Trade Strategy Analyzer: R Package for Statistical Arbitrage and High Frequency Trading Simulation and Visualization

Contributors:

Winnie Cheng (wwcheng@alum.mit.edu)

Tirto Adji (tirto_adji@yahoo.com)

Overview:

Statistical Arbitrage is a set of trading strategies that take advantage of temporary asset mispricing to profit from the market. Mispricing can be detected by the use of data mining techniques and leveraging observed statistical properties of stocks to forecast future movements. This project aims at developing an R package and associated visualization tools for traders to analyze the effectiveness of their strategies. The package also includes a small set of sample strategies that the open-source community can expand on.

We focus on the S&P 500 dataset for this package. However, other data sources can be used as long as they are adapted to our input csv format. Daily stock prices of S&P500 component stocks and their characteristics can be obtained from:

- http://www.standardandpoors.com/indices/sp-500/en/us/?indexId=spusa-500-usduf--p-us-l--
- http://pages.swcp.com/stocks/

General Concept behind Statistical Arbitrage:

The fundamental principle behind statistical arbitrage is in 1) identifying stocks that are correlated, and 2) exploiting the mean-reverting behavior to expose trading opportunities.

Correlation of stocks can be determined in numerous ways. For example, one can make a high-level assumption that stocks in the same industry are correlated and choose stock pairs based on industry data alone. Other techniques are also possible, which may involve more elaborate computation of correlation coefficients between two stocks at different points in time. After determining pairs of correlated stocks, we apply the mean-reverting algorithm. For example, for two stocks A and B that are highly correlated, if stock A rises and stock B falls, and the difference is 'wide enough', the mean-reverting principle suggests that one should long (buy) B and short (sell) A, since the assumption is that the two stocks will 'revert' to the same mean later on. When the stocks do revert, we close out the initial positions to make a profit.

Architecture:

The package is originally written as two separate R packages, one to simulate the trade (TradeStrategySimulator) and the other (TradeStrategyVisualizer) to visualize the results. The

two components communicate via a SQLite database which stores tables containing simulation results and characteristics of the stock data.

A. Simulation

The TradeStrategySimulator assumes that stock information and stock market data (i.e., stock prices time-series) are available in comma-delimited (.csv) files. When the simulator starts up, it reads in this information into R data frames and also writes them to the sqlite database. This data is used by the simulator logic itself but may also be used by individual trading algorithms.

For each evaluation of a trading algorithm, one creates a new portfolio specifying an initial investment amount. Then, to run the trading algorithm, one invokes the algorithm with the required information (e.g., recent history of stock prices) for that algorithm. All algorithms are designed to output an "order book" as their result. The order book is a data frame specifying the trade actions recommended by the algorithm. In the current version, the trade actions are: "BUY" and "SELL". The order specifies the ticker symbol of the stock to purchase or sell, and the number of shares.

The simulation logic takes in a portfolio Id and an order book and simulates the trades on stock market data. It maintains state in order to perform this simulation, keeping track of the portfolios, asset holding information for different portfolios, transactions executed, in its simulation environment. One can run different algorithms by creating a new portfolio and generating an order book with the desired algorithm on it. The simulation keeps updating the data frames PortfoliosInfo, HoldingsInfo, and TransactionsInfo. At the end of simulation, one can then call a function to store these data into sqlite database.

B. Trading Strategies

We provide 2 sample strategies in StrategyBuyHold.R and StrategyMeanRevert.R . These are intended as examples on how to develop a trading strategy that can be evaluated and visualized in our package.

1. *Random Buy-and-Hold*

   This algorithm selects a set of stocks at random and holds them for the entire trading period. The user can specify how many stocks he/she wants in this portfolio. This is a good baseline algorithm for comparing new techniques against.

2. *Mean Revert*

   This algorithm demonstrates the mean-reverting algorithm, allowing the user to configure the number of pairs to be considered in trading decisions. The current version uses Pearson correlations on stock price time-series to determine which

stocks are highly correlated. (There are more advanced and accurate techniques that could have been used. This is just an example.) The algorithm is flexible in allowing plug-ins for new correlation computation techniques. This is only one example.

C. Visualization

There are 2 R packages used for visualization: ggplot2 and googleVis. They are both powerful visualization tools that help graphically represent sets of data. They act as visual aids to the user to understand the data easier, especially when the data is abstract and large in size. In this case, we use them to generate charts to compare performance between different trading algorithms and stock prices over a time period, as well as movement of stock prices. For demonstration, the charts make use of S&P 500 stock data from the 2009-08-21 to 2010-08-20 time period.

ggplot2:
We produce 6 charts using ggplot2.

1. Portfolio Holding Summary Chart
This chart compares the performance of different trading algorithms: mean reverting and random buy hold. It plots the holding value on the Y-axis and date on the X-axis. It shows that the mean reverting algorithm gives a higher overall return compared to the random buy hold algorithm. We also added summary statistics, a linear fit, for each algorithm. The R function that produces this chart also accepts optional caption text to annotate the graphs.

2. Stock Correlation Matrix Chart
This chart shows the degree of relationship (correlation) between two ticker symbols. It plots the top positions of positively correlated stocks. It uses ggplot2's geom_tile() and the computed correlation number as its fill.

3. Stock Correlation Pairs Cartesian Chart
This chart plots the top positions of positively correlated stocks in pairs. Each pair has its own scale and it displays the price movement of the two tickers which are closely related. As a result, the stock pairs shown here have similar trends. It uses facet_grid() and geom_line() as the grammar of the graphics.

4. Portfolio Stock Performance Cartesian Chart
This chart displays stock price movements in our portfolio, each with a smoothed line that follows each original line, and a smoothed line that applies to the entire group. This way it's easier to identify which stocks are performing well, and which are not.

5. Portfolio Stock Prices BoxPlot
This boxplot chart is useful to identify outliers and the box plots are sorted from low to high stock prices from left to right. The users can quickly glance and detect anomaly, if any, in the stock prices. Larger boxplots mean more skewed distributions.

6. Stock Calendar Heat Map

This chart represents the individual stock prices time series data as a calendar with days filled with colors representing the values. Red means higher prices, green means lower prices. The chart makes it an interesting way to look at financial time series data. The users can quickly scan the chart and see the data partitioned by week, week day, and year. It uses geom_tile(), scale_fill_gradient() and facet_wrap() as the ggplot2's grammar.

googleVis:

As an experiment, we use the googleVis package to display the same data. We use annotated time line and motion charts. Here are their short descriptions:

1. Annotated Time Line Chart

This chart dynamically displays the movement of stock prices and event annotations (BUY/SELL) at any point in time. At the bottom at the chart, there is a slider that can be used to zoom in and out to a specific time period.

2. Motion Chart

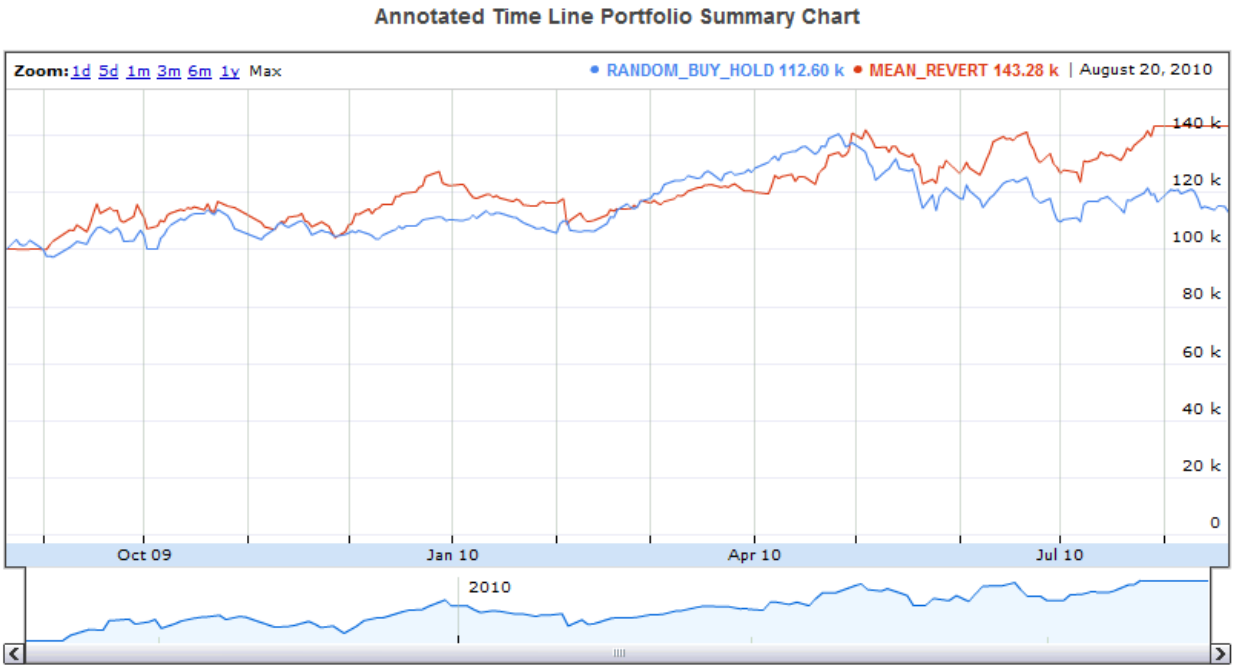
This chart dynamically shows the performance of the different algorithms used in the trade strategy simulator. The chart allows users to select and explore several variables over time. It uses animation to depict the movement of the variables. The motion chart measures the growth rate of one portfolio's value using a specific algorithm compared to the other.

Example on How to Use Package

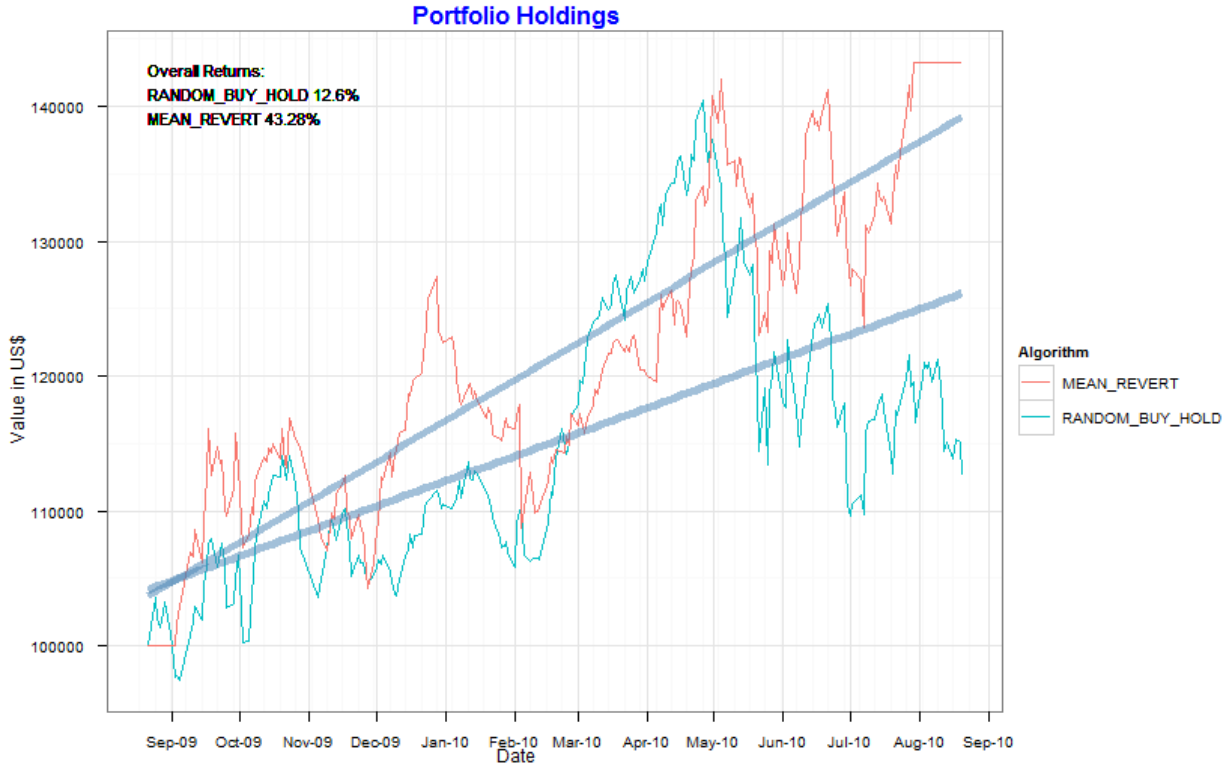
The best way to examine the capabilities of this package is to run TestEndToEnd.R which simulates the two strategies and compares their results visually.

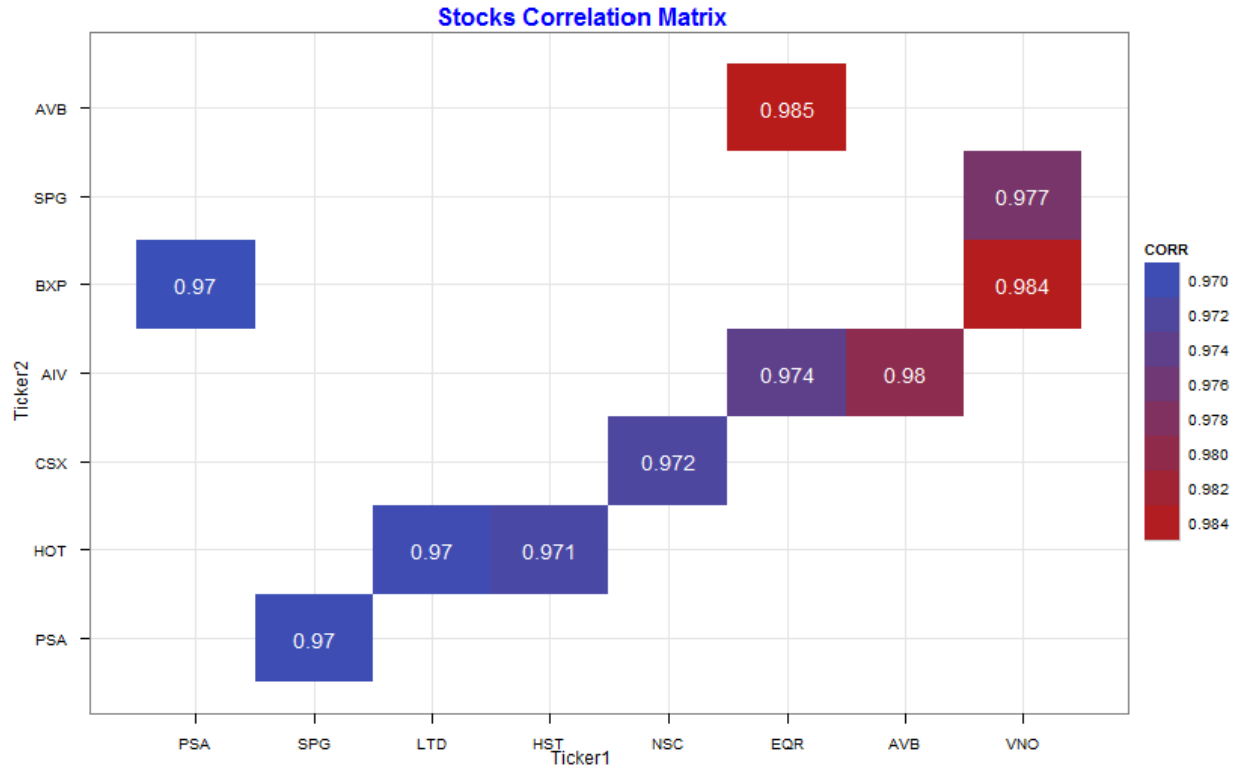
To add a new trading strategy, you may find it helpful to examine StrategyMeanRevert.R and how it is invoked in TestEndToEnd.R.

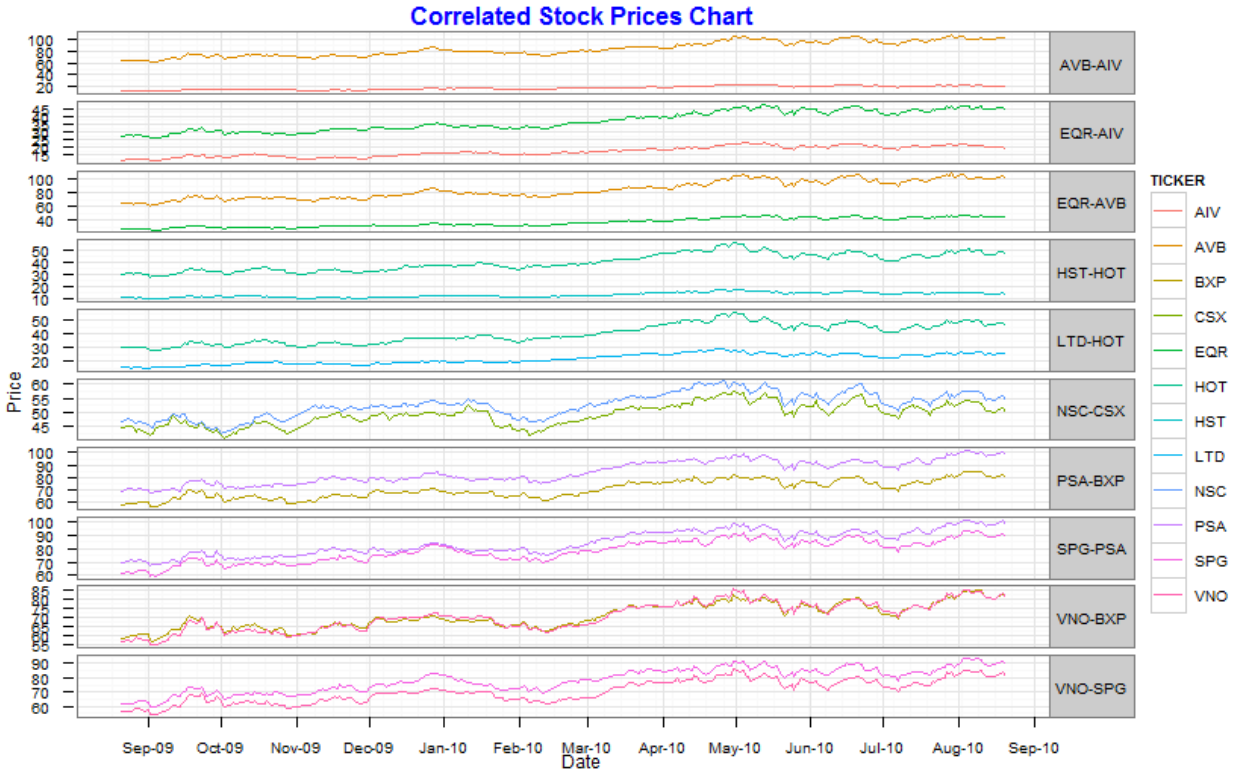
Sample Charts



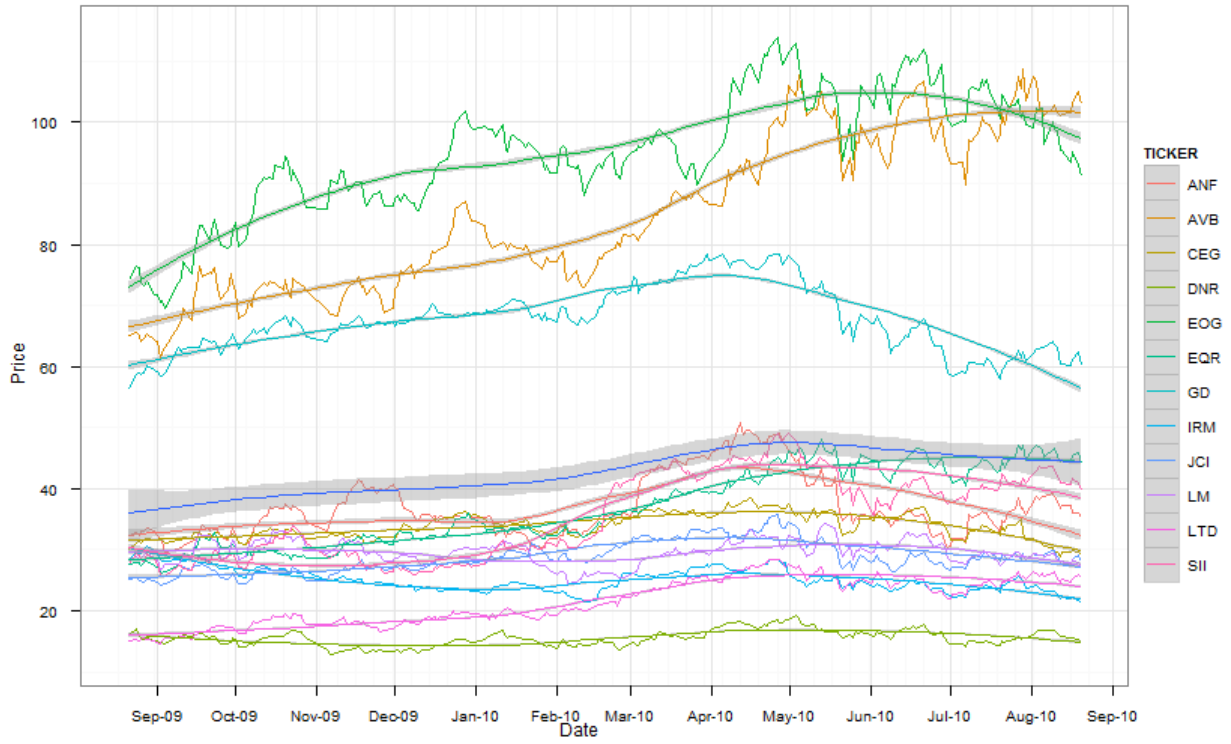
Trade Strategy Simulation Result of Mean Reverting vs. Buy Random and Hold Algorithm



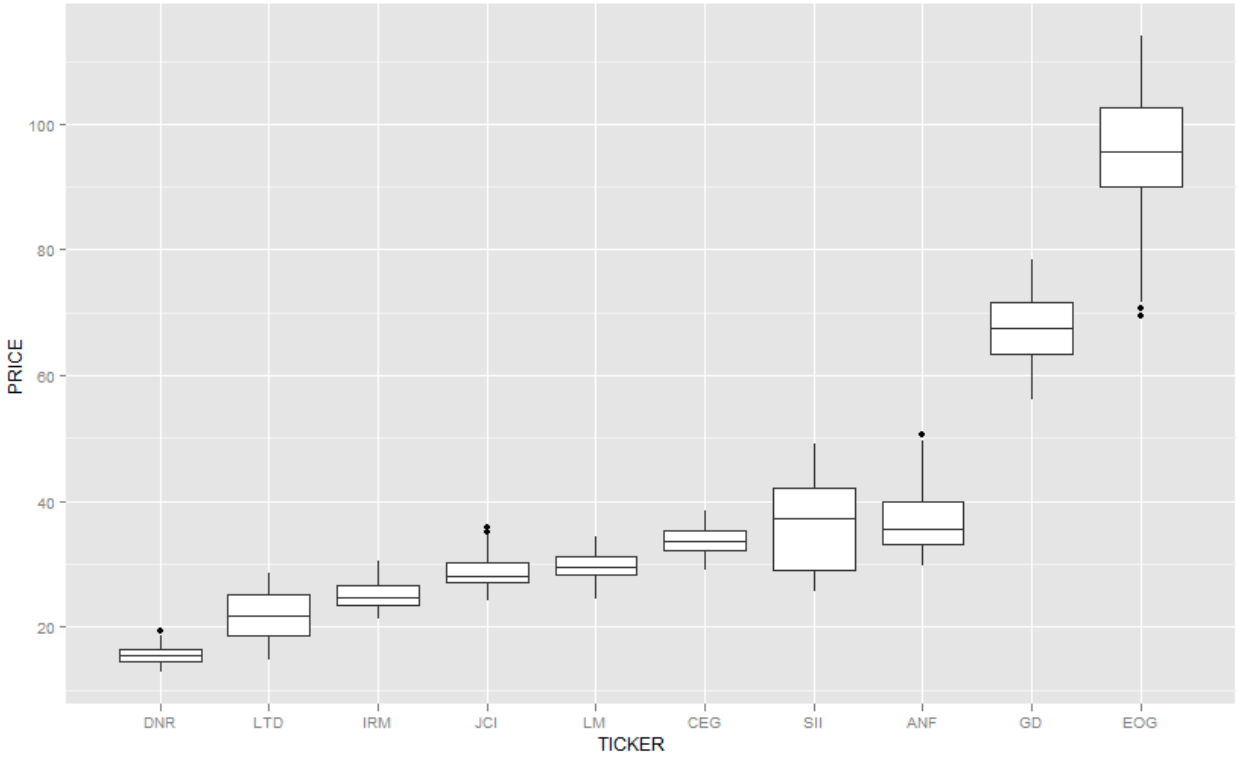

Stocks Correlation Matrix
AVB
SPG
BXP
AIV
CSX
HOT
PSA
Ticker2
0.985
0.977
0.97
0.984
0.974
0.98
0.972
0.97
0.971
0.97
PSA
SPG
LTD
HST
NSC
EQR
AVB
VNO
Ticker1
CORR
0.970
0.972
0.974
0.976
0.978
0.980
0.982
0.984

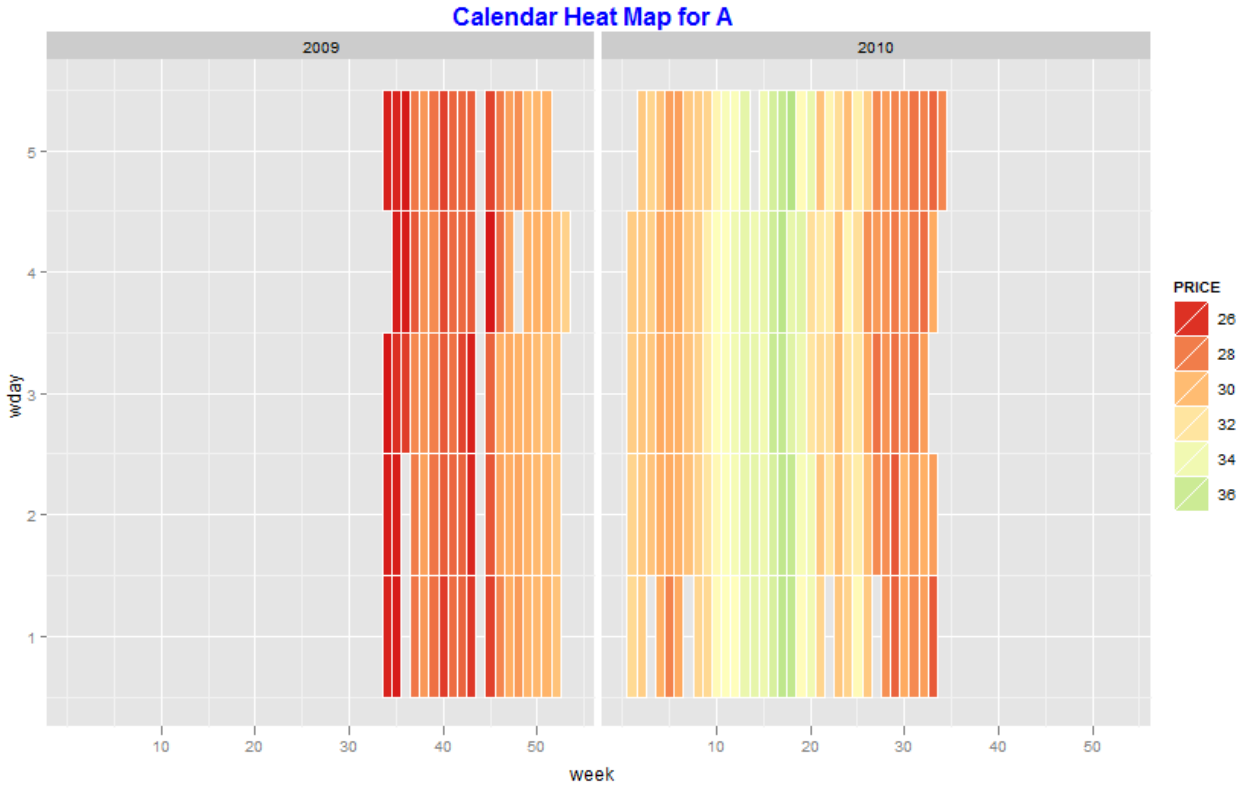

Correlated Stock Prices Chart
AVB-AIV
EQR-AIV
EQR-AVB
HST-HOT
LTD-HOT
NSC-CSX
PSA-BXP
SPG-PSA
VNO-BXP
VNO-SPG
Price
Sep-09
Oct-09
Nov-09
Dec-09
Jan-10
Feb-10
Mar-10
Apr-10
May-10
Jun-10
Jul-10
Aug-10
Sep-10
Date
TICKER
AIV
AVB
BXP
CSX
EQR
HOT
HST
LTD
NSC
PSA
SPG
VNO

Portfolio Stock Prices Chart



Stock Prices BoxPlot




Calendar Heat Map for A
2009
2010
wday
week
PRICE
26
28
30
32
34
36