A Time Series Analysis with The Julia Language by Dustin Ehling

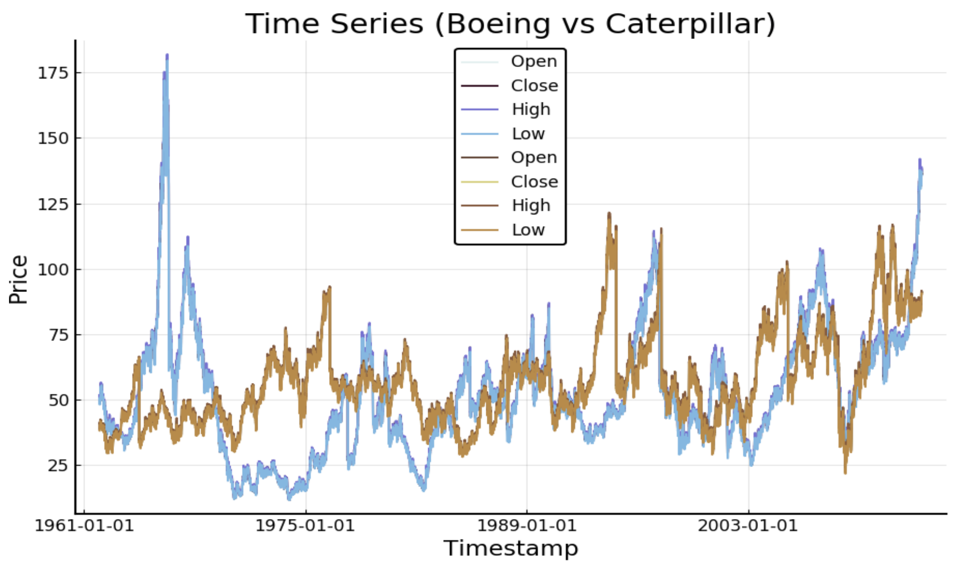
Abstract

Julia is a new high-level general-purpose language designed for numerical analysis that seeks to serve as an alternative to popular languages such as Python. Unique aspects of Julia's design include multiple dispatch integration and parametric polymorphism which allow a language to a language more expressive, while still maintaining full static type safety. Julia 1.0 was released in late 2018 and has undergone major optimization improvements and package development since. For the purpose of this report, we will look at the current state of some Time Series analysis techniques that currently exist in the Julia ecosystem.

It should be noted that Julia offers a robust R porting, allowing Julia users to call any R function with the *Rcall* Package, however this report will look at methods native to the current 1.1 version.

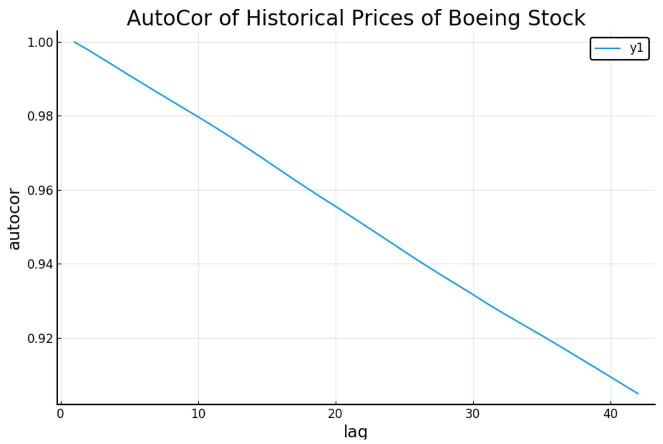
Introduction

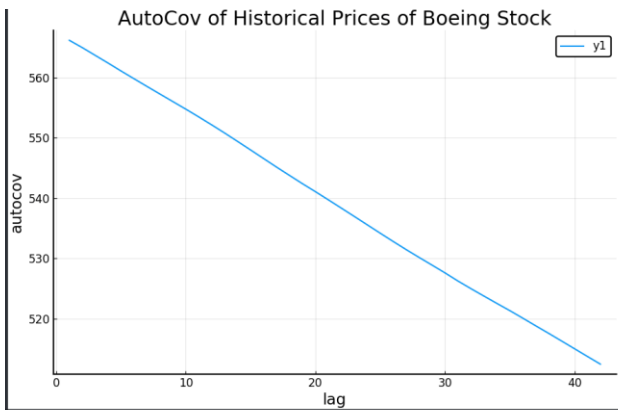
A time series is a collection of quantitative observations that are evenly spaced in time and measured successively. Examples of time series include the continuous monitoring of a person's heart rate, hourly readings of air temperature, daily closing price of a company stock, monthly rainfall data, and yearly sales figures. Time series analysis is generally used when there are at least 100 data points in a series, but often are utilized on massive data-sets that may otherwise be too large to analyze. Observations made over time can be either discrete or continuous. Both types of observations can be equally spaced, unequally spaced, or have missing data. Discrete measurements can be recorded at any time interval but are most often taken at evenly spaced intervals. In this report I used discrete historical stock data from Caterpillar and Boeing, from 1962 to 2012, containing daily stock open/close and high/low prices. The figure below plots the two series we will use, where Boeing is the blue series, and Caterpillar is the brown series.



Analysis and Interpretation

1. Auto-correlation and Auto-covariance





For our example, I looked at the daily closing price of Boeing stock from 1962 to 2012 and calculated the autocovariance/autocorrelation values at lag levels 1 to 42. The overall trend lines of these analysis are plotted above and show that as the lag value is increasing there is a decrease in correlation and co-variance. However, the stock data for Boeing seems to be highly correlated no matter the lag value, as it never falls below a 0.90 correlation value. These values can be calculated by hand with the following equation:

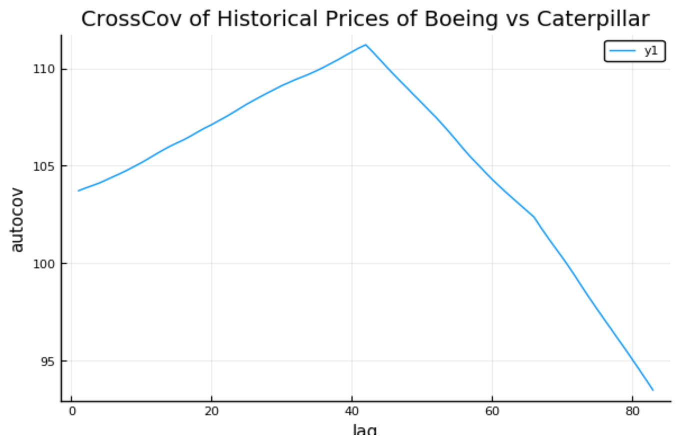
The autocovariance function, γ(t,s) is defined as:

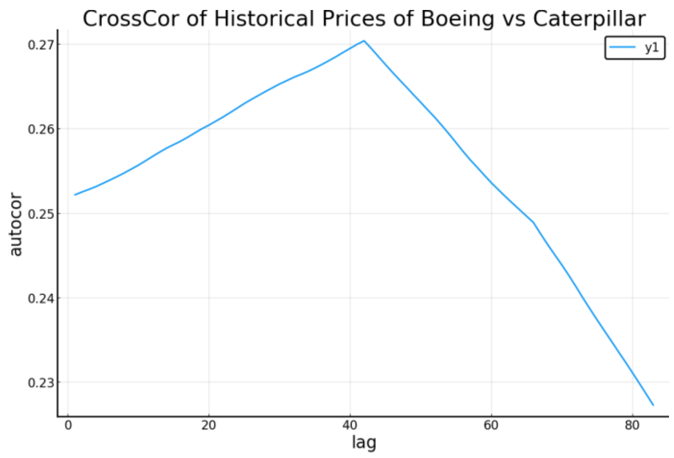
$$\mathrm{Cov}(Yt, Ys) = E[(Yt - \mu t)(Ys - \mu s)] = E(YtYs) - \mu t\ \mu s \qquad \text{for } t, s = 0, 1\pm, \pm 2 \ldots$$

The autocorrelation function, ρt,s, is given by:

$$\rho(t,s) = \mathrm{Corr}(Yt, Ys) = \mathrm{Cov}(Yt, Ys) / \mathrm{SQRT}(\mathrm{Var}(Yt)\mathrm{Var}(Ys)) \qquad \text{for } t\ s, = 0, 1\pm, \pm 2 \ldots$$

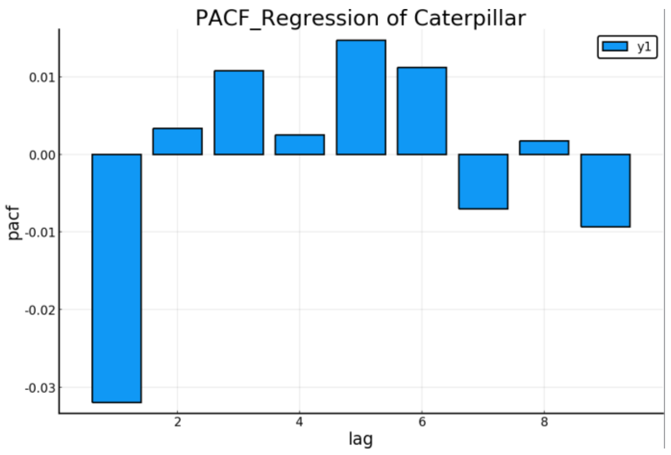
2. Cross-correlation and Cross-covariance

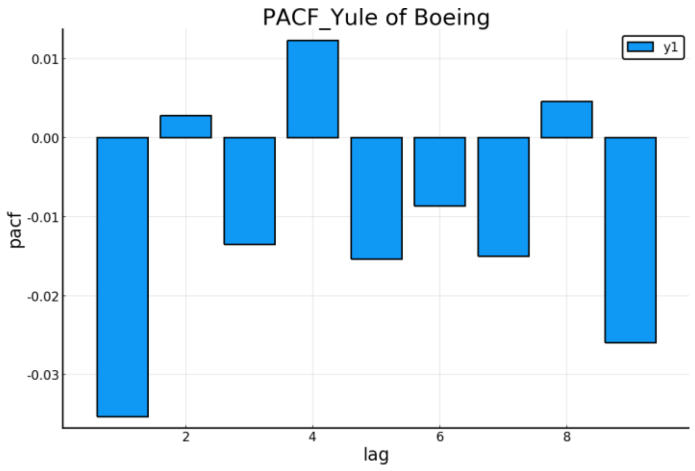




Given two stochastic processes or series, the cross-covariance is a function that gives the covariance of one of the processes with the other process at the paired set of time points. This is a measure of similarity of the two series, which is commonly used to find features in an unknown signal by comparing it to a known one, like the theory behind the Chi-Square test of distributions. From the figures above we can see that, at lag=42, there is a maximum critical value followed by a sharp decline. If the daily closing price of our two stocks were delayed copies (dependent), there would exist a peak of 1.0 on our plot. However, our plotting does not contain this value and demonstrates a weak correlation between the two stocks closing prices.

3. Partial Auto Correlation Functions





For a given stochastic process one is often interested in the connection between two random variables of a process at different points in time. One way to measure a linear relationship is with the PACF, which is the correlation between these two variables. This is the correlation between two variables under the assumption that we know and take into account the values of some other set of variables. I have included both the regression and yule-walker based PACF in my full code, and have included an plotting of each above.

In a regression context, you have a dependent variable(y) and predictor variables (x1,x2,…). The partial correlations between our dependent (pacf) and predictor(lag), is calculated by correlating the residuals of our variables. For a time series the PACF is the conditional correlation between the value at present time t and the value at time minus the lag level. For our plotting the pacf value at lag 1 is approx. equal to 1 and are excluded as to not skew the rest of the plot.

Yule-walker equations are defined by:

$$\gamma k = \varphi 1\gamma(k-1) + \varphi 2\gamma(k-2) \quad \text{for } k = 1, 2, 3, ...$$

or (divide above by y0)

$$\rho k = \varphi 1\rho(k-1) + \varphi 2\rho(k-2) \quad \text{for } k = 1, 2, 3, \dots$$

Successive values of ρk may be easily calculated from this recursive relationship. These ρk value are our pacf values at lag level k.

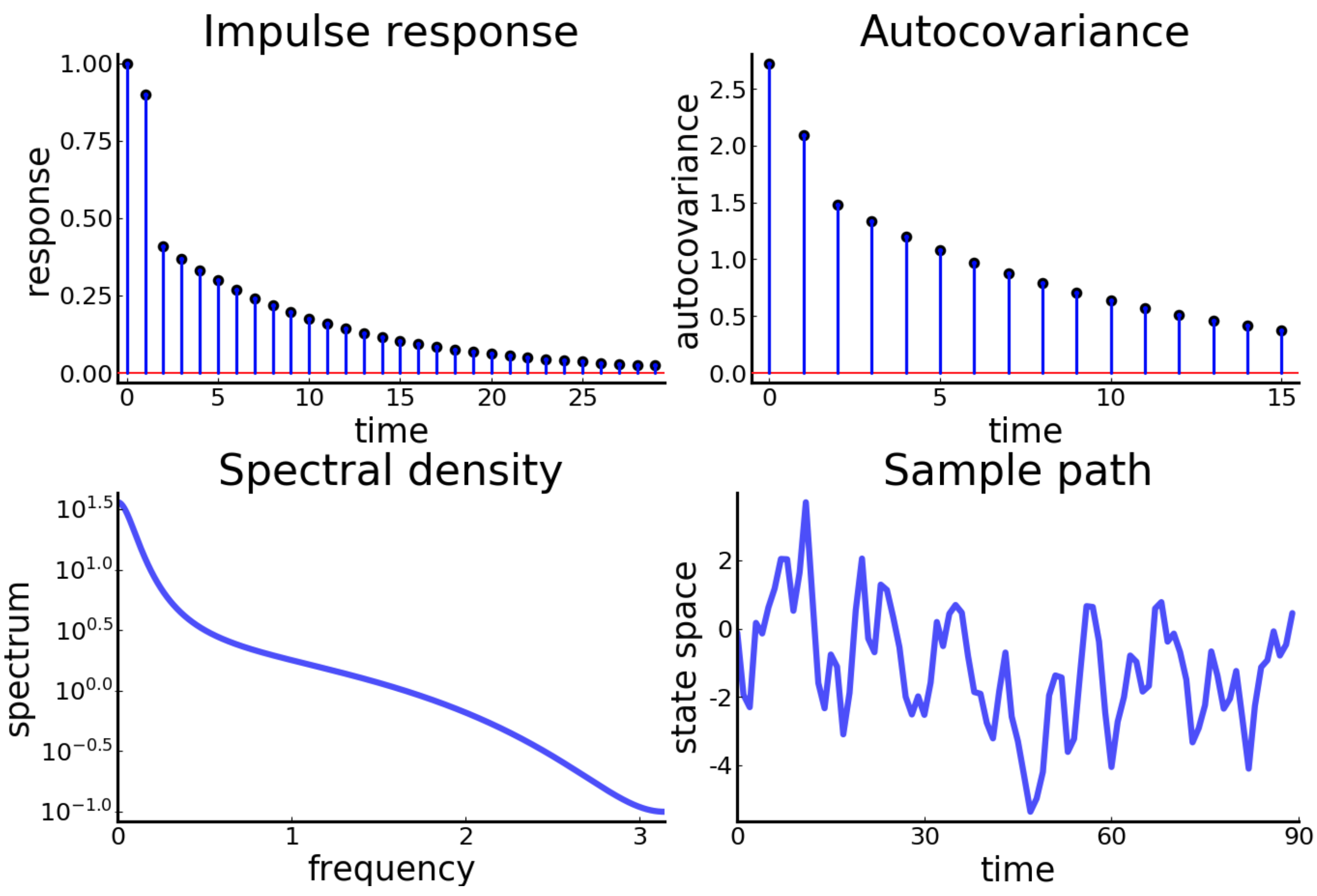
A general rule for interpreting PACF plots is that AR(p) models tend to cut off after lag p, MA(q) models tend to tail off as the lag level increases, and ARMA(p,q) models also tent to tail off in the same manner. The only difference between MA(q) and ARMA(p,q) models is that at ACF function also tails off in ARMA while it cuts off in MA(q) models in a similar manner to the AR(p) models PACF.
From the output we can see that both our PACF's for Caterpillar and Boeing tail off as lag increase (coded to lag 1 to 30). This means that we are likely, dealing with ARMA or MA models for our two stocks.

4. ARMA Modeling

A stochastic process {Xt} is called an autoregressive moving average process, or ARMA(p, q), can be written as:

$$X_t = \varphi_1 X_{(t-1)} + \cdots + \varphi_p X_{(t-p)} + e_t + \theta_1 e_{(t-1)} + \cdots + \theta_q e_{(t-q)}$$

There is an alternative notation for ARMA processes in common use, based around the back shift operator B:
$B^k Y_t := Y_{t-k}$



Fitting a necessary model gives us the output above. We can see that the ACF slowly tails off as we keep increasing the lag level, so we can determine a ARMA(p,q) model is needed. The specific model used to acquire the output above was an ARMA(1,2) model as it fits the data slightly better than any alternative that I have tried. Another method to determine p and q values is to calculate AIC values, however this is currently not supported in the available Time Series packages in Julia v1.1.

5. ARCH/GARCH Modeling

GARCH/ARCH (general autoregressive conditionally heteroscedastic) are models for the variance of a time series. GARCH models are used to describe a changing, possibly volatile variance. Although an ARCH model could possibly be used to describe the increase in variance over time, most often it is used in situations in which there may be short periods of increased variation(clustering). ARCH models were created in the context of econometric and finance problems having to do with the amount that investments or stock price increase (or decrease) over time, so commonly used as models for this type of data.

An uncorrelated time series can still be serially dependent due to a dynamic conditional variance process. A time series exhibiting conditional heteroscedasticity, ie autocorrelation can be tested with the Engle ARCH Test. The null hypothesis is that the series is white noise and the alternative is that the series is auto correlated. The test statistic is LM (similar to F statistic) for the regression of the auto correlation. Tests were conducted for both stocks below:

```
ARCH LM test for conditional heteroskedasticity (Boeing Close Price)
----------------------------------------------
Population details:
    parameter of interest:   T·R² in auxiliary regression of rₜ² on an intercept and its own lags
    value under h_0:         0
    point estimate:          13028.184742203026

Test summary:
    outcome with 95% confidence: reject h_0
    p-value:                 <1e-99

Details:
    sample size:                 13090
    number of lags:              1
    LM statistic:                13028.184742203026
```

```
ARCH LM test for conditional heteroskedasticity (Caterpillar Close Price)
----------------------------------------------
Population details:
    parameter of interest:   T·R² in auxiliary regression of rₜ² on an intercept and its own lags
    value under h_0:         0
    point estimate:          13028.184742203026

Test summary:
    outcome with 95% confidence: reject h_0
    p-value:                 <1e-99

Details:
    sample size:                 13090
    number of lags:              1
    LM statistic:                13028.184742203026
```

In both cases the null hypothesis is strongly rejected, and we conclude that there is evidence for the presence of volatility clustering at lag level 1. In other words, large changes in the closing prices of Caterpillar and Boeing stocks tend to cluster together.

Having established the presence of volatility clustering, we can begin by fitting the workhorse model of volatility modeling, a GARCH(1, 1), is fitted to both stocks.

```
TGARCH{0,1,1} model with Gaussian errors, T=13090.

Mean equation parameters:

    Estimate Std.Error z value Pr(>|z|)                          (Boeing Close Price)
μ    51.6586   0.90129 57.3162   <1e-99

Volatility parameters:

        Estimate      Std.Error      z value          Pr(>|z|)
ω       28.3111      1.32414        21.3808      <1e-99
β₁   5.47382e-48   0.0808632    6.76923e-47   1.0000
α₁      0.617964     0.0396168     15.5985      <1e-54
```

```
TGARCH{0,1,1} model with Gaussian errors, T=13090.

Mean equation parameters:

    Estimate Std.Error z value Pr(>|z|)                          (Caterpillar Close Price)
μ     53.106  0.607396 87.4323   <1e-99

Volatility parameters:

       Estimate        Std.Error     z value    Pr(>|z|)
ω       1.65504      1.28071      1.29229     0.1963
β₁   0.00375203   0.10574     0.0354837   0.9717
α₁         1.0          0.0961287   10.4027   <1e-24
```

Further model selection and fitting is done in the *#ARCH and GARCH Model* section of the code but has been committed for space purposes.

Reading

1. Time Series Analysis With Applications in R; Cryer, Jonathan D., Chan, Kung-Sik; Springer 2008
2. https://docs.julialang.org/en/v1/ (Julia Documentation)
3. http://quantecon.github.io/QuantEcon.jl/latest/api/QuantEcon.html#QuantEcon-1
4. htt/ps:/s-broda.github.io/ARCHModels.jl/stable/manual/
5. https://pkg.julialang.org/docs/TimeSeries/dmqCl/0.14.1/
6. https://pkg.julialang.org/docs/StatsBase/EZjIG/0.29.0/

Code

```
#Packages used
using TimeSeries, Plots, Dates, StatsBase, DelimitedFiles, QuantEcon, ARCHModels
pyplot()

#Data (Historic Financial Data)
Apple = readtimearray("AAPL.csv", format = "yyyy-mm-dd", delim = ',') #1986 - 2012
```

```julia
Boeing = readtimearray("BA.csv", format = "yyyy-mm-dd", delim = ',') #1962 - 2012
Caterpillar = readtimearray("CAT.csv", format = "yyyy-mm-dd", delim = ',') #1962 - 2012
Dell = readtimearray("DELL.csv", format = "yyyy-mm-dd", delim = ',') #1988 - 2015
Ebay = readtimearray("EBAY.csv", format = "yyyy-mm-dd", delim = ',') #1998 - 2016
Ford =readtimearray("F.csv", format = "yyyy-mm-dd", delim = ',') #1998 - 2016
GE = readtimearray("GE.csv", format = "yyyy-mm-dd", delim = ',') #1962 - 2016

#=
For this project we will be looking at the Boeing and Catipillar datasets
since they fall under the same timeframe.
=#

#Basic Plotting
plot(Boeing[:Open, :Close, :High, :Low], title = "Time Series (Boeing vs Caterpillar)",
        xaxis = "Timestamp", yaxis = "Price", palette = :dense)

plot!(Caterpillar[:Open, :Close, :High, :Low], palette = :turbid)

#Some Basic Methods
        #Variable order: Open   High    Low     Close
        #Autocovariance and Autocorrelation
        BA_close = readdlm("BA_close.txt")
        convert(Array{Real},BA_close)
        autocov_BA_close = autocov(BA_close, ; demean = true)
        autocor_BA_close = autocor(BA_close, ; demean = true)

        CAT_close = readdlm("CAT_close.txt")
        convert(Array{Real},CAT_close)
        autocov_CAT_close = autocov(CAT_close, ; demean = true)
        autocor_CAT_close = autocor(CAT_close, ; demean = true)

        plot(autocor_BA_close, title = "AutoCor of Historical Prices of Boeing Stock",
                xaxis = "lag", yaxis = "autocor", seriestype = :line)
        plot(autocov_BA_close, title = "AutoCov of Historical Prices of Boeing Stock",
                xaxis = "lag", yaxis = "autocov", seriestype = :line)

        #Cross-covariance and Cross-correlation
        crosscov_CAT_close = crosscov(CAT_close, BA_close, ; demean=true)
        plot(crosscov_CAT_close, title = "CrossCov of Historical Prices of Boeing vs Caterpillar",
                xaxis = "lag", yaxis = "autocov", seriestype = :line)
        crosscor_CAT_close = crosscor(CAT_close, BA_close, ;demean=true)
        plot(crosscor_CAT_close, title = "CrossCor of Historical Prices of Boeing vs Caterpillar",
                xaxis = "lag", yaxis = "autocor", seriestype = :line)

        #Partial Autocorrelation Function
        pacf_CAT_close_reg = pacf(CAT_close,[2,3,4,5,10,15,20,25,30]; method=:regression)
        plot(pacf_CAT_close_reg, title = "PACF_Regression of Caterpillar",
                xaxis = "lag", yaxis = "pacf", seriestype = :bar)
        pacf_CAT_close_yule = pacf(CAT_close,[2,3,4,5,10,15,20,25,30]; method=:yulewalker)
        plot(pacf_CAT_close_yule, title = "PACF_Yule of Caterpillar",
                xaxis = "lag", yaxis = "pacf", seriestype = :bar)
        pacf_BA_close_reg = pacf(BA_close,[2,3,4,5,10,15,20,25,30]; method=:regression)
```

```
    plot(pacf_BA_close_reg, title = "PACF_Regression of Boeing",
        xaxis = "lag", yaxis = "pacf", seriestype = :bar)
    pacf_BA_close_yule = pacf(BA_close,[2,3,4,5,10,15,20,25,30]; method=:yulewalker)
    plot(pacf_BA_close_yule, title = "PACF_Yule of Boeing",
        xaxis = "lag", yaxis = "pacf", seriestype = :bar)

    #ARMA Model Example for Boeing Data
    phi = 0.5
    theta = [0.0, -0.5]
    sigma = 1.0
    model1 = ARMA(phi, theta, sigma)
    quad_plot(model1)

    #ARCH and GARCH Models of Boeing Data
    BA_close2 = readdlm("BA_close.txt")[:,1]
    CAT_close2 = readdlm("CAT_close.txt")[:,1]
    autocor(BA_close.^2, 0:10, demean=true)
    autocor(CAT_close.^2, 0:10, demean=true)

    ARCHLMTest(BA_close2, 1)
    #The null is strongly rejected, again providing evidence for the presence of volatility clustering at lag=1
    ARCHLMTest(CAT_close2, 1)
    #The null is strongly rejected, again providing evidence for the presence of volatility clustering at lag=1

    fit(GARCH{1, 1}, BA_close2)
    fit(GARCH{1, 1}, CAT_close2)

    #Regression
    X = ones(length(BA_close2), 1)
    reg = Regression(X)
    fit(GARCH{1, 1}, BA_close2; meanspec=reg)
    fit(GARCH{1, 1}, CAT_close2; meanspec=reg)
    #Both the CAT and BA datasets have the same number of observations on the same timeframe.
    #Notice that because in this case X contains only a column of ones, the estimation results,
    #are equivalent to those obtained with our fits above.
    fit(EGARCH{1, 1, 1}, BA_close2; meanspec=NoIntercept, dist=StdT)
    fit(EGARCH{1, 1, 1}, CAT_close2; meanspec=NoIntercept, dist=StdT)

    selectmodel(ARCH, BA_close2; criterion=aic, maxlags=2, dist=StdT)
    selectmodel(GARCH, BA_close2; criterion=aic, maxlags=2, dist=StdT)
    selectmodel(EGARCH, BA_close2; criterion=aic, maxlags=2, dist=StdT)
    selectmodel(ARCH, CAT_close2; criterion=aic, maxlags=2, dist=StdT)
    selectmodel(GARCH, CAT_close2; criterion=aic, maxlags=2, dist=StdT)
    selectmodel(EGARCH, CAT_close2; criterion=aic, maxlags=2, dist=StdT)


#Functions for ARMA building
function plot_spectral_density(arma::ARMA)
   (w, spect) = spectral_density(arma, two_pi=false)
   p = plot(w, spect, color=:blue, linewidth=2, alpha=0.7,
         xlims=(0, pi), xlabel="frequency", ylabel="spectrum",
         title="Spectral density", yscale=:log, legend=:none, grid=false)
   return p
end

function plot_autocovariance(arma::ARMA)
   acov = autocovariance(arma)
   n = length(acov)
   N = repeat(0:(n - 1), 1, 2)'
   heights = [zeros(1,n); acov']
   p = scatter(0:(n - 1), acov, title="Autocovariance", xlims=(-0.5, n - 0.5),
        xlabel="time", ylabel="autocovariance", legend=:none, color=:blue)
   plot!(-1:(n + 1), zeros(1, n + 3), color=:red, linewidth=0.5)
   plot!(N, heights, color=:blue, grid=false)
   return p
end

function plot_impulse_response(arma::ARMA)
   psi = impulse_response(arma)
   n = length(psi)
   N = repeat(0:(n - 1), 1, 2)'
   heights = [zeros(1,n); psi']
   p = scatter(0:(n - 1), psi, title="Impulse response", xlims=(-0.5, n - 0.5),
        xlabel="time", ylabel="response", legend=:none, color=:blue)
   plot!(-1:(n + 1), zeros(1, n + 3), color=:red, linewidth=0.5)
   plot!(N, heights, color=:blue, grid=false)
   return p
end

function plot_simulation(arma::ARMA)
   X = simulation(arma)
   n = length(X)
   p = plot(0:(n - 1), X, color=:blue, linewidth=2, alpha=0.7,
         xlims=(0.0, n), xlabel="time", ylabel="state space",
        title="Sample path", legend=:none, grid=:false)
   return p
end

function quad_plot(arma::ARMA)
   p = plot(plot_impulse_response(arma), plot_autocovariance(arma),
        plot_spectral_density(arma), plot_simulation(arma))
   return p
end
```