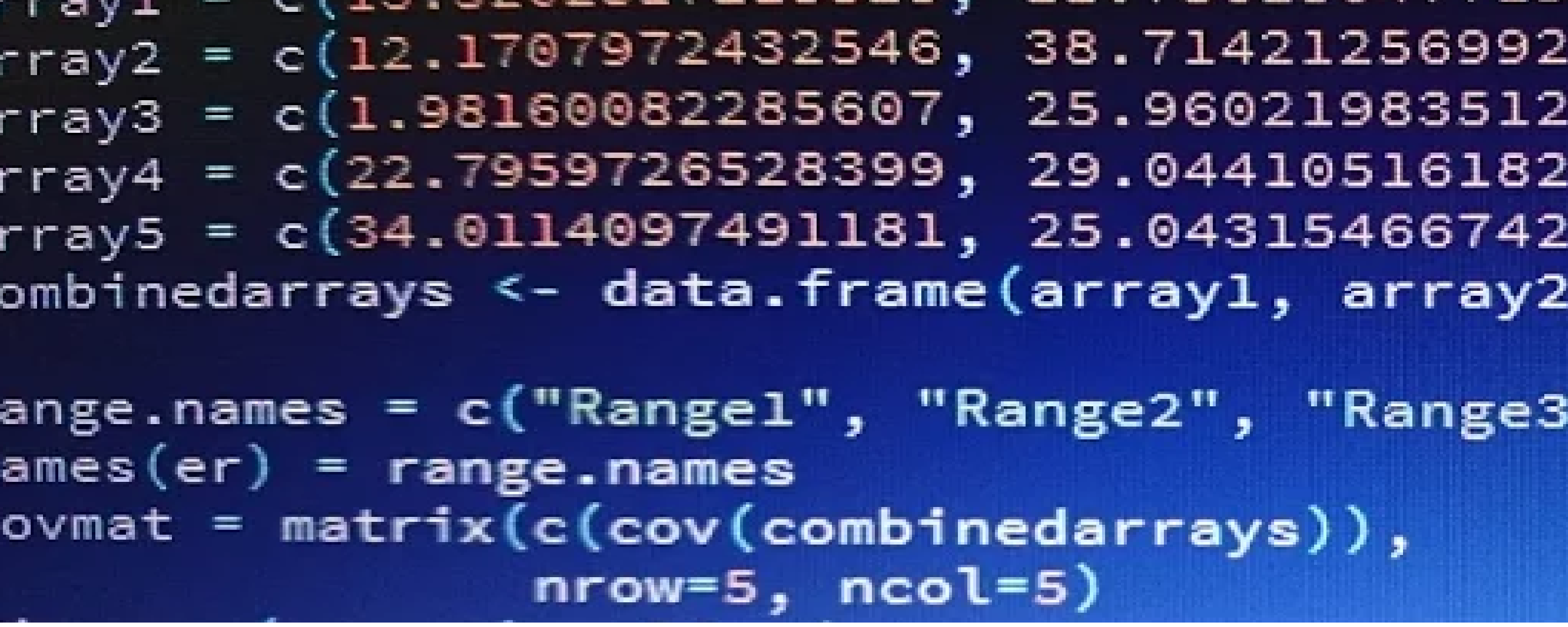


```
rray2 = c(12.1707972432546, 38.71421256992
rray3 = c(1.98160082285607, 25.96021983512
rray4 = c(22.7959726528399, 29.04410516182
rray5 = c(34.0114097491181, 25.04315466742
ombinedarrays <- data.frame(array1, array2

ange.names = c("Range1", "Range2", "Range3
ames(er) = range.names
ovmat = matrix(c(cov(combinedarrays)),
               nrow=5, ncol=5)
```

http://www.michaeljgrogan.com/

R: Regression Analysis and Data Structuring Methods

About The Author

My name is Michael Grogan. I am a data scientist with a profound passion for statistics and programming. Please visit my website where you can find my latest thoughts on data science, as well as detailed tutorials in Python, R, SQL, and other programming languages commonly used in data science. You can get in contact with me directly via e-mail, and please don't forget to connect on social media!

Contents

Introduction

Hello and thank you for subscribing to the **“R: Regression Analysis and Data Structuring Methods”** e-book! My name is Michael Grogan, and I am a data scientist and statistician.

R has long been my data science tool of choice, which I have utilised both in analysis of data for various client-based projects, as well as for my own purposes (particularly when it comes to analysing market-related data).

The purpose behind this e-book is to provide an introduction into how R can be utilised to conduct cross-sectional and time-series regression analysis, as well as how to utilise R in structuring data more effectively to facilitate analysis.

You will see that the first half of this e-book is dedicated to regression analysis as applied in financial markets. Today, we have a vast array of financial information available at our fingertips. However, the challenge arises in using that information in an effective way. For instance, it may be useful knowing that the S&P 500 is trading upwards in recent years – but what if as an investor you wish to diversify? What other market indices tend to be uncorrelated with the S&P 500? Contrary to popular belief, a simple correlation analysis will **not** tell you the whole story on whether two market indices move together or not. In the context of a time series analysis, simple correlation can be misleading.

In this context, we can see that having the right data is only half the story. More crucially, it is necessary to know **how** to best analyse that data to discern meaningful results and potential trading strategies. Therefore, the purpose of this e-book is three-fold:

1. To provide an introduction to the basic statistical principles and most commonly used programs underpinning statistical analysis on large datasets.

2. Demonstrate how an **Ordinary Least Squares** regression can be used to analyse determinants of stock market returns.

3. Illustrate how the principle of **cointegration** can be used across time series datasets to identify possible pairs trading strategies.

If the above three goals sound like a different language, don't worry! The purpose of this e-book is to lay out the above in very simple terms – and by the end of the tutorials you will have all the tools and knowledge you need to conduct this type of analysis on your own datasets. With that, let's get started!

Basic Principles of Regression Analysis

When we use regression analysis, we are doing so to predict the impact of a change in one variable on another. For instance, suppose that we have a dependent variable (Y) denoting consumption, and an independent variable (X) denoting income. Assume we have the following regression equation:

Consumption = $5000 + $1.50(Income)

This regression equation is divided into two parts:

- **Intercept:** In this equation, $5000 denotes our intercept. While the intercept can be spurious in many cases, $5000 is our minimum consumption level; i.e. assuming no increase in a person's income, this is the minimum amount a person will spend to “consume” in any given period.
- **Beta Coefficient:** Our coefficient of $1.50 refers to the change in consumption given a unit change in income. In this case, if income increases/decreases by $1 then consumption also increases/decreases by $1.50. Now, let us say that income increases by $100. What's going to happen? You guessed it – consumption will increase by $150 (100*1.5 = 150). In this case, total consumption would add up to $5000 = $150 = $5150.

Generally speaking, datasets can be one of three types:

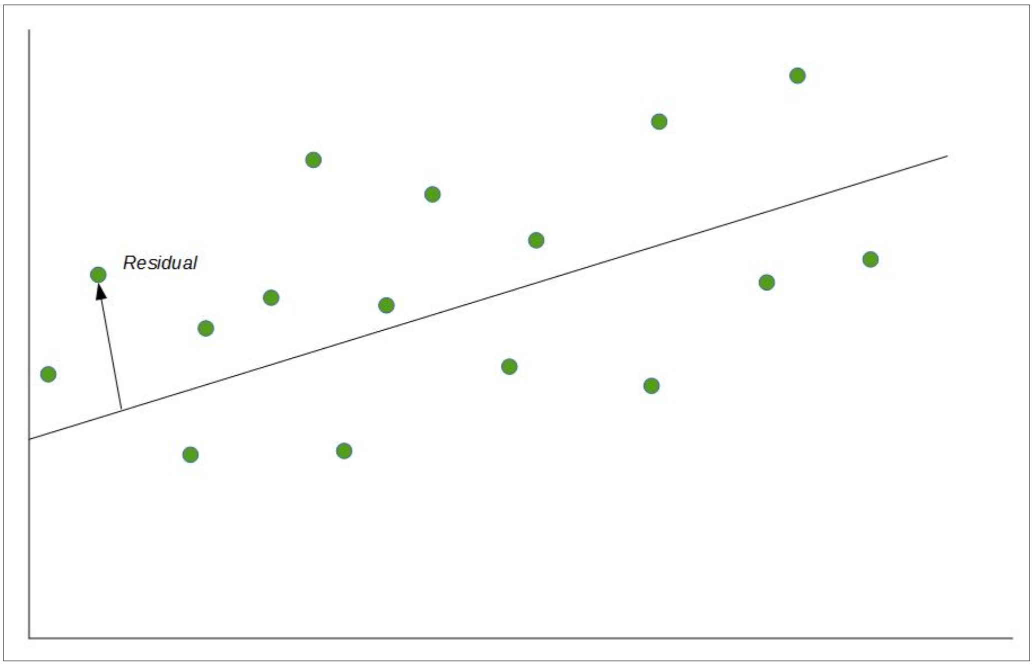
- **Cross-Sectional:** A cross-sectional type dataset is one where all data is collected at a specific point in time. For instance, surveying the heights of 100 random people over the same time period is cross-sectional, since such data cannot vary through time.

- **Time-series:** A time-series dataset is one where data is collected over time and changes with time. For instance, when we analyse the returns of a currency, stock, market index, etc, this is a time series dataset since the data is streamed over different periods.

- **Panel data:** Panel data consists of data that is both time series and cross sectional.

When conducting regression analysis, it is not only the results of our analysis that we are interested in – but also the **statistical significance** of those results. When we compute a regression equation (as above), we are actually estimating a trend line based on a range of observations.

For instance, we can see that the trend line below is formed based on an estimate of the various observations in our dataset (a scatter plot). The distance between the trend line and our observation is known as our **residual.** Our primary goal is to minimise the distance between the observed values and our trend line. Should the distances between the two become too large, then we risk **statistical insignificance** in our model, where the true values deviate greatly from our expected values, so as to render any predictive analysis from our model invalid.



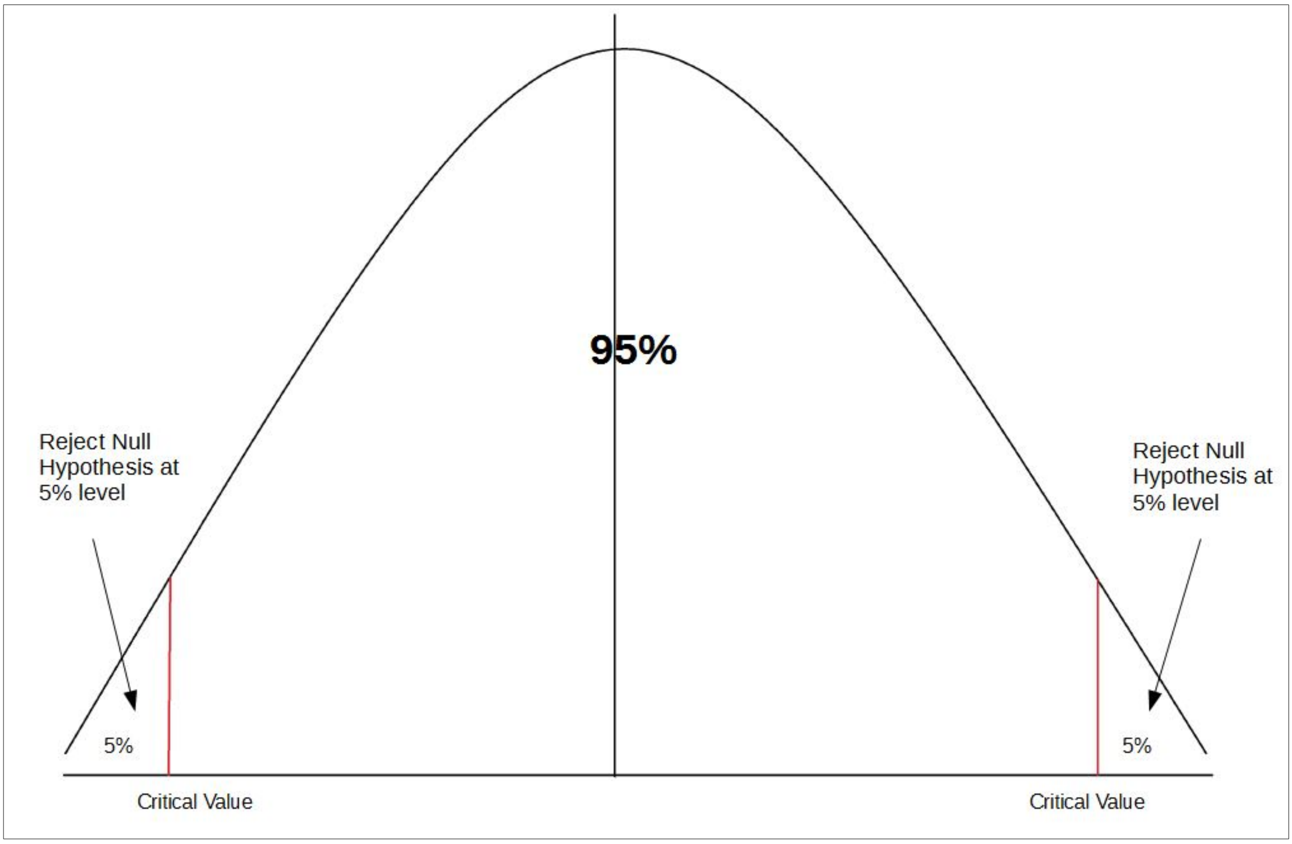
Conventionally, we measure statistical significance through the **t-statistic** – given that we are using a 5% level of significance (or a 95% confidence interval) – the p (probability) value is what allows us to determine significance. For instance, if our p-value for a particular coefficient lies below 0.05, then we assume that the coefficient is significant at the 5% level. The most common form of significance test is based on a two-tailed test. For instance:

Null Hypothesis (the result we do not expect): ***Income = 0***

Alternative Hypothesis (the result we do expect): ***Income ≠ 0***

If our p-value comes back as significant (under 0.05 at the 5% level), then we reject the null hypothesis and conclude that the **income** variable is significantly different from zero.

Normal Distribution Curve


95%
Reject Null Hypothesis at 5% level
Reject Null Hypothesis at 5% level
5%
5%
Critical Value
Critical Value

As previously explained, statistical computing packages will typically calculate a p-value from which we can decipher whether our coefficient is significant at a certain threshold. However, in the instance where we wish to calculate the t-statistic manually, we are comparing the same to the critical value to determine whether or not to reject the null hypothesis. For example:

t-value = Estimate/Standard Error

- Assuming estimate of **681.402** along with a standard error of **103.270**, this yields a t-value of **6.598** (681.402/103.270 = 6.598).
- At the **5%** significance level our z-value is **1.96** at the 95% confidence level (which is the critical value for a normal distribution at this level).
- Given that 6.598 > 1.96, we would reject the null hypothesis at the 5% significance level and conclude that the alternative hypothesis is significantly different from zero.

Ordinary Least Squares – An Analysis of Stock Returns

In the website tutorial given on Ordinary Least Squares (which I recommend reviewing before reading further), we ran an analysis on a hypothetical dataset of **49** stocks. The purpose of the regression was to determine the impact of dividends, earnings and debt to equity on stock returns. We are using a cross-sectional dataset in this instance – meaning that all data is collected at a specific point in time.

Given that we have laid out the essentials behind running and interpreting a regression on this dataset, the primary aim here is to illustrate the commands run in the **R Programming Language** to read data from the appropriate csv file, and perform the various calculations.

1. To download the Rstudio IDE package, please refer to http://www.rstudio.com/.

2. Save the ols_stock csv file to the relevant directory; e.g. C:\\Users\\Your Computer Name\\Documents\R\\ols_stock.csv.

3. Open the RStudio interface and click File -> New File -> R Script.

4. Set the relevant directory by entering the command:

```
setwd ("C:\\Users\\Your Computer Name\\Documents\R")
```

5. Read the relevant data:

```
mydata<- read.csv("C:\\Users\\Your Computer Name\\Documents\R\\ols_stock.csv")
attach(mydata)
```

6. Run the OLS regression:

```
reg <- lm(stock_return_scaled ~ dividend + earnings_ranking + debt_to_equity)
summary(reg)
```

7. Plot the data:

```
plot (stock_return_scaled ~ dividend + earnings_ranking + debt_to_equity)
reg1 <- lm(stock_return_scaled ~ dividend + earnings_ranking + debt_to_equity)
```

8. Redefine the variables:

```
Y <- cbind(stock_return_scaled)
X <- cbind(dividend, earnings_ranking, debt_to_equity)
```

9. Install the “car” library:

library(car)

10. Conduct tests for multicollinearity by regressing the independent variables against each other, along with conducting a variance inflation factor (VIF) test:

reg1m <- lm(dividend ~ earnings_ranking + debt_to_equity)
summary(reg1m)
reg2m <- lm(earnings_ranking ~ dividend + debt_to_equity)
summary(reg2m)
reg3m <- lm(debt_to_equity ~ earnings_ranking + dividend)
summary(reg3m)

vif(reg1m)
vif(reg2m)
vif(reg3m)

11. Conduct a Breusch-Pagan test for heteroscedasticity:

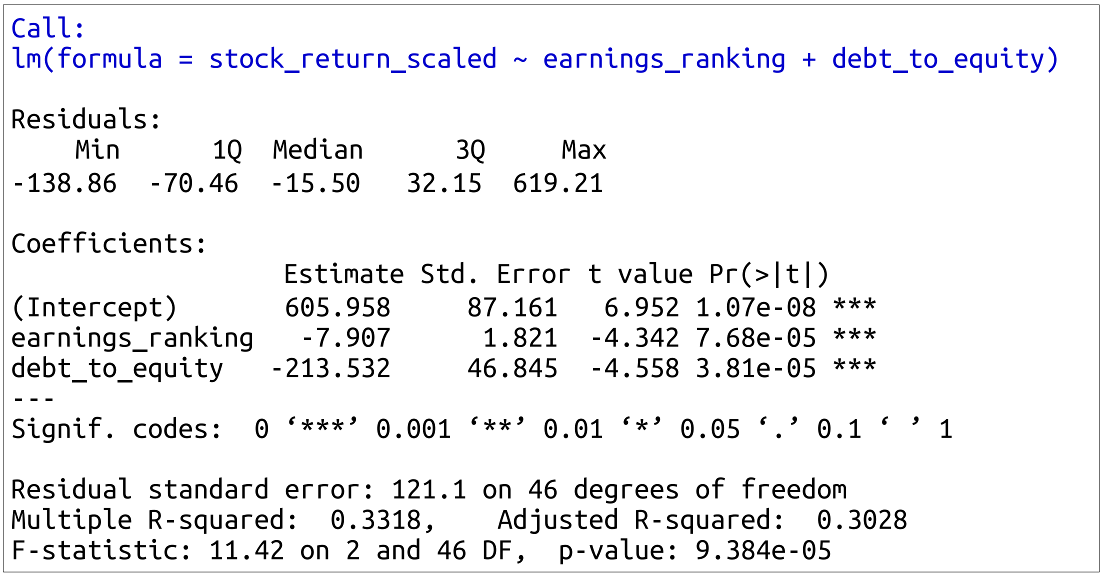
library(lmtest)
bptest(Y ~ X)

Regression Results

When we run the model initially, we see that the “dividend” variable comes back as insignificant:

```
Call:
lm(formula = stock_return_scaled ~ earnings_ranking + dividend +
    debt_to_equity)

Residuals:
    Min      1Q  Median      3Q     Max
-142.56  -53.63  -15.25   22.21  603.66

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        681.402    103.270   6.598 4.03e-08 ***
earnings_ranking   -10.147      2.465  -4.116 0.000162 ***
dividend          -102.704     76.943  -1.335 0.188653
debt_to_equity    -182.134     52.068  -3.498 0.001068 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 120.1 on 45 degrees of freedom
Multiple R-squared:  0.3573,	Adjusted R-squared:  0.3144
F-statistic: 8.338 on 3 and 45 DF,  p-value: 0.0001616
```

If we drop this variable and run the regression again, we obtain:



```
Call:
lm(formula = stock_return_scaled ~ earnings_ranking + debt_to_equity)

Residuals:
    Min      1Q  Median      3Q     Max 
-138.86  -70.46  -15.50   32.15  619.21 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)       605.958     87.161   6.952 1.07e-08 ***
earnings_ranking   -7.907      1.821  -4.342 7.68e-05 ***
debt_to_equity   -213.532     46.845  -4.558 3.81e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 121.1 on 46 degrees of freedom
Multiple R-squared:  0.3318,	Adjusted R-squared:  0.3028 
F-statistic: 11.42 on 2 and 46 DF,  p-value: 9.384e-05
```

From the above, our regression equation is as follows:
stock_return_scaled = 605.958 – 7.907 (earnings_ranking) – 213.532 (debt_to_equity)

With our stock return (dependent variable) being measured in basis points, our regression equation can be interpreted as follows:

- A higher earnings ranking (or lower earnings rate in ranking terms) corresponds to a **-10.147** drop in a stock's return in basis points.
- An increase of the debt/equity ratio by 1 corresponds to a -182.134 drop in a stock's return in basis points. On an incremental basis (-182.134/100), an increase of 0.01 in the debt/equity ratio corresponds to a **-1.82** drop in basis points in overall stock returns.

Note that our Breusch-Pagan test for heteroscedasticity is insignificant at the 5% level. However, you will also notice that our dependent variable has *scaled* the stock returns. This means that we are calculating the returns assuming a constant market capitalisation of $100 million. This is important as the returns of various companies will vary depending on size.

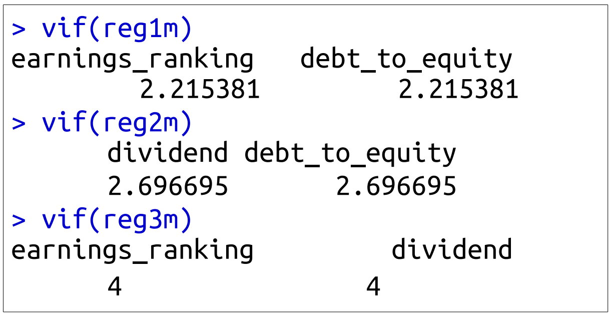
For instance, on a particular stock market index – the large-cap stocks will typically skew the overall return to a greater extent than the small-caps. In other words, we have an uneven variance across our samples, and this will result in unreliable hypothesis testing. However, from scaling our returns in this manner the Breusch-Pagan test for heteroscedasticity came back as statistically insignificant at the 5% level, and therefore we cannot reject our null hypothesis of homoscedasticity.

Variance Inflation Factor Test For Multicollinearity

Additionally, it is also necessary to test for multicollinearity using the Variance Inflation Factor (VIF) test. Multicollinearity arises when two independent variables are significantly correlated with each other, which biases our OLS estimates and leads to unreliable hypothesis testing.

We test for this using the Variance Inflation Factor test, where a reading of **5** or higher is considered to be a warning sign for multicollinearity.

From the below, we see that our **vif** function is below **5** across our independent variables. Therefore, our VIF test suggests that our independent variables are free of multicollinearity.



```
> vif(reg1m)
earnings_ranking   debt_to_equity
        2.215381         2.215381
> vif(reg2m)
      dividend debt_to_equity
      2.696695       2.696695
> vif(reg3m)
earnings_ranking         dividend
      4                4
```

Cointegration and Pairs Trading

Imagine a drunk woman walking home with her dog one night. Now, imagine two scenarios:

1) the woman is holding the dog with a long leash

2) the dog does not have any leash at all

In the first scenario, the woman and the dog may at times walk far apart from each other, but will eventually always rejoin each other since the leash is keeping both of them together. In this regard, the woman and the dog are a **cointegrated series –** no matter how far apart they move from each other, the leash will bring them back together.

However, the second scenario is what is described as a **random walk –** the drunk woman and the dog will wander off far away from each other and may never rejoin – there is nothing linking the two of them together. Since we have a random walk, this series is **not** cointegrated.

Cointegration is an important element of **time series analysis –** where our observations vary through time. Ordinary Least Squares is usually insufficient in analysing relationships across time series data, since the data varies through time. For instance, let us suppose that we were to run **1)** a cross-sectional regression analysis on height across all 30-year old males versus **2)** a time-series regression analysis where we measure the height of a select group of males year-after-year until they turn 18.

For the first regression, we will obtain useful information since there is likely wide variation across the height of 30-year old males and we can investigate further the characteristics associated with height. However, in the case of a time series, our results will not give us any useful information. Since each person in the sample is going to continue growing until they turn 18, we will have highly significant t-statistics and a very high R-Squared, along with highly positive slope coefficients. This does not tell us anything about the variation in heights, since differences in height may not become apparent until the latter years.

Similarly, when attempting to analyse financial data in this manner – standard analysis will be prone to a high degree of error. For instance, if we were to analyse the correlation coefficient of the S&P 500 and FTSE 100 stock indices from 1950-present, we would likely find high correlation between them since the two indices have appreciated over time - consistent with most other developed market indices. However, this on its own would not account for the short-term periods where there may have been significant deviation between the two. Moreover, we have no concrete indicators to assume that the two indices will continue to move in tandem over the next fifty years.

We can solve this issue using cointegration analysis – where we examine if a relationship between two time series variables is significant or simply due to chance.

Here, we will examine the cointegration patterns between two time series – denoted **Series 1** and **Series 2**. Please refer to the *cointegration_sample.csv* spreadsheet for further details.

As discussed, we use cointegration analysis to determine if any correlation between the two time series is significant or simply due to random chance. To do this, we use what is called the **two-step Engle-Granger** cointegration test. This method assumes that if Series 1 and Series 2 are cointegrated, then a linear combination of the two must be **stationary.** When we say that a time series is stationary, it means that its mean, variance and autocorrelation are consistent over time, making predictions from the model more reliable. When running the Dickey-Fuller test – a p-value of 0.05 or below means that our test is significant at the 5% level, and supports the presence of stationarity in our model.

Additionally, when we refer to a "linear combination" of the two time series, we are referring to a first-differenced version of the two time series – which denotes the incremental change in the variable from one period to the next. When regressing the first differences of the two time series **A** and **B**, we expect that the Dickey-Fuller test will be significant at the 5% level.

1. To download the Rstudio IDE package, please refer to http://www.rstudio.com/.

2. Save the ols_stock csv file to the relevant directory; e.g. C:\\Users\\Your Computer Name\\Documents\R\\cointegration_sample.csv.

3. Open the RStudio interface and click File -> New File -> R Script.

4. Set the relevant directory by entering the command:

```
setwd ("C:\\Users\\Your Computer Name\\Documents\R")
```

5. Read the relevant data:

```
mydata<- read.csv("C:\\Users\\Your Computer Name\\Documents\R\\cointegration_sample.csv")
attach(mydata)
```

6. Install "tseries" library for time series data:

```
library(tseries)
```

7. Run the Augmented Dickey-Fuller Test on both series 1 and 2 in ln (natural logarithmic) format:

```
adf.test(lnseries1)
adf.test(lnseries2)
adf.test(lnseries1_firstdifference)
adf.test(lnseries2_firstdifference)
```

8. Run an OLS regression on the first differences of the two series (the results of an OLS regression can only be valid in this instance if both time series are stationary):

olsreg <- lm(lnseries1_firstdifference ~ lnseries2_firstdifference)
summary(olsreg)

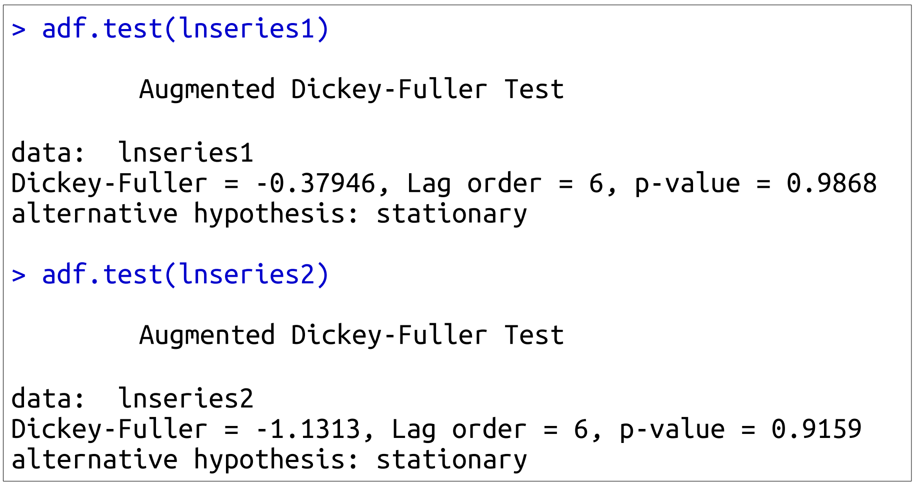
9. Run a Granger Causality analysis – i.e. an OLS regression on Series 1 and the lagged value of Series 2 (time t-1):

olsreg <- lm(lnseries1_timet ~ lnseries2_timetminusone)
summary(olsreg)

10. Plot the data:

plot (lnseries1)
plot (lnseries1_firstdifference)
plot (lnseries2)
plot (lnseries2_firstdifference)

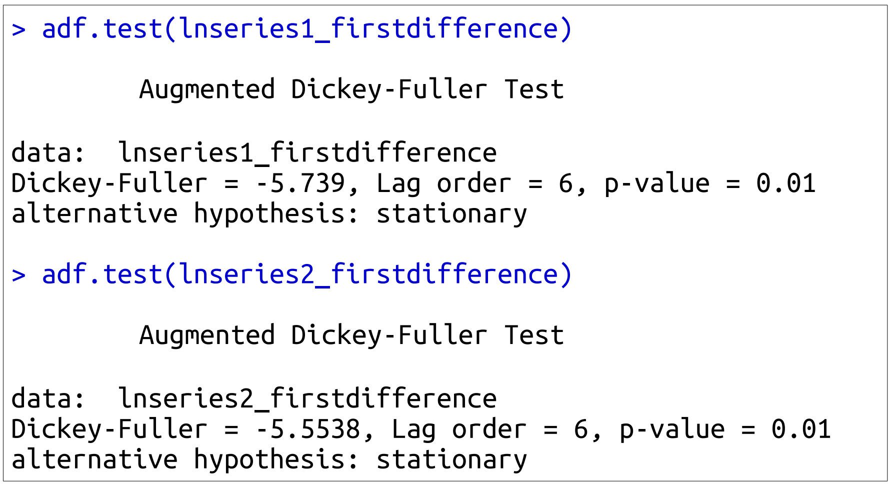
When we look at the results of the above analysis, we yield the following observations:



```
> adf.test(lnseries1)

        Augmented Dickey-Fuller Test

data:  lnseries1
Dickey-Fuller = -0.37946, Lag order = 6, p-value = 0.9868
alternative hypothesis: stationary

> adf.test(lnseries2)

        Augmented Dickey-Fuller Test

data:  lnseries2
Dickey-Fuller = -1.1313, Lag order = 6, p-value = 0.9159
alternative hypothesis: stationary
```

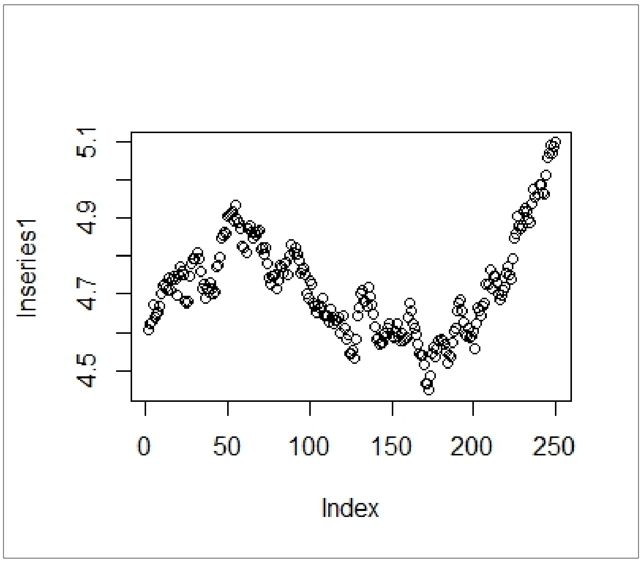
With p-values well above 0.05, both time series at time t are non-stationary; i.e. they do not have a constant mean, variance and autocorrelation.

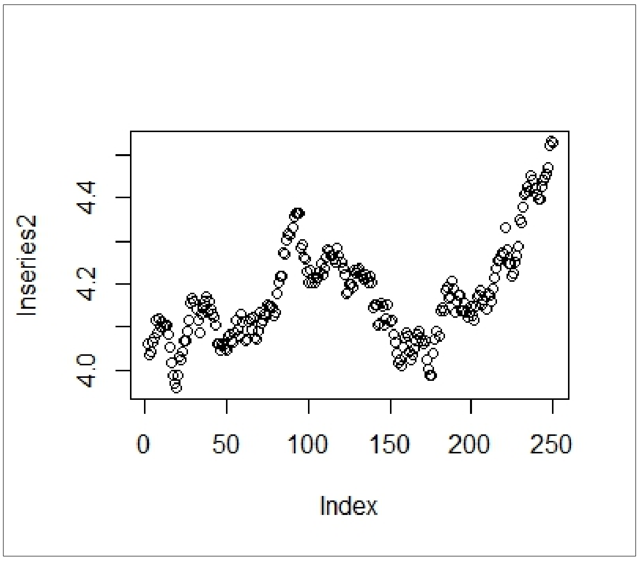
However, when we run the Dickey-Fuller test on the data that has been first-differenced, we find that our p-values are highly significant at the 5% level:



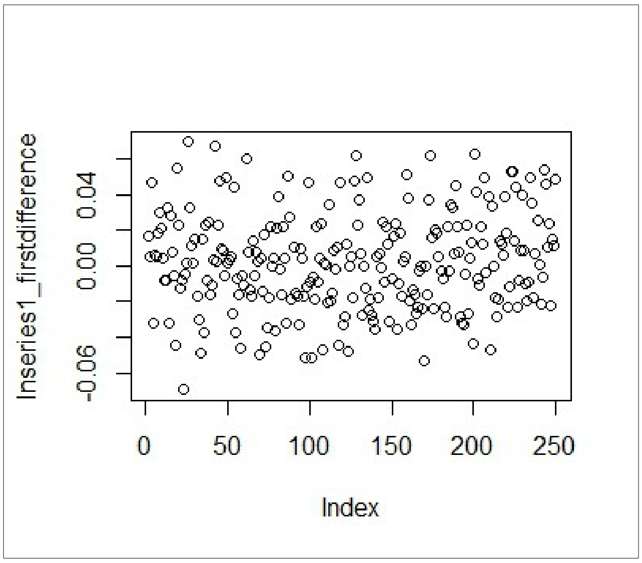
```
> adf.test(lnseries1_firstdifference)

        Augmented Dickey-Fuller Test

data:  lnseries1_firstdifference
Dickey-Fuller = -5.739, Lag order = 6, p-value = 0.01
alternative hypothesis: stationary

> adf.test(lnseries2_firstdifference)

        Augmented Dickey-Fuller Test

data:  lnseries2_firstdifference
Dickey-Fuller = -5.5538, Lag order = 6, p-value = 0.01
alternative hypothesis: stationary
```

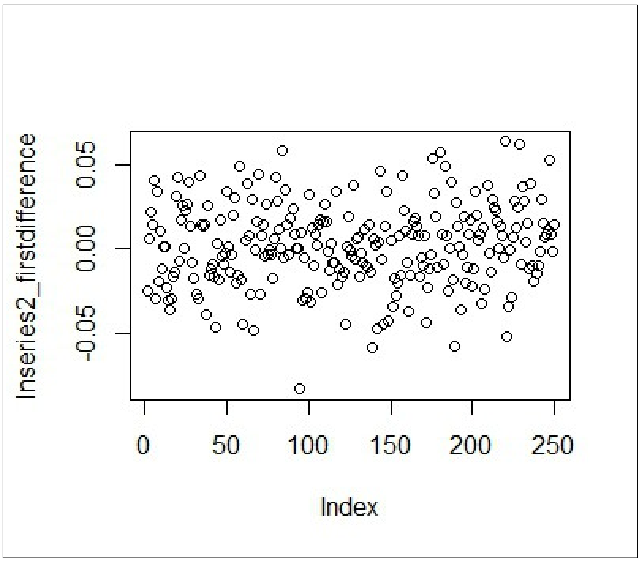
In terms of graphing the plots, we see that the data at time t shows a clear trend:





However, when we plot our residuals across first-differences, we see that our residuals have no clear pattern and are scattered randomly:





It is notable that for the first **50** trading days, series 1 appreciated while series 2 remained stagnant. Given that we know cointegration exists between the two time series, a possible pairs trading strategy could have been exploited by taking a **short** position on series 1 and a **long position** on series 2. From day 50 to 100, we see that series 1 indeed depreciates while series 2 appreciates. Eventually, the two series rise together over the long-term, owing to convergence.

Additionally, when we run an OLS regression on data at time t and time t-1 (Granger Causality), we find that while we have an insignificant independent variable for time t; we have a highly significant independent variable for series 2 at time t-1. This suggests that the time series 1 and 2 may show Granger causality; i.e. a change in the value of series 2 at time t has a direct effect on the value of series 1 at time (t+1).

OLS Regression

```
Call:
lm(formula = lnseries1_firstdifference ~ lnseries2_firstdifference)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.074223 -0.018406 -0.000841  0.016694  0.067024 

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)               0.001970   0.001734   1.136    0.257
lnseries2_firstdifference 0.104681   0.071190   1.470    0.143

Residual standard error: 0.02733 on 248 degrees of freedom
Multiple R-squared:  0.008643,  Adjusted R-squared:  0.004646 
F-statistic: 2.162 on 1 and 248 DF,  p-value: 0.1427
```

OLS Regression – Granger Causality

```
Call:
lm(formula = lnseries1_timet ~ lnseries2_timetminusone)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.21157 -0.08398 -0.01628  0.08336  0.27868 

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)    
(Intercept)              2.25097    0.26500   8.494 1.86e-15 ***
lnseries2_timetminusone  0.59187    0.06347   9.326  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1143 on 248 degrees of freedom
Multiple R-squared:  0.2596,    Adjusted R-squared:  0.2566 
F-statistic: 86.97 on 1 and 248 DF,  p-value: < 2.2e-16
```

While the above example is hypothetical, there are a myriad of pairs trading strategies that could be identified using this method. Should two time series be cointegrated over time, then it is statistically likely that they will converge at a certain point, which allows for potential opportunities during times of significant deviation between the two series.

Data Cleaning and Structuring in R

One of the big issues when it comes to working with data in any context is the issue of data cleaning and merging of datasets, since it is often the case that you will find yourself having to collate data across multiple files, and will need to rely on R to carry out functions that you would normally carry out using commands like VLOOKUP in Excel.

Data Cleaning and Merging Functions

For our example, we have two datasets:

1. sales.csv: This file contains the variables Date, ID (which is Product ID), and Sales. We load this into R under the name mydata.

2. customers.csv: This file contains the variables ID, Age, and Country. We load this into R under the name mydata2.

The following are examples of popular techniques employed in R to clean a dataset, along with how to format variables effectively to facilitate analysis. The below functions work particularly well with panel datasets, where we have a mixture of cross-sectional and time series data:

1. Storing variables in a data frame:

To start off with a simple example, let us choose the customers dataset. Suppose that we only wish to include the variables ID and Age in our data. To do this, we define our data frame as follows:

```
dataframe<-data.frame(ID,Age)
```

2. Mimic VLOOKUP by using the merge functions:

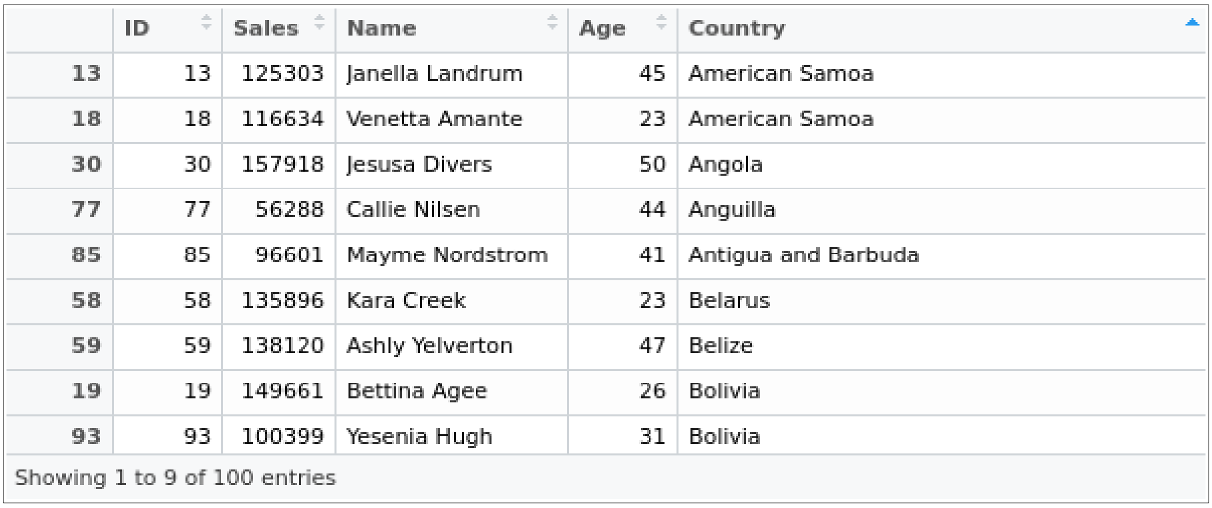
Often times, it is necessary to combine two variables from different datasets similar to how VLOOKUP is used in Excel to join two variables based on certain criteria. In R, this can be done using the merge function.

For instance, suppose that we wish to link the Date variable in the sales dataset with the Age and Country variables in the customers dataset – with the ID variable being the common link.

Therefore, we do as follows:

```
mergeinfo<-merge(mydata[, c("ID", "Sales")],mydata2[, c("ID", "Age", "Country")])
```

Upon doing this, we see that a new dataset is formed in R joining our chosen variables:



	ID	Sales	Name	Age	Country
13	13	125303	Janella Landrum	45	American Samoa
18	18	116634	Venetta Amante	23	American Samoa
30	30	157918	Jesusa Divers	50	Angola
77	77	56288	Callie Nilsen	44	Anguilla
85	85	96601	Mayme Nordstrom	41	Antigua and Barbuda
58	58	135896	Kara Creek	23	Belarus
59	59	138120	Ashly Yelverton	47	Belize
19	19	149661	Bettina Agee	26	Bolivia
93	93	100399	Yesenia Hugh	31	Bolivia

Showing 1 to 9 of 100 entries

3. Using as.date to format dates and calculate duration

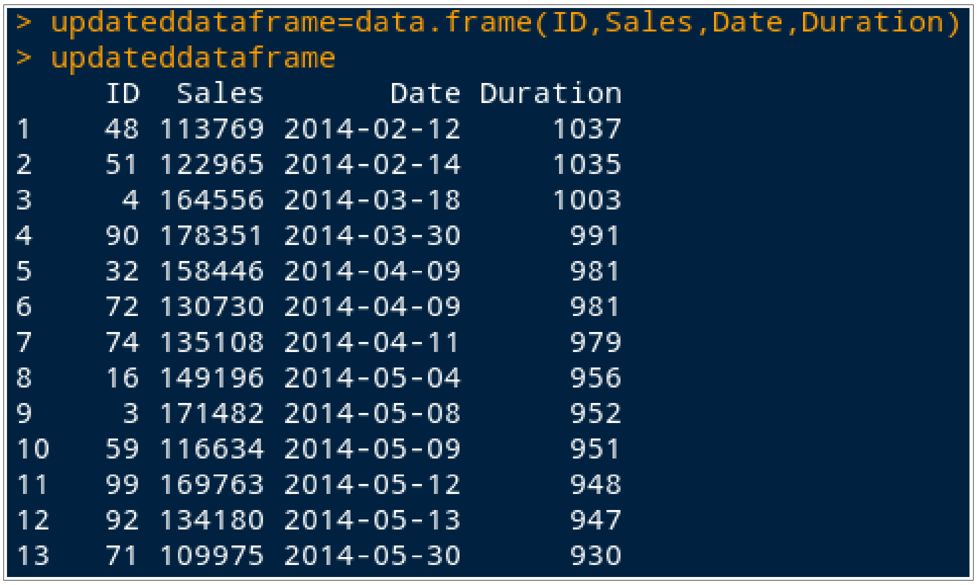
Suppose that we now wish to calculate the number of days between the current date and the date of sale as listed in the sales file. In order to accomplish this, we can use as.date as follows:

```
currentdate=as.Date('2016-12-15')
dateinfile=as.Date(Date)
Duration=currentdate-dateinfile
```

Going back to the example above, suppose that we now wish to combine this duration variable with the rest of our data.

Hence, we can now combine our new **Duration** variable with the merge function as above, and can do this as follows:

```
durationasdouble=as.double.difftime(Duration, units='days')
updateddataframe=data.frame(ID,Sales,Date,durationasdouble)
updateddataframe
```



```
> updateddataframe=data.frame(ID,Sales,Date,Duration)
> updateddataframe
    ID  Sales       Date Duration
1   48 113769 2014-02-12     1037
2   51 122965 2014-02-14     1035
3    4 164556 2014-03-18     1003
4   90 178351 2014-03-30      991
5   32 158446 2014-04-09      981
6   72 130730 2014-04-09      981
7   74 135108 2014-04-11      979
8   16 149196 2014-05-04      956
9    3 171482 2014-05-08      952
10  59 116634 2014-05-09      951
11  99 169763 2014-05-12      948
12  92 134180 2014-05-13      947
13  71 109975 2014-05-30      930
```

4. grepl: Remove instances of a string from a variables

Let us look to the Country variable. Suppose that we wish to remove all instances of “Greenland” from our variable. This is accomplished using the **grepl** command:

```
countryremoved<-mydata2[!grepl("Greenland", mydata2$Country),]
```

5. Delete observations using head and tail functions

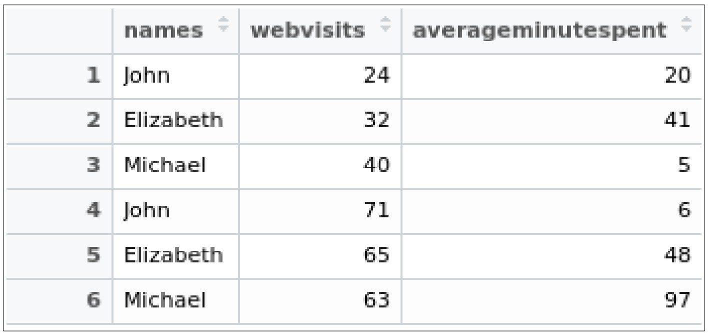
The head and tail functions can be used if we wish to delete certain observations from a variable, e.g. Sales. The head function allows us to delete the first 30 rows, while the tail function allows us to delete the last 30 rows.

When it comes to using a variable edited in this way for calculation purposes, e.g. a regression, the as.matrix function is also used to convert the variable into matrix format:

```
Salesminus30days←head(Sales,-30)
X1=as.matrix(Salesminus30days)
X1

Salesplus30days<-tail(Sales,-30)
X2=as.matrix(Salesplus30days)
X2
```

6. Replicate SUMIF using the "aggregate" function

Let us suppose that we have created the following table as below (created in R, not from dataset), and want to obtain the sum of web visits and average minutes spent on a website in any particular period:

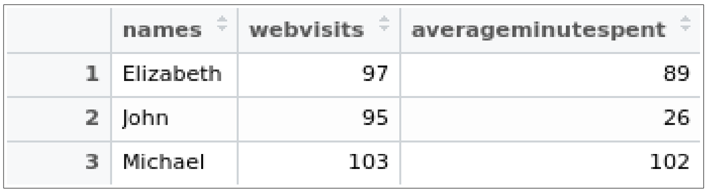


	names	webvisits	averageminutespent
1	John	24	20
2	Elizabeth	32	41
3	Michael	40	5
4	John	71	6
5	Elizabeth	65	48
6	Michael	63	97

In this instance, we can replicate the **SUMIF** function in Excel (where the values associated with a specific identifier are summed up) by using the **aggregate** function in R. This can be done as follows (where **raw_table** is the table specified as above):

```
sumif_table<-aggregate(. ~ names, data=raw_table, sum)
sumif_table
```

Thus, the values associated with identifiers (in this case, names) are summed up as follows:



	names	webvisits	averageminutespent
1	Elizabeth	97	89
2	John	95	26
3	Michael	103	102

Other Useful R Links

- ARIMA and Holt-Winters: Stock Price Forecasting
- Binomial Logistic Regression – An Analysis of Stock Dividends
- Cumulative Binomial Probability Plot
- Is A Low R-Squared Statistic A Bad Thing?
- K-Means Clustering: An Example of Stock Returns and Dividend Yields
- Kruskal-Wallis Test: Nonparametric ANOVA
- Ordinary Least Squares – An Analysis of Stock Returns
- Quantmod: How To Analyse Stock Data Using R
- Stationarity and Cointegration
- Testing For Seasonality With ANOVA