + D_{t+1}}{1+R} \right] \quad (3)$$

Solving forward for  $K$  periods:

$$P_t = E_t \left[ \sum_{k=1}^K \left( \frac{1}{1+R} \right)^k D_{t+k} \right] + E_t \left[ \left( \frac{1}{1+R} \right)^K P_{t+K} \right] \quad (4)$$

If  $K \rightarrow \infty$  then (4) becomes:

$$E_t \left[ \left( \frac{1}{1+R} \right)^K P_{t+K} \right] \rightarrow 0$$

## Random Walk or Martingale Model of Stock Prices

$$P_t = E_t \left[ \sum_{k=1}^{\infty} \left( \frac{1}{1+R} \right)^k D_{t+k} \right] \quad (5)$$

If we reinvest all dividends in buying more shares, the number of shares we own is the following:

$$N_{t+1} = N_t \left( 1 + \frac{D_{t+1}}{P_{t+1}} \right) \quad (6)$$

$N_t$  – number of stocks we own at time  $t$

The discounted value of the portfolio is the following:

$$M_t = \frac{N_t P_t}{(1+R)^t} \quad (7)$$

## Gordon Growth Model

Named after Myron Gordon, although it was pioneered by John Burr Williams

Assumes that dividends grow at a constant rate

$$E_t D_{t+k} = (1 + G)^{k-1} E_t D_{t+1} \quad (8)$$

$$P_t = \frac{E_t D_{t+1}}{R - G} \quad (9)$$

(9) can also be written as following:

$$\frac{D}{P} = R - G \quad (10)$$

D denotes the next-period dividend.

## Autoregressive Processes and Random Walks

We can forecast stock returns assuming that they follow an AR(1) process.

$y$  – stock returns

$y_t = \alpha y_{t-1} + \varepsilon_t$  – AR(1) process

$\varepsilon_t$  is normally distributed with mean 0 and variance  $\sigma^2_\varepsilon = 1$

$\varepsilon_t$  and  $\varepsilon_s$  are independent for all  $t \neq s$

$\alpha$  estimated from the historical data on  $y$

- Random walk:  $\alpha = 1$  – the current stock return provides no information about the future movement of the return
- AR(1):  $-1 < \alpha < 1$  – mean-reverting process. If the stock provides a high return currently, it is expected that the return will post a downward trend in the future.

### Example:

Microsoft stock return – AR(1) process.

$\alpha = 0.75$  - Microsoft stock provides a daily return of 1 % currently. It is expected to provide a daily return of 0.75 % the next day

$\alpha = 1$  – Microsoft stock return is a random process. Microsoft stock provides a daily return of 1 % currently. We cannot forecast the daily return for the next day.

## VAR Methodology

We can forecast stock returns using a vector autoregressive (VAR) model.

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{e}_{t+1} \quad (11)$$

$\mathbf{x}_t$  – vector of state variables including stock returns

$$E_t \mathbf{x}_{t+1+j} = \mathbf{A}^{j+1} \mathbf{x}_t \quad (12)$$

## Variables that Explain Expected Stock Return

- Lagged stock return
- Dividend yield
- P/E ratio
- Trading volume
- Default spread
- Yield on T-bill
- Change in yield of T-bill
- Term spread
- yield spread between overnight fixed income security and T-bill
- January dummy
- Growth rate of industrial production
- Change in inflation or unexpected inflation

# Nonlinear Models

## Nonlinear Model Regression

- Nonlinear regression analysis occurs when data is fit to a model expressed as a non-linear mathematical function.
- Nonlinear regression uses an iterative approach.
- A model is nonlinear if a derivative of  $y$  with respect to a model parameter depends on a model parameter

## Methods Used for Fitting the Data

- Grid search
- Steepest descent
- Marquardt algorithm
- Gauss-Newton algorithm
- Newton-Raphson

## Non-linear Growth Models

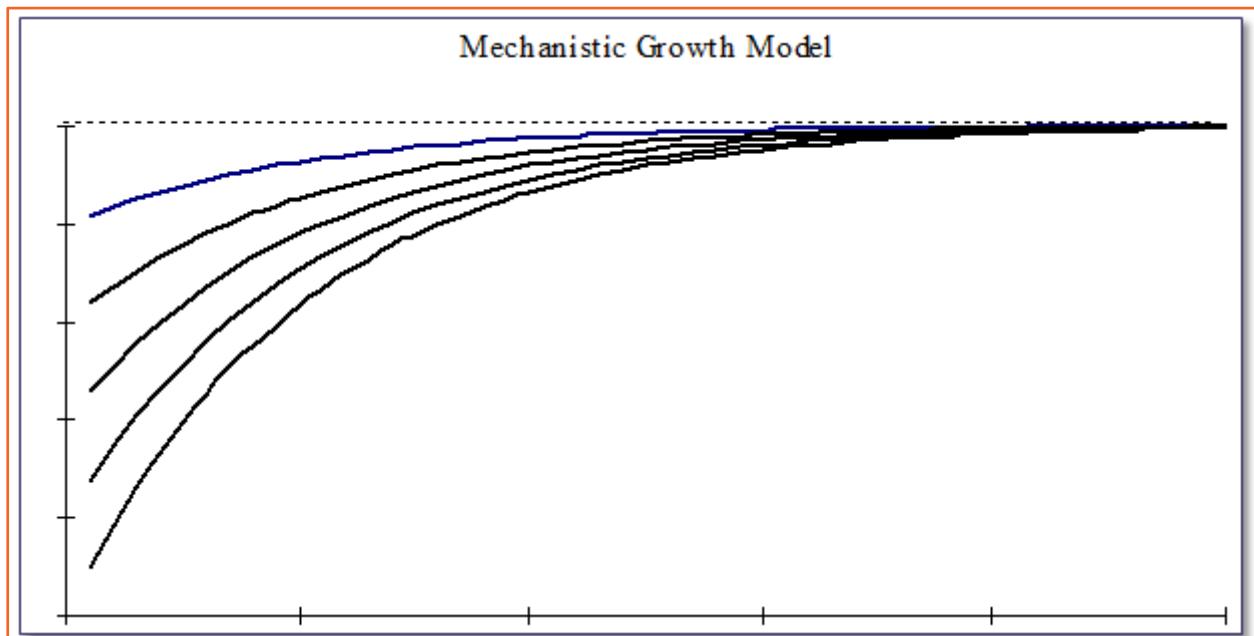
Many models cannot be transformed into a linear model

## The Mechanistic Growth Model

Equation

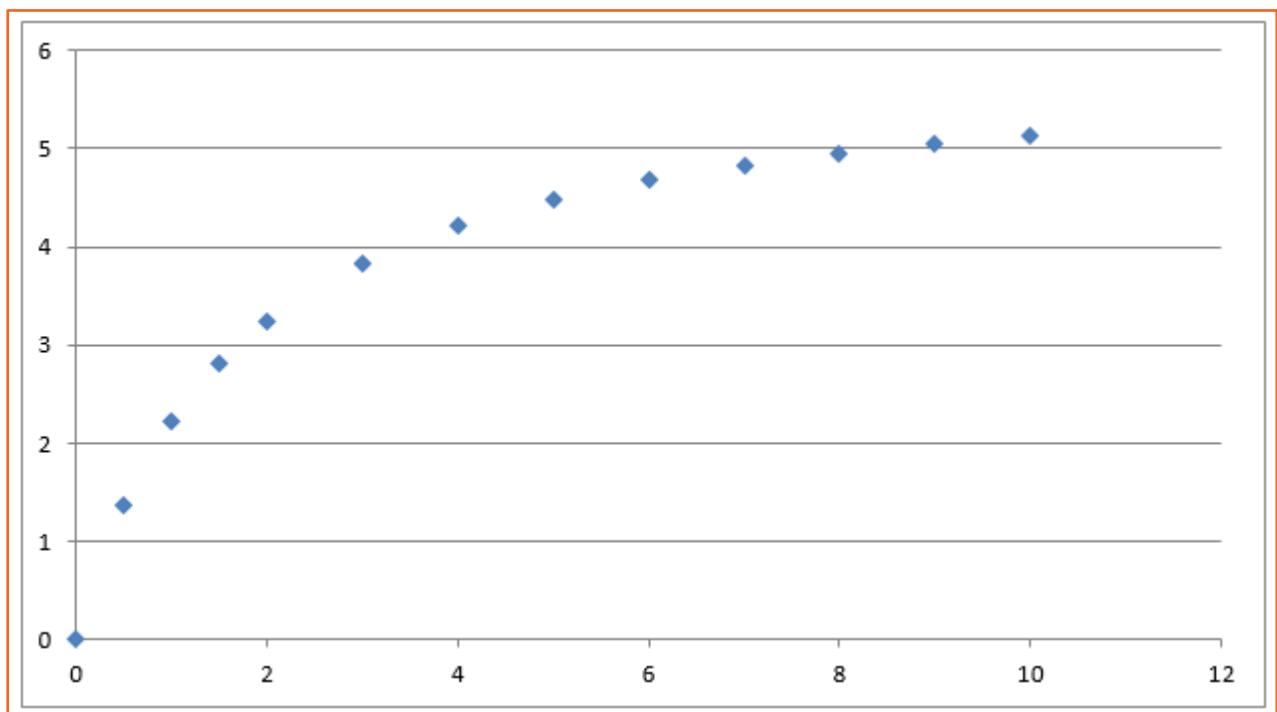
$$Y = \alpha(1 - \beta e^{-kx}) + \varepsilon$$

or (ignoring  $\varepsilon$ ) "rate of increase"  $\frac{dY}{dx} = k(\alpha - Y)$



## Saturation Growth Model

- $$Y = \frac{ax}{b+x}$$
- a and b are parameters to be estimated
- Example suppose that you knew (we shall estimate this using Python later) that a=6 and b=1.7. Then you can get a graph of the relationship between y and x such as in the following graph:

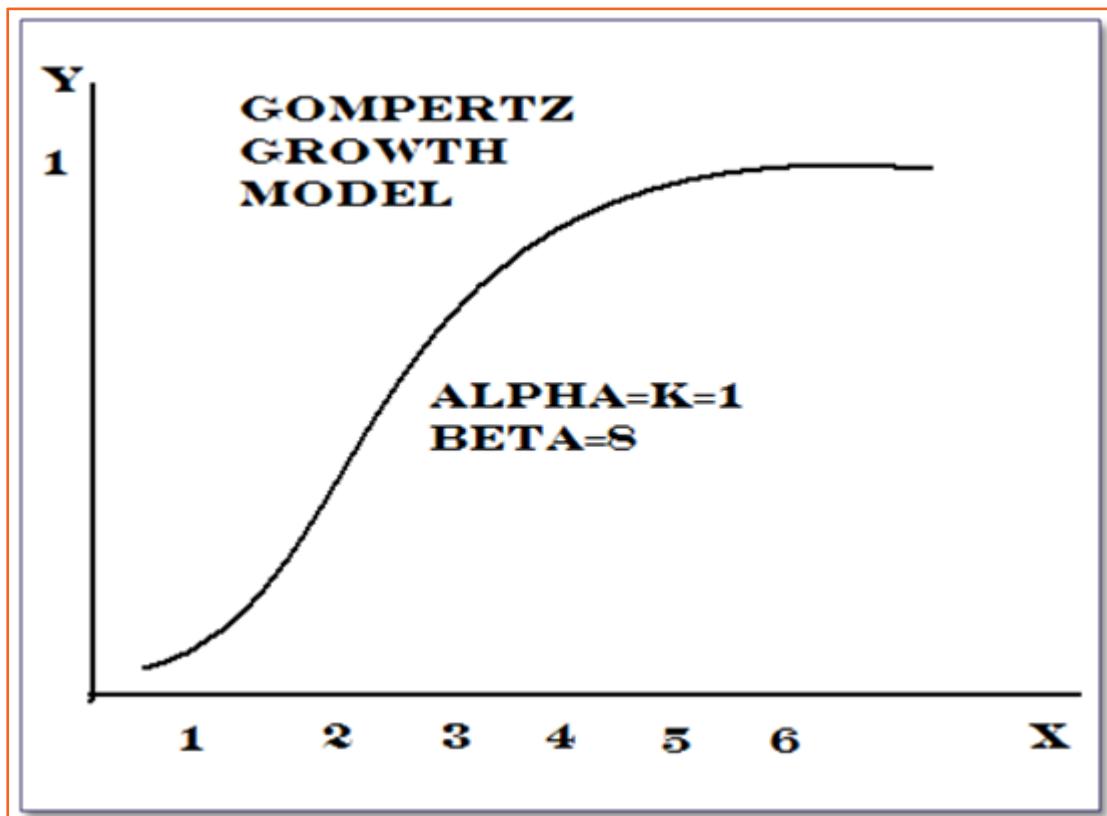


## Gompertz Growth Graph

$$Y = \alpha e^{-\beta e^{-kx}} + \varepsilon$$

With a “rate of increase in Y”

$$\frac{dY}{dx} = kY \ln\left(\frac{\alpha}{Y}\right)$$



## Polynomial Fitting: Example 1

```
import numpy as np
import pandas as pd
import scipy.optimize as optimize
import matplotlib.pyplot as plt
# theory tells us the function should be of the form y=a exp(-bx) -c as in func below:
def func(x,a,b,c):
    return a*np.exp(-b*x)-c

x data called xdata

2.3400000e-01,    4.6800000e-01,    7.0200000e-01,
9.3600000e-01,    1.1700000e+00,    1.4040000e+00,
1.6380000e+00,    1.8720000e+00,    2.1060000e+00,
```

```

2.34000000e+00,    4.68000000e+00,    7.02000000e+00,
9.36000000e+00,    1.17000000e+01,    1.40400000e+01,
1.63800000e+01,    1.87200000e+01,    2.10600000e+01,
2.34000000e+01,    4.68000000e+01,    7.02000000e+01,
9.36000000e+01,    1.17000000e+02,    1.40400000e+02,
1.63800000e+02,    1.87200000e+02,    2.10600000e+02,
2.34000000e+02,    4.68000000e+02,    7.02000000e+02,
9.36000000e+02,    1.17000000e+03,    1.40400000e+03,
1.63800000e+03,    1.87200000e+03,    2.10600000e+03,
2.34000000e+03])

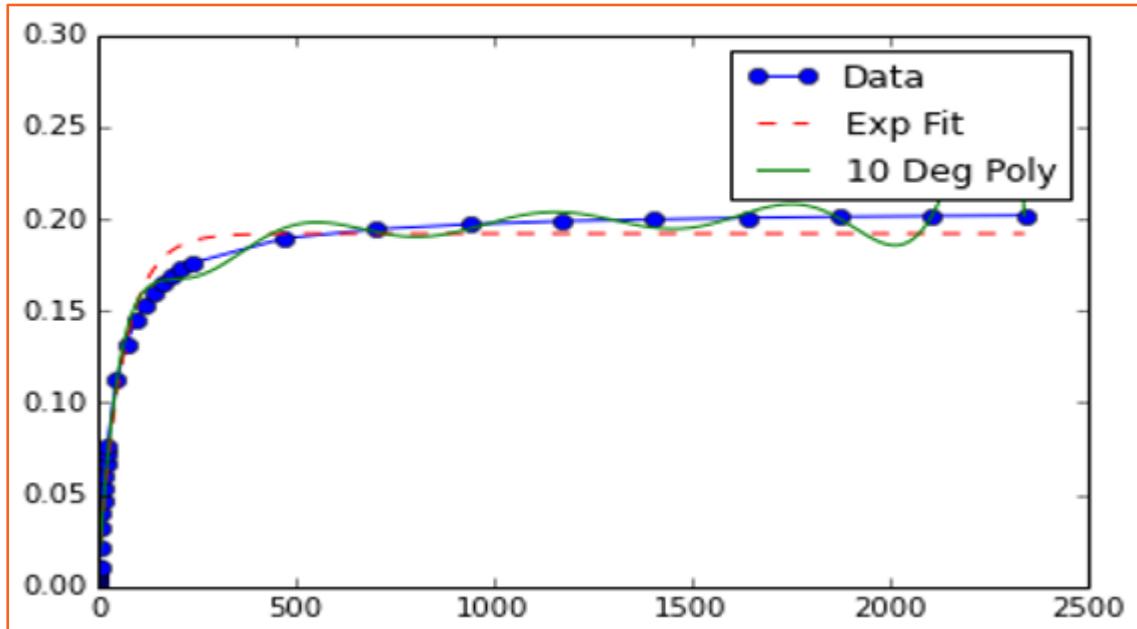
y data called ydata

0.00140041,  0.00248441,  0.0033578 ,  0.00481614,  0.00595956,
0.00743808,  0.00781616,  0.00909158,  0.01038178,  0.01067379,
0.02095667,  0.03188081,  0.04021974,  0.04616741,  0.05349782,
0.05978192,  0.06654972,  0.07173193,  0.07652576,  0.11274733,
0.13148609,  0.1443491 ,  0.15342685,  0.16009537,  0.16492582,
0.16920322,  0.17293625,  0.17567369,  0.18924931,
0.19427476, 0.19695312,  0.1986319 ,  0.19958502,  0.20045236,
0.20104401,  0.20143822,  0.20177028

trialX = np.linspace(xdata[0], xdata[-1], 1000)
fitted = np.polyfit(xdata, ydata, 10)[:-1]
y = np.zeros(len(trialX))
for i in range(len(fitted)):
    y += fitted[i]*trialX**i
popt, pcov = optimize.curve_fit(func, xdata, ydata)      # popt
gives the estimated parameters based on x and y

[-0.1842403   0.01609788 -0.19188716]
# Are the estimated parameters a=-.18424 b=.016097 and c= -
.191887
yEXP = func(trialX, *popt)
plt.figure()
plt.plot(xdata, ydata, label='Data', marker='o')
plt.plot(trialX, yEXP, 'r-',ls='--', label="Exp Fit")
plt.plot(trialX, y, label = '10 Deg Poly')
plt.legend()
plt.show()  # will show the graph below

```



## Nonlinear Data Fitting

Fit the following function (this is courtesy of Mike Croucher see further at:  
<http://www.walkingrandomly.com/?p=5215>)

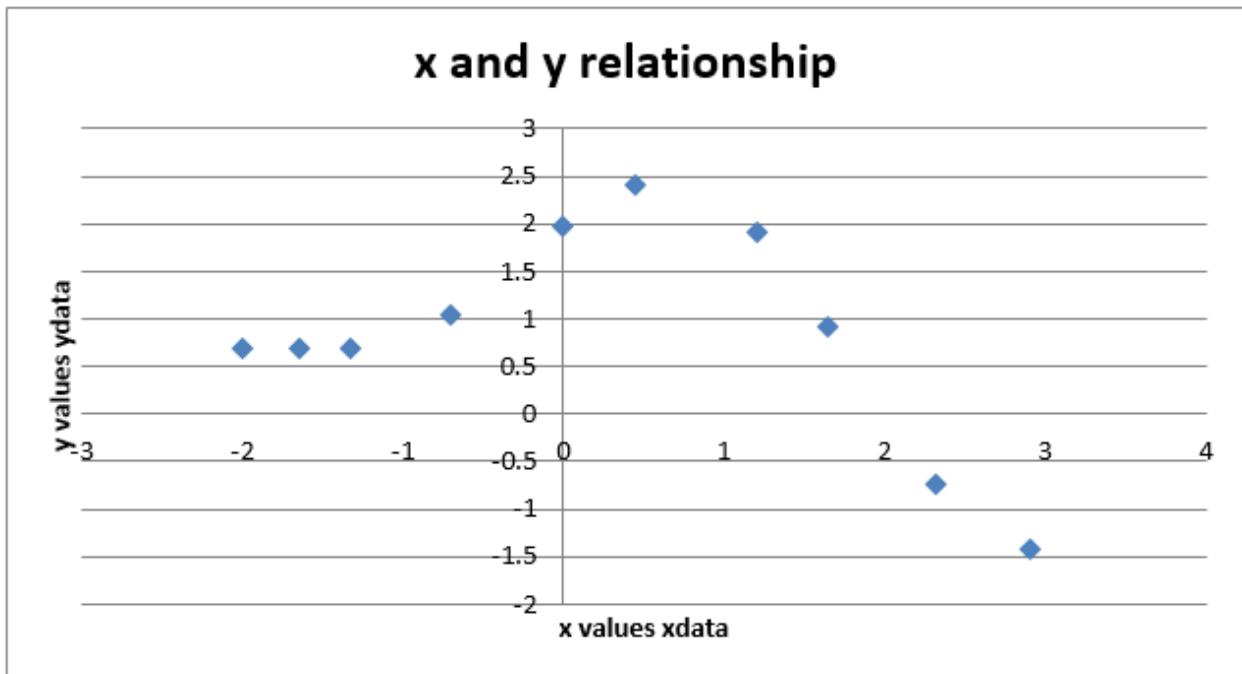
$$y = F(\beta_1, \beta_2, x) = \beta_1 \cos(\beta_2 x) + \beta_2 \sin(\beta_1 x)$$

Using the following x y data:

$$X = [-2, -1.64, -1.33, -0.7, 0, 0.45, 1.2, 1.64, 2.32, 2.9]$$

$$y = [0.699369, 0.700462, 0.695354, 1.03905, 1.97389, 2.41143, 1.91091, 0.919576, -0.730975, -1.42001]$$

## Data Graph We Think Involves Sin and Cos



## Python Code:

```
In [32]: import numpy as np
In [33]: Import pandas as pd
In [34]: from scipy.optimize import curve_fit
In [35]: xdata = np.array([-2,-1.64,-1.33,
0.7,0,0.45,1.2,1.64,2.32,2.9])
ydata=np.array([0.699369,0.700462,0.695354,1.03905,1.97389,2.41
143,1.91091,0.919576,-0.730975,-1.42001])
In [36]: def func(x, p1,p2):
....:     return p1*np.cos(p2*x) + p2*np.sin(p1*x)
....: popt, pcov = curve_fit(func, xdata,
ydata,p0=(1.0,0.2))
popt
Out[37]: array([ 1.88185099,  0.70022986]) # these are the
estimates for p1 and p2
```

## Python: Extracting Coefficients

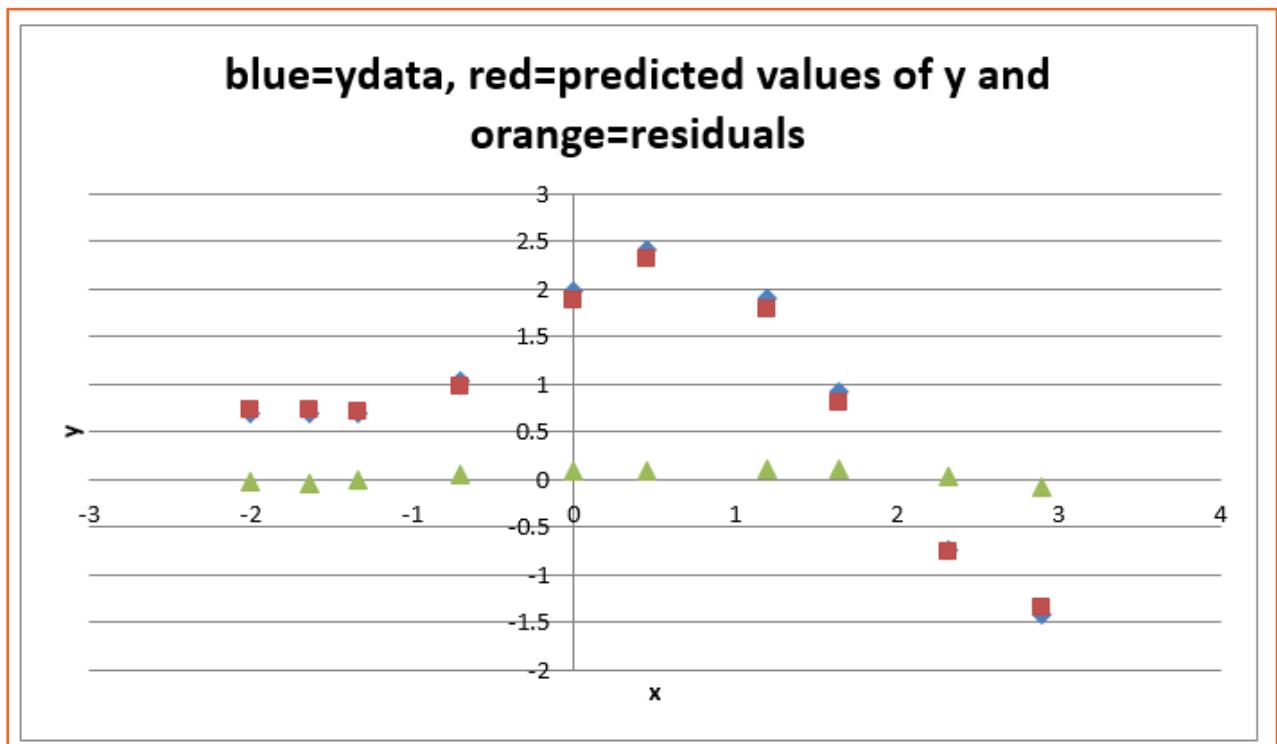
The first item in popt above (the coefficient array generated by python) was p1. To get it we write:

```
p1 = popt[0]      # p1= 1.88185099
P2= popt[1]      # p2 =0.70022986
Predicted values of y are found from:
pred=func(xdata,p1,p2)
In [48]: pred
Out[48]:
array([ 0.72705965,  0.7327565 ,  0.7056029 ,  0.98242564,
       1.88185099,
       2.31379951,  1.7968745 ,  0.81024219, -0.75969069, -
       1.34996592])
```

## Finding Residuals in Python

```
And from these the residuals are found as follows:
residuals = ydata - func(xdata,p1,p2)
residuals
Out[50]:
array([-0.02769065, -0.0322945 , -0.0102489 ,  0.05662436,
       0.09203901,
       0.09763049,  0.1140355 ,  0.10933381,  0.02871569, -
       0.07004408])
```

## Graph of Predicted y and Residuals



## Testing Using Sum of Squared Residuals

The sum of squared residuals gives one an idea of how well the model conforms to the actual data.

The sum of the squared residuals is found by:

$$\text{SSR} = \text{sum}(\text{residuals}^{**2})$$

SSR

Out[52]: 0.053812696418763392

# Heath Jarrow Morton (HJM)

## Introduction to HJM Framework

- Heath, Jarrow & Morton (1992) framework is used to model the forward interest rate
- HJM framework can be used for pricing bonds and derivatives
- Assumes that there is a connection between the drift and the volatility parameters of the forward-rate dynamics in a no-arbitrage world
- HJM models are non-Markovian
- Markovian property – future evolution of assets depend on the present state and not on their past evolution. For example, the Brownian motion has the Markov property.

## HJM

The zero-coupon bond price at t:  $Z(t;T)$  (discounting factor)

The yield curve at time t:

$$Y(t; T) = -\frac{\log Z(t; T)}{T - t}$$

The forward rate  $f(t; T, T + \delta)$  as seen at time t for the period between time  $T$  and time  $T + \delta$

$$Z(t; T + \delta) = Z(t; T) \exp(-f(t; T, T + \delta) \delta)$$

Or

$$f(t; T, T + \delta) = -\frac{\ln Z(t; T + \delta) - \ln Z(t; T)}{\delta}$$

The instantaneous forward rate:

$F(t; T)$  at time t

$$F(t; T) = \lim_{\delta \rightarrow 0} f(t; T, T + \delta)$$

So

$$F(t; T) = -\frac{\partial}{\partial T} \ln Z(t; T)$$

Links with  $(r; T)$ ,  $Y(t; T)$  and the spot rate  $r(T)$

$$\begin{aligned}
 Z(t; T) &= \exp - \left( \int_t^T F(t; s) ds \right) \\
 Y(t; T) &= \frac{1}{T-t} \int_t^T F(t; s) ds \\
 r(t) &= F(t; t) \\
 Z(t; T + \delta) &= Z(t; T) \frac{1}{(1 + \delta F(t; T, T + \delta))}
 \end{aligned}$$

Or

$$F(t; T, T + \delta) = \frac{1}{\delta} \left( \frac{Z(t; T)}{Z(t; T + \delta)} - 1 \right)$$

So

$$F(t; T) = -\frac{\partial}{\partial T} \ln Z(t; T)$$

In a real world all zero-coupon bonds evolve according to

$$dZ(t; T) = \mu(t, T) Z(t; T) dt + \sigma(t, T) Z(t; T) dW$$

Time  $t$  evolves but the maturity date  $T$  is fixed and

$$\sigma(t, t) = 0$$

Applying Ito Lemma gives:

$$dF(t; T) = \frac{\partial}{\partial T} \left( \frac{1}{2} \sigma^2(t, T) - \mu(t, T) \right) dt - \frac{\partial \sigma(t, T)}{\partial T} dW$$

This is the forward rate process in a real world.

## Spot Rate

$$Z(t;T) = r(t)Z(t;T)dt + \sigma(t,T)Z(t;T)dW$$

Here  $r(t)$  is the spot rate in a risk-neutral world.

## Forward Rate Curve

In the risk-neutral world, the forward rate curve follows:

$$dF(t;T) = m(t;T)dt + v(t,T)Z(t;T)dW$$

where

$$m(t, T) = v(t, T) \int_t^T v(t, s)ds$$

- $P(t, T)$ : price at time  $t$  of a discount bond with principal of \$1 maturing at  $T$
- $W_t$  : represents a vector of past and present values of interest rates and bond prices at time  $t$  that are relevant for determining bond price volatilities at that time
- $v(t, T, W_t)$ : volatility of  $P(t, T)$
- $f(t, T_1, T_2)$ : forward rate as seen at  $t$  for the period between  $T_1$  and  $T_2$
- $F(t, T)$ : instantaneous forward rate as seen at  $t$  for a contract maturing at  $T$
- $r(t)$ : short-term risk-free interest rate at  $t$
- $dz(t)$ : Wiener process driving term structure movements

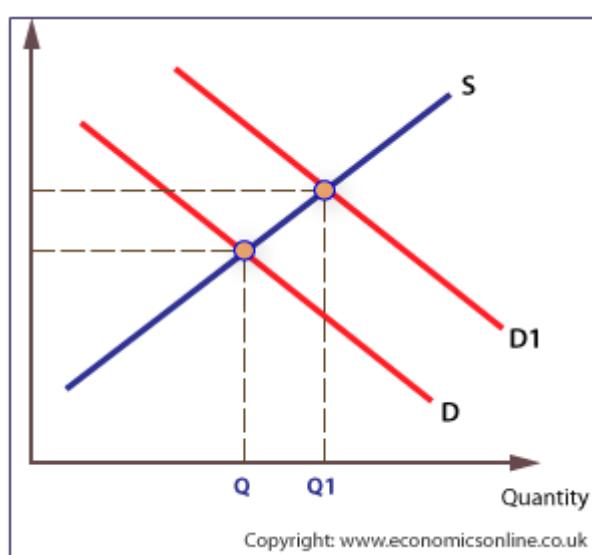
# Foreign Exchange Markets

## Foreign Exchange

The foreign exchange market is a market for converting the currency of one country into the currency of another. Exchange rate is the rate at which one currency is converted into another. The foreign exchange risk is the risk that arises for changes in exchange rates.

## Determining the Equilibrium Exchange Rate

The exchange rate is determined by supply and demand. In the graph the equilibrium exchange rate between the EUR and USD is at the intersection of the supply and demand for USD. EUR/USD – how does it cost for EUR in terms of USD dollars.



## Equilibrium Rate – Q1

There are several factors that cause the exchange rate to change:

The interest rate in U.S. increases (*caeteris paribus*) – makes U.S. bonds more attractive to European investors. They convert their EUR in USD in order to buy U.S. Bonds. The demand for U.S dollars increases as in the graph bellow from D1 to D2 and the exchange rate rises to .9 from 0.85 what we had originally as exchange rate. The dollar appreciates in value or conversely the EUR depreciates in value. The demand curve for dollars shifts up or to the right and we arrive at a new intersection point where the demand curve D2 intersects the same supply curve that we had before and we end up at the exchange rate of 0.90.

### Question:

Suppose there was an increase in income in Europe and no change in U.S. What is the effect on the exchange rate for dollars? Try to assess what is the impact in that market. If Europeans have more income they will buy more goods, buy more European goods but they also buy more imports and some of the imports will come from the U.S. Europeans would increase their purchases including purchases from the States they would increase their demand for dollars in order to affect those purchases. As above Europeans will pay 0.9 EUR to get a dollar.

Another factor that might influence the exchange rate is the relative inflation rate in the two economies. Suppose that prices were climbing in U.S. but remain the same in Europe. This would decrease European desire to buy dollars but increase American desire to buy EUR. The demand for dollars falls as in the graph below from D<sub>1</sub> from D<sub>3</sub> causing the exchange rate to change from 0.85 to 0.75. The demand curve is shifting down or to the left and we arrive at a new equilibrium point the intersection of the supply curve and demand curve.

### **Example:**

Furrukh Bashir and Adeel Luqman – “Long run Determinants of Real Exchange Rate: An Econometric Analysis from Pakistan” – Pakistan Journal of Commerce and Social Sciences 2014, Vol. 8 (2), 471-484

Furrukh and Bashir (2014) examine the macroeconomic determinants of real exchange rate and find real exchange rate elasticities with respect to various factors. Their study uses a log log model.

$$\ln RER_i = \alpha + \beta_0 \ln TR_i + \beta_1 \ln INF_i + \beta_2 \ln TOT_i + \beta_4 \ln WR_i + u_i$$

$\ln RER$  = Log of Real exchange rate

$\ln TR$  = Log of trade restrictions

$\ln INF$  = Log of Price Level

$\ln TOT$  = Log of terms of trade

InWR = Log of workers remittances

$$\text{Real Exchange Rate} = \frac{\text{Nominal Exchange Rate}}{\text{GDP Deflator}}$$

$$\text{Terms of Trade} = \frac{\text{Price index of Exportable commodity}}{\text{Price index of Importable commodity}}$$

$$\text{Trade Restriction} = \frac{\text{Nominal Gross Domestic Product}}{\text{Price of Exports} + \text{Price of Imports}}$$

Table 1. Furrukh and Bashir (2014) results for the Pakistan economy based on Vector Correction Model

Real Exchange Rate	Macroeconomic variables
Depreciates by 0.26 %	One percent rise in domestic price level
Appreciates by 0.35%	Workers' remittances are increased by one percent
Depreciates by 0.29 %	Terms of trade increase by one percent
Appreciates by 0.28 %	Trade restrictions increase by one percent