

Project Report

on

“MV Portfolio Optimization”

Towards Partial Fulfillment
of the Requirements for the Degree of Master of Science in
Information Systems and Decision Sciences

California State University, Fullerton

ISDS 570 – Business Data Transformation

Team E5

Bansal, Benu (bansalbenu@csu.fullerton.edu)

Ebrahimi, Farhang (mikeebrahimi@csu.fullerton.edu)

Eftekharian, Sepehr (sepehr485@csu.fullerton.edu)

Henhapl, Elisabeth (e.henhapl@csu.fullerton.edu)

May 10, 2020

ABSTRACT

The ironic thing about the stock market is that a vast majority of people trading on it think they can beat it (Statman, 2019).ⁱ Supposedly, they believe they can come up with a portfolio (defined as a “grouping of financial assets such as stocks, bonds, commodities, currencies and cash equivalents, as well as their fund counterparts, including mutual, exchange-traded and closed funds”ⁱⁱⁱ) that has a higher total return than the market average. If you ask them, they might tell you that they know stocks better than the average person, or that they keep an eye on a few stocks and always know at what price to buy or sell. Unfortunately, the average investor receives less return on individual investments than they would through most stock index portfolios, arguably proving their confidence in themselves wrong. For this project, we are putting our investment capabilities to the test by creating our own S&P 500 portfolio and comparing it to the market average, testing whether our cumulative return can in fact beat the index. Who knows? We may get intricate forecasting insights and have a whole new endeavor on our hands.

TABLE OF CONTENTS

ABSTRACT.....	2
TABLE OF FIGURES	4
0. MOTIVATION	5
1. INTRODUCTION	6
1.1 S&P 500 Stocks.....	6
1.2 Project Portfolio	7
2. METHODOLOGY.....	9
3. ETL PROCESS IN DETAIL	11
3.1 PostgreSQL - Database Setup	11
3.1.1 Eod_quotes.....	11
3.1.2 Eod_indices.....	12
3.1.3 Custom_calendar.....	13
3.2 R – Data Preparation	14
3.2.1 Connecting R to PostgreSQL.....	14
3.2.2 Checking for completeness & transforming tables for pivoting.....	14
3.2.3 Using dates & merging with the custom calendar.	14
3.2.4 Imputing missing data.....	15
3.3 R – Analysis & Forecasting.....	15
3.3.1 Tabular Return Data Analytics.	15
3.3.2 Graphical return data analytics & cumulative returns chart.	16
3.3.3 MV Portfolio Optimization.....	18
3.3.4 Value & Hypothetical Portfolio Returns.....	18
4. RESULTS.....	19
5. CONCLUSION & FUTURE SCOPE	23
APPENDIX: FULL SQL / R CODE.....	24
REFERENCES (ENDNOTES)	36

TABLE OF FIGURES

Figure 1 - <i>Selected Stock Market Tickers with Respective Company Names and Stock Exchange</i>	7
Figure 2 - <i>Initial HOS & HOV Tickers Changed with HA & HAE due to Negative Cumulative Returns</i>	8
Figure 3 - <i>Column Analysis of the quandl_wiki.csv in R due to Large File Size</i>	11
Figure 4 - <i>Column Analysis and Preparation of the SP500TR.csv in Excel</i>	12
Figure 5 - <i>Column Analysis of the custom_calendar.csv in Notepad after Setup in Excel</i>	13
Figure 6 - <i>Return Distribution Comparison for Selected Stock Tickers and the S&P 500 Benchmark Sorted by Variance</i>	17
Figure 7 - <i>Drawdown Chart Outlining the Negative Performance of Individual Stock Tickers</i> ..	17
Figure 8 - <i>Cumulative Return Chart for Selected Tickers</i>	19
Figure 9 - <i>R Output for Optimized Portfolio Weights and Their Sum</i>	20
Figure 10 - <i>Summarized Optimized and Sorted Portfolio Weights</i>	20
Figure 11 - <i>Cumulative Return Chart for Optimized Portfolio versus S&P 500 Benchmark</i>	21
Figure 12 - <i>R Output for Cumulative Return Results</i>	21
Figure 13 – <i>R Output for Annualized Daily Returns for Optimized Portfolio versus S&P 500 Benchmark</i>	21

0. MOTIVATION

The primary motivation for this report is to test our ability to Extract, Transform, and Load (ETL) non real-time data for the stock market to compare the return of a group of stocks to the S&P 500 benchmark. Learning how to use ETL for investments is a useful skill for everyone interested in trading. This includes investment managers attempting to identify portfolios that can either stay on par or perhaps even beat the market average as well as individuals trying to make a return in the process. If nobody utilizes forecasting tools, what is the point of using a portfolio manager and paying their rate anyway? After all, depending on the type of fund you purchase, transaction costs can be as high as 4.5% of what you decide to invest, meaning even a modest investment of \$20,000 would result in a fee of \$900 right off the top.

There are several secondary motivations for this project as well. For one, this is an opportunity to test any group stocks and compare them to the market average. Another interesting aspect is to observe how different tickers interact with the benchmark based on the overall market developments. Most stock prices rise when the market is booming, yet this in no way applies to all of them. Perhaps it is even more important to note that some stocks even exceed the market in favorable circumstances but are considered highly risky, while other stocks yield a lower return with less volatility in those same market conditions. Observing the performance of respective tickers and perhaps analyzing why they fluctuate in certain patterns can educate investors on how to best allocate their money in the S&P 500 in the future.

1. INTRODUCTION

For our project, we have decided to use the following stocks with their respective ticker abbreviations in parenthesis: The Boeing Company (BA), Bank of America (BAC), Bristol-Myers Squibb Company (BMY), eBay (EBAY), Ebix Inc. (EBIX), US Ecology (ECOL), Encore Capital Group (ECPG), Euronet Worldwide (EFT), Equity Lifestyle Properties (ELS), Hawaiian Holdings, Inc. (HA), and Haemonetics (HAE), Honeywell International (HON). These tickers will be used in the portfolio that is to be compared to the S&P 500 as a whole.

The approach we are taking is not only about beating the market - worrying about winning and losing can take away from the learning experience, which is the real objective. Once we have perfected the process of how to compare the portfolio to the market average, we can do so in the future with even more of a focus on a specific strategy, such as picking stocks that are considered riskier, but may also have more upside, with the help of forecasting packages in R.

1.1 S&P 500 Stocks

S&P Dow Jones Indices maintains the S&P 500 stock market index. It consists of 505 stocks which are issued by 500 large market capitalization companies and cover approximately 80% of the American equity market. The S&P 500 stock index is the indicator of the best-performing stocks according to the market capitalization traded on American stock exchange. Since the S&P 500 is taken as an average for the entire U.S. stock market, it makes trading predictable and profitable. Most of the traders use the S&P 500 stocks as speculation, hedge market risk and delta-neutral trading. There are specific rules to be considered as one of the S&P 500 stocks which need to fulfil are listed as follows:

- The capital must be legally established in the United States.
- The market capitalization of the stock should be at least 5.3 billion dollars.

- The stock must trade at a reasonable share price at the U.S. stock exchange.
- The stock should have at least 50% of its shares in the hands of the public.
- The stock must have positive earnings reported in the recent quarter.

Although meeting the requirements, as mentioned above, do not guarantee that the stock will be added to the S&P 500 stock index. It also depends on the committee, which further assesses the attributes of the capital.

1.2 Project Portfolio

For this project, each member selected three stocks from the U.S. market and made a project's portfolio. We compared project's stocks with S&P 500 stocks to analyze the performance of the selected stocks. With the goal to achieve a higher return than the benchmark in mind, the tickers HOS and HOV were exchanged with HA and HAE. The customized portfolio stocks are listed in the form of Figure 1 below:

Stock Market Tickers	Company Name	Stock Exchange
BA	Boeing Company	NYSE
BAC	Bank of America	NYSE
BMJ	Bristol-Myers Squibb Company	NYSE
EBAY	eBay	NASDAQ
EBIX	Ebix Inc.	NASDAQ
ECOL	US Ecology	NASDAQ
ECPG	Encore Capital Group	NASDAQ
EEFT	Euronet Worldwide	NASDAQ
ELS	Equity Lifestyle Properties	NYSE
HA	Hawaiian Holdings, Inc.	NASDAQ
HAE	Haemonetics	NYSE
HON	Honeywell International	NYSE

Figure 1 - Selected Stock Market Tickers with Respective Company Names and Stock Exchange

We would like to remark that the choice of these tickers was an iterative process. Initially, the same tickers as chosen by individual team members were used for analysis (homework 2 assignment). However, the cumulative return of a portfolio built with these stocks was severely underperforming. As a remedy, the two lowest tickers in terms of cumulative returns ('HOS' and 'HOV') were identified, which is also visualized in Figure 2. For the further process, the ticker selection was thus changed to the listings in Figure 1.

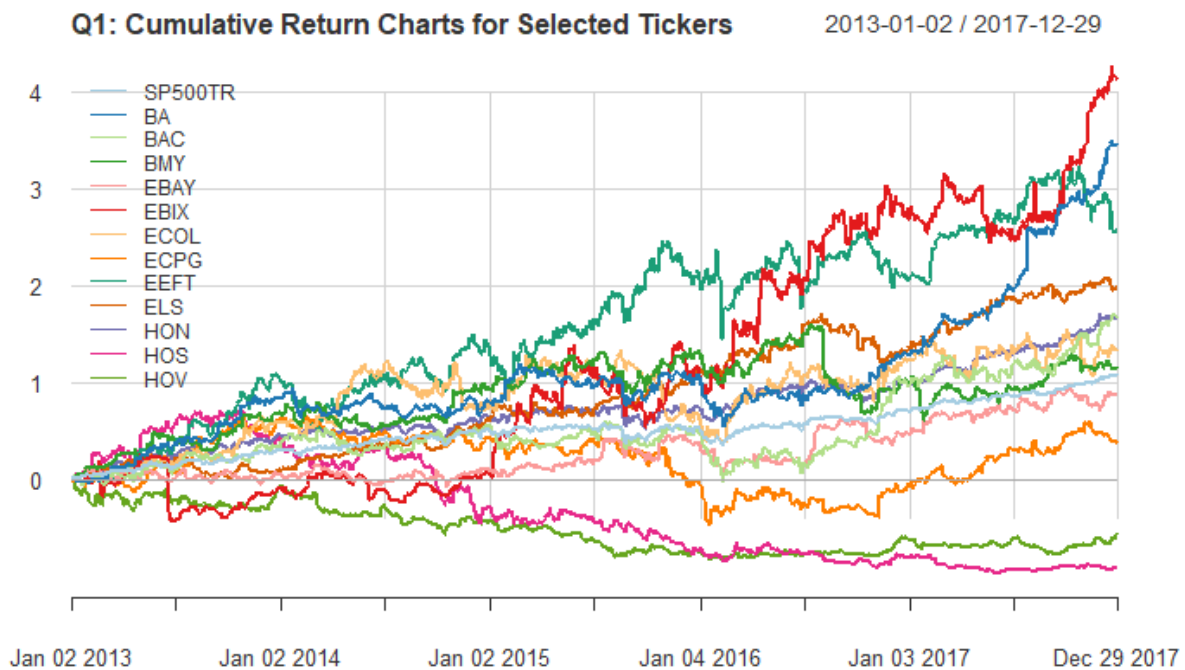


Figure 2 - Initial HOS & HOV Tickers Changed with HA & HAE due to Negative Cumulative Returns

2. METHODOLOGY

Backtesting is a common methodology used to calculate the viability of our trading strategy. Positive results on backtesting mean that the trader can employ this plan of action on the selected portfolio going forward. Using this method, an investor may simulate this strategy using historical data to publish the results and analyze the risk and profitability. A well-conducted backtest with positive results provides confidence to the trader that the strategy is solid and yields profits when implemented. A backtest with suboptimal results will signal to the trader that they need to adjust their strategy for a more desirable return. The ideal scenario is using a dataset as a sample that is showing a variety of market conditions over time. This way, it is easier to judge whether the result represents a fluke or a valid spike.

In this project, each of the team members selected three tickers that were subsequently combined to a total of 12 for a customized, profitable portfolio. A forecasting model was then trained based on the backtesting method to forecast future stock values. To lower the risk, the S&P 500 market index was chosen as a benchmark to compare the results of our portfolio to. The historical data selected was retrieved from the Quandl dataset for the last five years (2013-2018) in a CSV format. From this dataset, three months of data (Jan 2018 –March 2018) were used for forecasting, while the rest of the data from 12/31/2012 to 12/31/2017 was applied to model training. Further, we created a custom calendar ranging from 12/31/2012 to 03/31/2018 to represent the market trading days and exempt days.

Using the open-source software PostgreSQL, we set up the needed database with two main tables. First, eod quotes, which contains historical data for individual stocks, and second, eod indices, which includes SP500 historical data to see the open, high, low, and adjusted close prices of each day for the SP500 index. Then, we used R to import specific data ranges from

PostgreSQL for our analysis, while providing visualizations for better interpretation of the results. The DBI package was loaded to connect R to Postgres. Then, queries to get the eod indices for S&P 500 and the eod quotes for individual stocks from the last day of Dec 2012 up to the 27th of March 2018 were executed.

Using the 'eod' variable, we extracted the data in union (between eod quotes and eod indices) from the query result which met the requirements of 99 or more percent completeness. Next, we extracted the trading days from the custom calendar to use in our analysis. Finally, we merged the calendar with the filtered data to have all the needed information including the date for each stock into one single data frame. To compare our portfolio with S&P 500, a reference point to monitor the transactions was created. For this purpose, the date column from the union was extracted and used as an identifier for each row respectively. Then, missing data was replaced with historical values, with the restriction that no more than three consecutive rows held NA or NAN values.

The next step after cleaning and structuring the data was to calculate the annualized return using the Performance Analytics package. For the completion of this step, the maximum daily return was calculated and entries with less than one as a maximum daily return were removed. Then, the prepared data was backed up as a CSV file. The final steps are calculating annualized return of our portfolio versus the S&P 500 benchmark, including cumulative returns, and the optimized MV (Markowitz 1950s) portfolio weights based on historical data. For forecasting, as mentioned before, the first three months of 2018 were analyzed. Finally, we visualized the outcome with different charts such as cumulative return charts, drawdown graphs, and boxplots.

3. ETL PROCESS IN DETAIL

3.1 PostgreSQL - Database Setup

3.1.1 Eod_quotes.

In order to set up the database needed, an SQL script was written that consists of database table creation, data import and handling, as well as role creation including the granting of rights. For the setup of the quotes table, it was necessary to identify columns in the `quandl_wiki.csv`. As this file requires 1GB in storage capacity, it was read into R to be able to retrieve column names as opening a .csv of this size through Excel would most likely result in a fatal crash. In line with the retrieved results displayed in Figure 3, the table `eod_quotes` with the columns `ticker`, `date`, `open`, `high`, `low`, `close`, `volume`, `ex.dividend`, `split_ratio`, `adj_open`, `adj_high`, `adj_low`, `adj_close`, and `adj_volume` was created. The primary key for this table was specified through the composite key of `ticker` and `date`. Then, ownership was assigned to `postgres` and the data was imported from the `quandl_wiki.csv` file.

```
> head(quandl)
  ticker      date  open  high   low close  volume ex.dividend split_ratio adj_open adj_high adj_low adj_close adj_volume
1      A 1999-11-18 45.50 50.00 40.00 44.00 44739900         0          1 31.0420 34.1120 27.2896 30.0186 44739900
2      A 1999-11-19 42.94 43.00 39.81 40.38 10897100         0          1 29.2954 29.3363 27.1600 27.5489 10897100
3      A 1999-11-22 41.31 44.00 40.06 44.00  4705200         0          1 28.1834 30.0186 27.3306 30.0186  4705200
4      A 1999-11-23 42.50 43.63 40.25 40.25  4274400         0          1 28.9952 29.7662 27.4602 27.4602  4274400
5      A 1999-11-24 40.13 41.94 40.00 41.06  3464400         0          1 27.3783 28.6132 27.2896 28.0128  3464400
6      A 1999-11-26 40.88 41.50 40.75 41.19  1237100         0          1 27.8900 28.3130 27.8013 28.1015  1237100
```

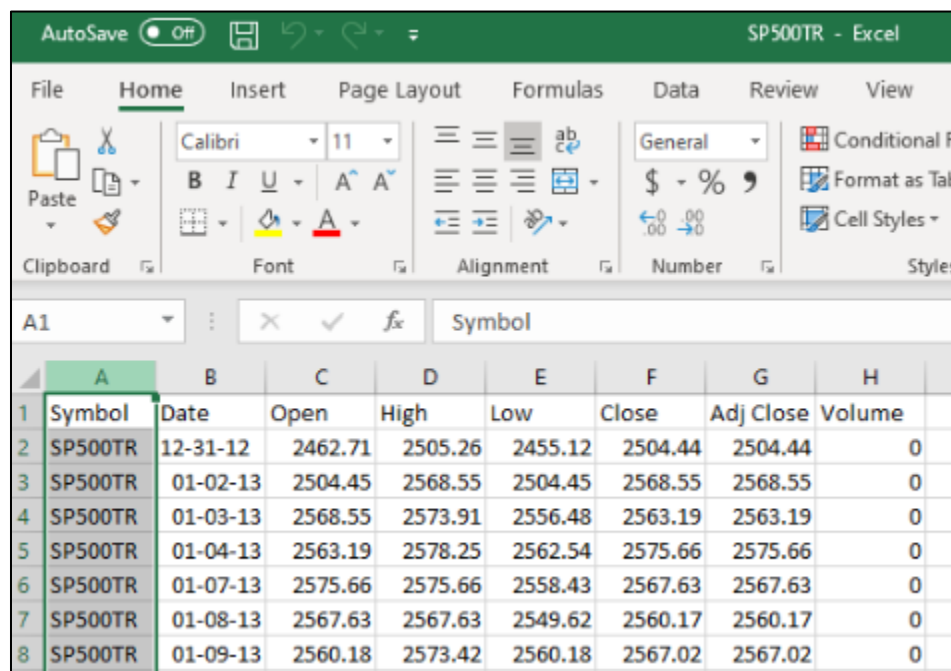
Figure 3 - Column Analysis of the `quandl_wiki.csv` in R due to Large File Size

Next, successful data import was checked through the queries displaying the top rows and returning the result of around 13 million rows which was cross-referenced with the previously mentioned analysis in R, specifically pertaining to a count of 12,926,649 rows. It was further important to identify the date range included in the `quandl` data, resulting in the dates from

1962/01/02 to 2018/03/27. The information that the last few trading days of March 2018 are missing was an important insight as to which date to use in the forecasting analysis in R.

3.1.2 Eod_indices.

Then, index data of S&P 500 was imported from <https://finance.yahoo.com/quote/%5ESP500TR/history?p=^SP500TR>ⁱⁱⁱ, specifying the date range of interest from 2012/12/31 to 2018/03/27. While the first date was chosen with the goal in mind to conduct training on data between 2013 and 2017 (i.e. date range specified on day before start day), the second date corresponds to the final available data entry in the quandl_wiki.csv as outlined above. The file was opened in Excel to identify the columns and is depicted in Figure 4. In addition, the column “Symbol” was inserted to provide for similar structure and respective eod_indices identifiers as can be found in the “Ticker” column in eod_quotes.



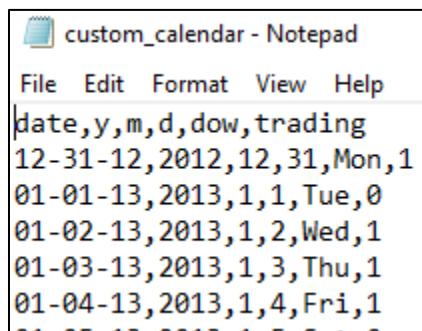
	A	B	C	D	E	F	G	H
1	Symbol	Date	Open	High	Low	Close	Adj Close	Volume
2	SP500TR	12-31-12	2462.71	2505.26	2455.12	2504.44	2504.44	0
3	SP500TR	01-02-13	2504.45	2568.55	2504.45	2568.55	2568.55	0
4	SP500TR	01-03-13	2568.55	2573.91	2556.48	2563.19	2563.19	0
5	SP500TR	01-04-13	2563.19	2578.25	2562.54	2575.66	2575.66	0
6	SP500TR	01-07-13	2575.66	2575.66	2558.43	2567.63	2567.63	0
7	SP500TR	01-08-13	2567.63	2567.63	2549.62	2560.17	2560.17	0
8	SP500TR	01-09-13	2560.18	2573.42	2560.18	2567.02	2567.02	0

Figure 4 - Column Analysis and Preparation of the SP500TR.csv in Excel

After ownership was assigned to postgres, the S&P 500 data was then imported into this table and checked via the top row data entries, as well as checked for minimum and maximum date, which returned the values 2012/12/31 and 2018/03/27 as expected.

3.1.3 Custom_calendar.

In order to link these two data tables with calendar dates and identify trading days, a third table called custom_calendar was created. The setup for this table was taken from an Excel file listing the columns date, year (y), month (m), day (d), day of the week (dow), and trading (0=FALSE, 1=TRUE). The trading data for each day was retrieved through the built-in Excel formula “NETWORKDAYS.INTL”. In order to capture all market holidays from 01/2013 to 03/2018, 64 dates were manually entered in a second tab, including nine respective holidays¹ per year for said time period. As this file was still an Excel Worksheet, the first tab was saved as csv, displayed in Figure 5. Yet again, ownership of the custom_calendar table in pgadmin was assigned to postgres and the data was imported from the modified custom_calendar.csv. Ultimately, the table showed the correct top row dates and trading days as well as a date range from 2012/12/31 and 2018/03/31.



```
custom_calendar - Notepad
File Edit Format View Help
date,y,m,d,dow,trading
12-31-12,2012,12,31,Mon,1
01-01-13,2013,1,1,Tue,0
01-02-13,2013,1,2,Wed,1
01-03-13,2013,1,3,Thu,1
01-04-13,2013,1,4,Fri,1
01-05-13,2013,1,5,Sat,0
```

Figure 5 - Column Analysis of the custom_calendar.csv in Notepad after Setup in Excel

¹ Namely: New Year Day, Martin Luther King Jr. Day, Presidents Day, Good Friday, Memorial Day, Independence Day, Labor Day, Thanksgiving Day, Christmas Day

In order to identify the number of trading days, two columns named end of month (eom) and previous trading day (prev_trading_day) were added and updated through several (nested) queries that can be found in the SQL appendix. Lastly, a role to access this database was created with the rolename “stockmarketreader” and the password “read123” and all read rights for existing as well as future tables were given.

3.2 R – Data Preparation

3.2.1 Connecting R to PostgreSQL.

After successful database setup, the packages “RPostgreSQL” and “DBI” were loaded to access the project database in R. The first query accessed the custom calendar with the date range of 2012/12/31 to 2018/03/27. Two other queries (qry1, qry2) were created for indices and quotes respectively but for the same date range. Their union was saved into the variable end of day (eod). The database was disconnected, and the successful import was tested with head(), tail(), and nrow() commands.

3.2.2 Checking for completeness & transforming tables for pivoting.

In order to clean the data, it was essential to only use symbols which fit the requirements of being at least 99% complete. For this purpose, the variable eod_complete was created. In the further transformations, the package “reshape2” provided the function dcast(), which helped us establish the pivoted end of day results (eod_pvt).

3.2.3 Using dates & merging with the custom calendar.

In order to only adopt trading days for our forecasting model, the variable tdays was created from where these would yield results as “TRUE” in the custom calendar. Subsequently, the pivoted end of day results were merged with the calendar dates and yielded a complete, pivoted end of day result (eod_pvt_compelte). In order to adopt a data structure for further

financial analysis, the first column identifiers were replaced by the respective dates and the standalone date column was removed.

3.2.4 Imputing missing data.

After this iterative process, data values could still be partially missing. For this purpose, the package “zoo” was called, and NA or NaN values were replaced with historical data, as long as this imputation would not exceed three consecutive rows.

3.2.5 Calculating returns & adjusting extreme values. Next, the returns were calculated with the help of the “PerformanceAnalytics” package. The CalculateReturns() function was applied to the complete, pivoted end of day data and further stored in a variable containing the end of day returns (eod_ret). Analysis of this outcome variable showed that some returns exceeded 100%, which seemed unlikely and was thus excluded from the following analytics. As this was the final step of the data extraction and transformation, we stored end of day returns (eod_ret) as a separate csv that could be loaded separately.

3.3 R – Analysis & Forecasting

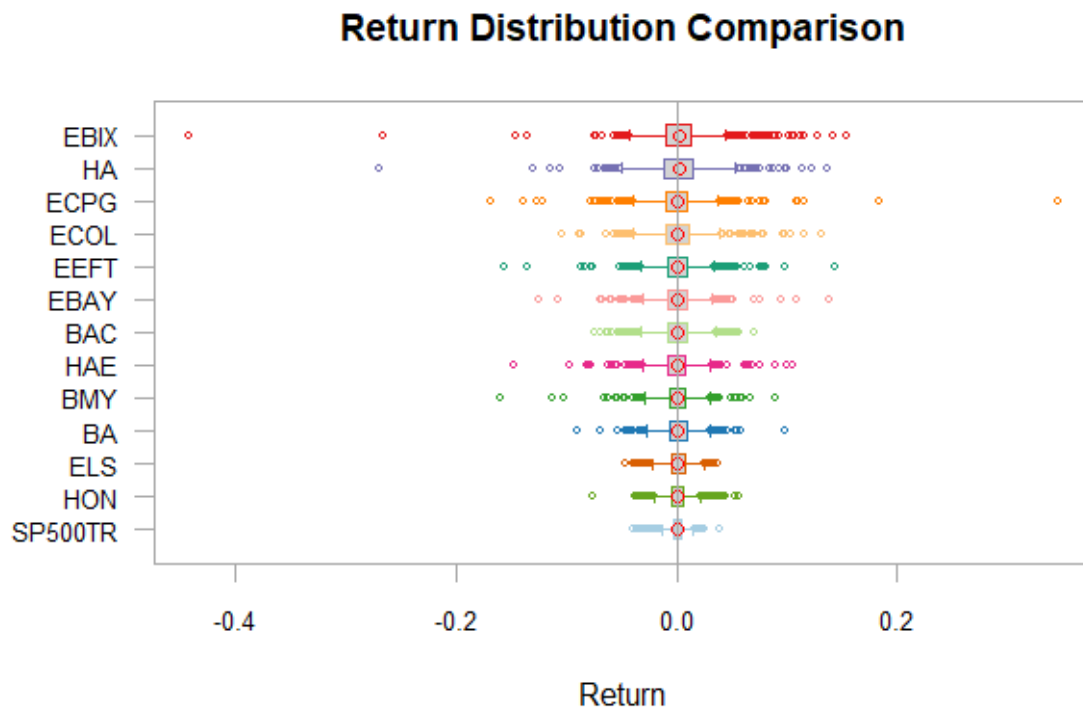
3.3.1 Tabular Return Data Analytics.

To conduct these analytics, we defined two extensible time series (.xts) in the variables Ra and Rb. Ra included the selected tickers mentioned above ('BA','BAC','BMY','EBAY','EBIX','ECOL','ECPG','EEFT','ELS','HA','HAE','HON'), while Rb contained the data from the S&P 500 benchmark. As our testing and training data was further divided into the time frames (1) 2013-2017 and (2) the first three months of 2018 respectively, the variables Ra2013_2017, Rb2013_2017, Ra2018, Rb2018, were created. To be more precise, Ra2013_2017 constitutes the training data for the selected tickers while Rb2013_2017 includes the training data for S&P 500.

Ra2018 then constitutes the testing data for the selected tickers while Rb2018 includes the testing data for S&P 500.

3.3.2 Graphical return data analytics & cumulative returns chart.

In order to visualize the returns of our cumulative returns for both the selected tickers as well as the benchmark, the function `Return.cumulative()` was applied to Ra2013_2017 and Rb2013_2017 respectively. For further information, the R code to retrieve the capital assets pricing model can be found in the appendix. As R's standard color choices did not clearly distinguish between the different tickers, the package "RColorBrewer" was loaded and the cumulative returns chart was depicted with the function `chart.CumReturns()`, binding the two training data frames. The palettes "Paired" and "Dark2" were concatenated for optimal illustration on the graphs. This graph is included in the following results section [see Q1]. In addition, the boxplot in Figure 6 lists the distribution comparison for all tickers and the benchmark sorted by variance. Figure 7 illustrates the chart drawdown of the ticker training data from 2013 to 2017.



*Figure 6 - Return Distribution Comparison for Selected Stock Tickers and the S&P 500 Benchmark
Sorted by Variance*

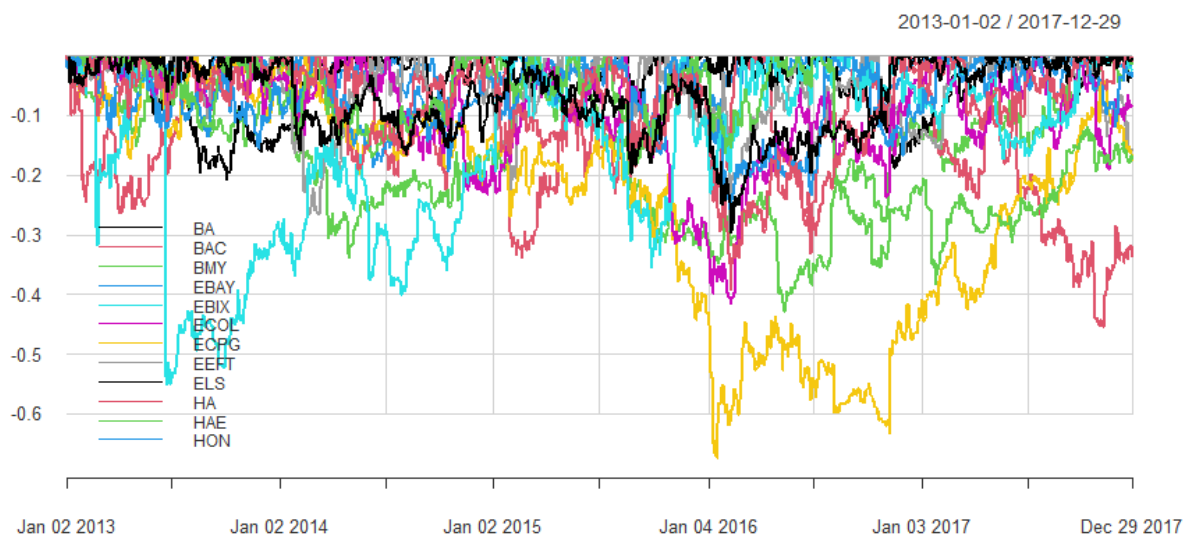


Figure 7 - Drawdown Chart Outlining the Negative Performance of Individual Stock Tickers

3.3.3 MV Portfolio Optimization.

Then, the function `table.AnnualizedReturns()` was applied to `Rb_training` (i.e. the S&P 500 annualized returns from 2013-2018) in order to set the daily minimum acceptable return as our benchmark. The packages “PortfolioAnalytics”, “ROI”, and “ROI.plugin.quadprog” were loaded in order to specify the desired portfolio type, objectives, and constraints in terms of risk, amount of investments, and the previously mentioned minimum margin. The function `optimize.portfolio()` was then applied to `Ra_training` (i.e. the training data 2013-2017 of our tickers) and optimized with the relevant objectives and restrictions, as well as the specified return on investment (ROI) method into the optimized portfolio variable (`opt_p`). The ideal weights were then extracted through `opt_p$weights` and saved as optimal weights (`opt_w`). The sum of these weights resulted in the value 1, supporting the correctness of our analysis. [see Q2]

3.3.4 Value & Hypothetical Portfolio Returns.

Finally, the retrieved weights were applied to our final portfolio (`Rp`) and the function of `table.AnnualizedReturns()` as well as `Return.cumulative()` were applied. The results of the final annualized and cumulative returns [see Q4] can be found in the next section alongside a chart that illustrates how our portfolio beats the market [see Q3].

4. RESULTS

4.1 Cumulative return chart for 2013-2017 for the selected stock tickers:

The following chart illustrates the cumulative return for our selected tickers ('BA','BAC','BMY','EBAY','EBIX','ECOL','ECPG','EEFT','ELS','HA','HAE','HON'). The color set is derived from the Package “RColorBrewer” to provide better visibility of each ticker. As can be seen in Figure 8 below, the S&P 500 benchmark provides a small but stable return. Most of the selected stock tickers share a similar trend with some more volatility and somewhat larger or smaller returns, apart from ‘HA’ reaching cumulative returns of up to 8% before dropping back down to around 5% at the end of the testing period.

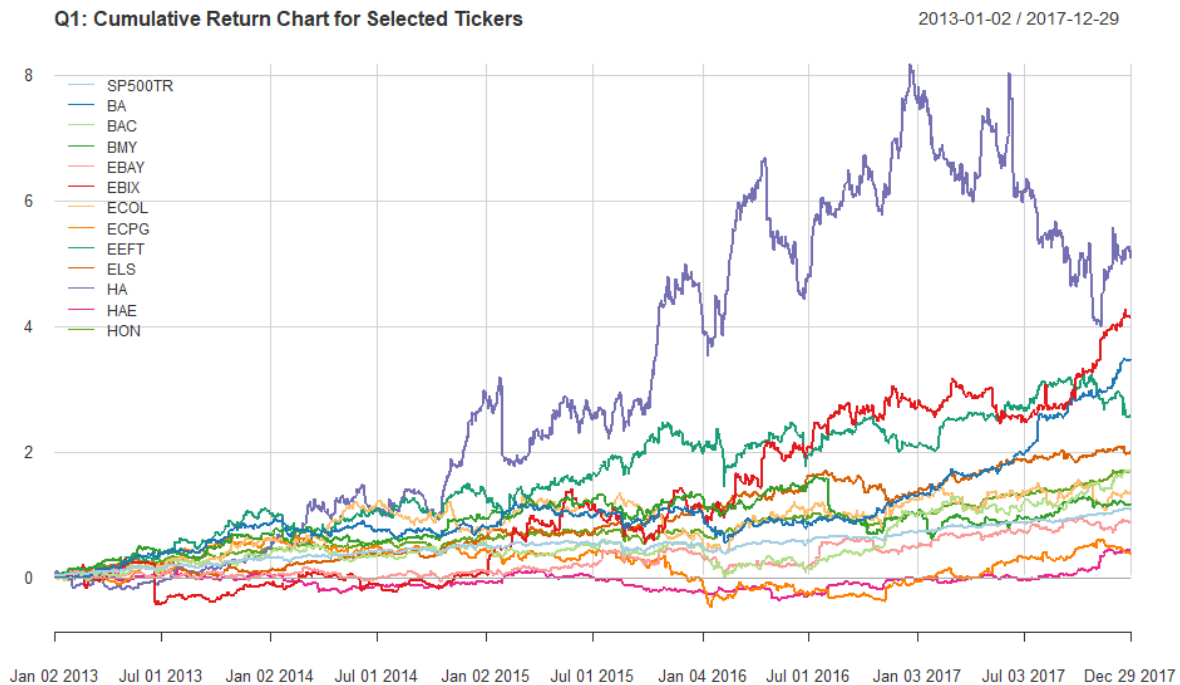


Figure 8 - Cumulative Return Chart for Selected Tickers

4.2 Weights of the optimized portfolio and the sum of these weights.

The sum of all optimized weights results in 1. This supports our goal of building a complete portfolio with full investment. The optimized weights for each stock can be seen in Figure 9. Furthermore, Figure 10 provides a ranked overview, i.e. sorted by optimized weights.

```
> opt_w
      BA      BAC      BMY      EBAY      EBIX      ECOL
-0.05701916  0.04892705  0.09956366  0.11306721 -0.05087794  0.03708897
      ECPG      EEFT      ELS      HA      HAE      HON
 0.04693971 -0.01656231  0.37975335 -0.06890893  0.17455194  0.29347646
> sum(opt_w)
[1] 1
```

Figure 9 - R Output for Optimized Portfolio Weights and Their Sum

Stock Ticker	Weight	Stock Ticker	Weight
1. ELS	0.37975335	7. ECPG	0.04693971
2. HON	0.29347646	8. ECOL	0.03708897
3. HAE	0.17455194	9. EEFT	-0.01656231
4. EBAY	0.11306721	10. EBIX	-0.05087794
5. BMY	0.09956366	11. BA	-0.05701916
6. BAC	0.04892705	12. HA	-0.068900893

Figure 10 - Summarized Optimized and Sorted Portfolio Weights

4.3 Cumulative return chart for the optimized portfolio and SP500TR index (2018):

The graph below illustrates the cumulative returns for both S&P 500 as well as our portfolio over the time period from 2nd of January to 27th of March 2018. While our portfolio does not perform as well until February, it exceeds the benchmark consistently after that point and ends quite a bit higher in March 2018. Yet, we warrant caution as this time period is rather short and might not be representative of a long-lasting trend.

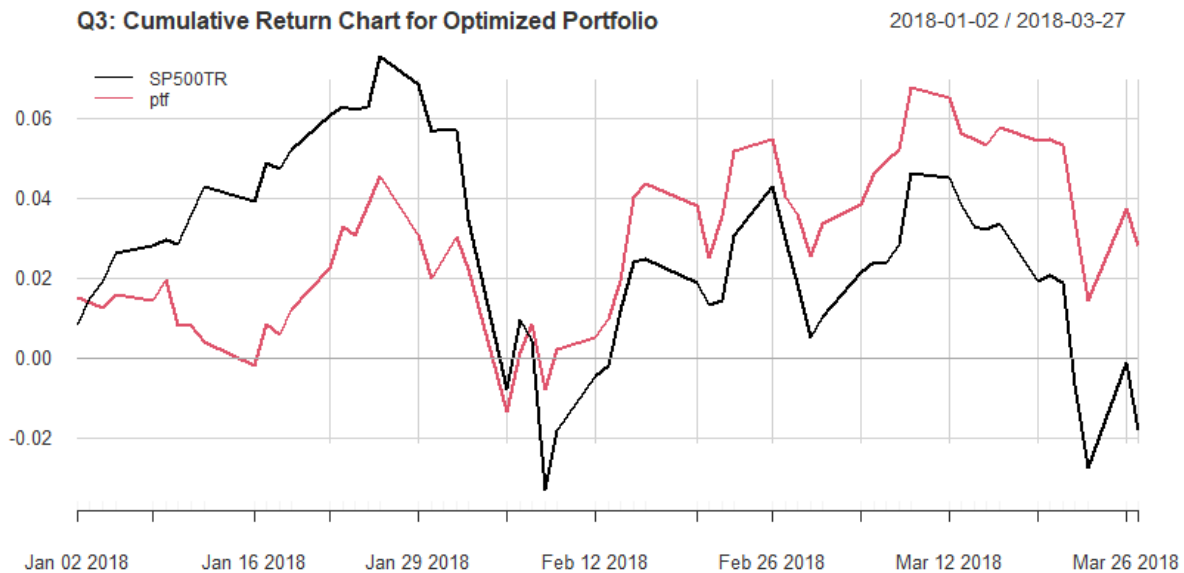


Figure 11 - Cumulative Return Chart for Optimized Portfolio versus S&P 500 Benchmark

The cumulative return for our optimized portfolio results in 0.02811848, which is a good result compared to the benchmark outcome of -0.01841827. The original R output can be seen in Figure 12.

```
> acc_Rp
      SP500TR      ptf
Cumulative Return -0.01841827 0.02811848
```

Figure 12 - R Output for Cumulative Return Results

4.4 Annualized returns for the optimized portfolio and SP500TR index (2018):

The annualized returns of S&P 500 are subpar, resulting in a value of -0.0763, while our portfolio provides an annualized return of 0.1257.

```
> table.AnnualizedReturns(Rp)
      SP500TR      ptf
Annualized Return    -0.0763 0.1257
Annualized Std Dev    0.1998 0.1657
Annualized Sharpe (Rf=0%) -0.3820 0.7586
```

Figure 13 – R Output for Annualized Daily Returns for Optimized Portfolio versus S&P 500 Benchmark

4.5 One paragraph comment on the performance of your portfolio against the index.

Fortunately, we beat the market – something some economists even consider impossible.

The forecast shows our portfolio's performance behind the market for a little over a month but by mid-February of 2018, the rate of return rose above the market and did not succumb to the Market since. While it is successful to create a portfolio that beats the market for any length of time, three months is a very short period to draw robust conclusions from. In practice, most traders, especially those who primarily use index and mutual funds, hold stocks much longer than three months. Sometimes, they even hold a stock for decades when it is involved with a retirement fund, for example. Yet, even outside of how one may handle their 401k, there are significant tax advantages to holding stocks for over a year. We conclude that for getting a great job at a major investment firm, we would need to create a forecast that would beat the market regularly and regarding the future. Then again, if we had achieved this goal in this project, we would likely not have to work at all.

5. CONCLUSION & FUTURE SCOPE

This project can be improved in several areas for the future. Currently, this project only focuses on the past values of the stock, that cannot be always the case in the real-time problem. There are numerous factors which affect the stock market. Sometimes these factors are unpredictable such as a sudden statement from the CEO of the company, natural or human-made disasters, which can affect the stock prices. These factors cannot be predicted but can be resolved before the market starts to react to the situation. One way to perform it is by using Natural Language Processing to analyze the current feedback and reaction on the affected stocks. NLP can provide real-time feedback on the shares. Several algorithms can be used to perform text classification such as Naïve Bayes, Random Forest, Convolutional Neural Networks and Linear support vector machines.

We suggest using Random Forest as it is efficient, flexible, and easy to implement. It provides the most accurate results for text classification. It builds multiple decision trees and merges them to deliver the results when compared with other algorithms. The factors which can be predicted, such as interest rates, foreign market, inflation, and deflation can be analyzed using machine learning to predict the outcomes. The machine learning model can study the relation of these factors with the current stock market to get more accurate results. The updated algorithm will be prepared to handle the other factors such as economically or non-economically, to provide a better prediction of stocks.

APPENDIX: FULL SQL / R CODE

```
/*
ISDS 570 Data Transformation for Business - Project SQL
*/

-----
----- Import Wiki -----
-----

-- Create table eod_quotes
-- NOTE: ticker and date will be the PK; volumes in double
precision (8 bytes) other numbers real (4 bytes)

CREATE TABLE public.eod_quotes
(
    ticker character varying(16) COLLATE pg_catalog."default"
NOT NULL,
    date date NOT NULL,
    open real,
    high real,
    low real,
    close real,
    volume double precision,
    "ex.dividend" real,
    split_ration real,
    adj_open real,
    adj_high real,
    adj_low real,
    adj_close real,
    adj_volume double precision,
    CONSTRAINT eod_quotes_pkey PRIMARY KEY (ticker, date)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.eod_quotes
    OWNER to postgres;

-- Import quandl_wiki.csv to the table - it will take some time
(almost 13 million rows)

-- Check!
SELECT * FROM eod_quotes LIMIT 10;
```



```
SELECT COUNT(*) FROM eod_quotes; -- this will take some time the
first time; should be 12,926,649
```

```
-----
-- Let us check the stock market data we have -----
-----
```

```
-- What is the date range?
SELECT min(date),max(date) FROM eod_quotes;
```

```
-- Really?
SELECT date_part('year',date), COUNT(*)/252 FROM eod_quotes
GROUP BY date_part('year',date);
```

```
-- Let's decide on a practical time range (e.g. 2013-2017)
SELECT MAX(date) FROM eod_quotes WHERE date BETWEEN '2013-01-01'
AND '2018-12-31';
SELECT ticker, date, adj_close FROM eod_quotes WHERE date
BETWEEN '2013-01-01' AND '2018-03-27';
```

```
-----
-----
-- We have stock quotes but we could also use daily index data -
-----
-----
```

```
-- Let's download 2013-2018 of SP500TR from Yahoo
https://finance.yahoo.com/quote/%5ESP500TR/history?p=^SP500TR
-- An analysis of the CSV indicated that a "symbol" column must
be added with the value SP500TR
-- Import the (modified) CSV to a (new) data table eod_indices
which reflects the original file's structure
```

```
/*
```

```
LIFELINE:
```

```
-- DROP TABLE public.eod_indices;
*/
```

```
CREATE TABLE public.eod_indices
(
    symbol character varying(16) COLLATE pg_catalog."default"
NOT NULL,
    date date NOT NULL,
    open real,
```

```

        high real,
        low real,
        close real,
        adj_close real,
        volume double precision,
        CONSTRAINT eod_indices_pkey PRIMARY KEY (symbol, date)
    )
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.eod_indices
    OWNER to postgres;

-- Check
SELECT * FROM eod_indices LIMIT 10;
SELECT min(date),max(date) FROM eod_indices;

-----
-- Next, let's prepare a custom calendar (using a spreadsheet) -
-----

-- We need a stock market calendar to check our data for
completeness
-- Because it is faster, we will use Excel (we need market
holidays to do that)
-- We will use NETWORKDAYS.INTL function
-- date, y,m,d,dow,trading (format date and dow!)
-- Save as custom_calendar.csv and import to a new table

/*
LIFELINE:
-- DROP TABLE public.custom_calendar;
*/

CREATE TABLE public.custom_calendar
(
    date date NOT NULL,
    y bigint,
    m bigint,
    d bigint,
    dow character varying(3) COLLATE pg_catalog."default",
    trading smallint,
    CONSTRAINT custom_calendar_pkey PRIMARY KEY (date)
)
WITH (

```

```

        OIDS = FALSE
    )
    TABLESPACE pg_default;

ALTER TABLE public.custom_calendar
    OWNER to postgres;

-- CHECK:
SELECT * FROM custom_calendar LIMIT 10;
SELECT min(date),max(date) FROM custom_calendar;

-- Let's add some columns to be used later: eom (end-of-month)
and prev_trading_day

/*
-- LIFELINE
*/

ALTER TABLE public.custom_calendar
    ADD COLUMN eom smallint;

ALTER TABLE public.custom_calendar
    ADD COLUMN prev_trading_day date;

-- CHECK:
SELECT * FROM custom_calendar LIMIT 10;

-- Populate these columns

-- Identify trading days
SELECT * FROM custom_calendar WHERE trading=1;
-- Identify previous trading days via a nested query
SELECT date, (SELECT MAX(CC.date) FROM custom_calendar CC WHERE
CC.trading=1 AND CC.date<custom_calendar.date) ptd
FROM custom_calendar;
-- Update the table with new data (this will take some time)
UPDATE custom_calendar
SET prev_trading_day = PTD.ptd
FROM (SELECT date, (SELECT MAX(CC.date) FROM custom_calendar CC
WHERE CC.trading=1 AND CC.date<custom_calendar.date) ptd FROM
custom_calendar) PTD
WHERE custom_calendar.date = PTD.date;
-- CHECK
SELECT * FROM custom_calendar ORDER BY date;

```

```

-- Update the table with new data (this will take some time)
UPDATE custom_calendar
SET eom = EOMI.endofm
FROM (SELECT CC.date,CASE WHEN EOM.y IS NULL THEN 0 ELSE 1 END
endofm FROM custom_calendar CC LEFT JOIN
(SELECT y,m,MAX(d) lastd FROM custom_calendar WHERE trading=1
GROUP by y,m) EOM
ON CC.y=EOM.y AND CC.m=EOM.m AND CC.d=EOM.lastd) EOMI
WHERE custom_calendar.date = EOMI.date;

UPDATE custom_calendar
SET prev_trading_day = '2012-12-28'
WHERE custom_calendar.date = '2012-12-31';

-- CHECK
SELECT * FROM custom_calendar ORDER BY date;
SELECT * FROM custom_calendar WHERE eom=1 ORDER BY date;
-- Identify trading days in 2017
SELECT COUNT(*) FROM custom_calendar WHERE trading=1 AND y=2018;

-----
-- Create a role for the database -----
-----
-- rolename: stockmarketreader
-- password: read123

/*
-- LIFELINE:
-- REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM
stockmarketreader;
-- DROP USER stockmarketreader;
*/

CREATE USER stockmarketreader WITH
    LOGIN
    NOSUPERUSER
    NOCREATEDB
    NOCREATEROLE
    INHERIT
    NOREPLICATION
    CONNECTION LIMIT -1
    PASSWORD 'read123';

-- Grant read rights (on existing tables and views)

```

```
GRANT SELECT ON ALL TABLES IN SCHEMA public TO
stockmarketreader;

-- Grant read rights (for future tables and views)
ALTER DEFAULT PRIVILEGES IN SCHEMA public
    GRANT SELECT ON TABLES TO stockmarketreader;
```

Group Project

```
# Displaying the format of quandl_wiki.csv data file (as it is >
1GB)
quandl = read.csv('/Users/elisa/Desktop/ISDS 570 - Business Data
Transformation/data/quandl_wiki.csv')
head(quandl)
summary(quandl)

# Stock Market Case in R
rm(list=ls(all=T)); gc() # this just removes everything from
memory

# Connect to PostgreSQL -----
-----

# Make sure you have created the reader role for our PostgreSQL
database
# and granted that role SELECT rights to all tables

require(RPostgreSQL)
require(DBI)
pg = dbDriver("PostgreSQL")
conn = dbConnect(drv=pg
                  ,user="stockmarketreader"
                  ,password="read123"
                  ,host="localhost"
                  ,port=5432
                  ,dbname="project"
)

#custom calendar
qry="SELECT * FROM custom_calendar WHERE date BETWEEN '2012-12-
31' AND '2018-03-27' ORDER by date"
ccal<-dbGetQuery(conn,qry)
#eod prices and indices
```

```

gry1="SELECT symbol,date,adj_close FROM eod_indices WHERE date
BETWEEN '2012-12-31' AND '2018-03-27'"
gry2="SELECT ticker,date,adj_close FROM eod_quotes WHERE date
BETWEEN '2012-12-31' AND '2018-03-27'"
eod<-dbGetQuery(conn,paste(gry1,'UNION',gry2))
dbDisconnect(conn)

#Explore
head(ccal)
tail(ccal)
nrow(ccal)

head(eod)
tail(eod)
nrow(eod)

head(eod[which(eod$symbol=='SP500TR'),])

# Use Calendar -----
-----

tdays<-ccal[which(ccal$trading==1),,drop=F]
head(tdays)
nrow(tdays)-1

# Completeness -----
-----
# Percentage of completeness
pct<-table(eod$symbol)/(nrow(tdays)-1)
selected_symbols_daily<-names(pct)[which(pct>=0.99)]
eod_complete<-eod[which(eod$symbol %in%
selected_symbols_daily),,drop=F]

#check
head(eod_complete)
tail(eod_complete)
nrow(eod_complete)

# Transform (Pivot) -----
-----

require(reshape2)
eod_pvt<-dcast(eod_complete, date ~
symbol,value.var='adj_close',fun.aggregate = mean, fill=NULL)
#check
eod_pvt[1:10,1:5] #first 10 rows and first 5 columns
ncol(eod_pvt) # column count

```

```

nrow(eod_pvt)

# Merge with Calendar -----
-----
eod_pvt_complete<-
merge.data.frame(x=tdays[, 'date', drop=F], y=eod_pvt, by='date', all
.x=T)

#check
eod_pvt_complete[1:10,1:5] #first 10 rows and first 5 columns
ncol(eod_pvt_complete)
nrow(eod_pvt_complete)

#use dates as row names and remove the date column
rownames(eod_pvt_complete)<-eod_pvt_complete$date
eod_pvt_complete$date<-NULL

#re-check
eod_pvt_complete[1:10,1:5] #first 10 rows and first 5 columns
ncol(eod_pvt_complete)
nrow(eod_pvt_complete)

# Missing Data Imputation -----
-----
# We can replace a few missing (NA or NaN) data items with
previous data
# (No more than 3 in a row.)
require(zoo)
eod_pvt_complete<-
na.locf(eod_pvt_complete, na.rm=F, fromLast=F, maxgap=3)
#re-check
eod_pvt_complete[1:10,1:5] #first 10 rows and first 5 columns
ncol(eod_pvt_complete)
nrow(eod_pvt_complete)

# Calculating Returns -----
-----
require(PerformanceAnalytics)
eod_ret<-CalculateReturns(eod_pvt_complete)

#check
eod_ret[1:10,1:5] #first 10 rows and first 4 columns
ncol(eod_ret)
nrow(eod_ret)

#remove the first row
eod_ret<-tail(eod_ret,-1) #use tail with a negative value

```

```

#check
eod_ret[1:10,1:4] #first 10 rows and first 4 columns
ncol(eod_ret)
nrow(eod_ret)

# Check for extreme returns -----
-----
# There is colSums, colMeans but no colMax so we need to create
it
colMax <- function(data) sapply(data, max, na.rm = TRUE)
# Apply it
max_daily_ret<-colMax(eod_ret)
max_daily_ret[1:10] #first 10 max returns
# And proceed just like we did with percentage (completeness)
selected_symbols_daily<-
names(max_daily_ret)[which(max_daily_ret<=1.00)]
length(selected_symbols_daily)

#subset eod_ret
eod_ret<-eod_ret[,which(colnames(eod_ret) %in%
selected_symbols_daily)]
#check
eod_ret[1:10,1:4] #first 10 rows and first 4 columns
ncol(eod_ret)
nrow(eod_ret)

# Export data from R to CSV -----
-----
write.csv(eod_ret,'/Users/elisa/Desktop/ISDS 570 - Business Data
Transformation/eod_ret_project.csv')

# Tabular Return Data Analytics -----
-----

# We will select 'SP500TR' and
c('BA','BAC','BMY','EBAY','EBIX','ECOL','ECPG','EEFT','ELS','HON
','HOS','HOV')
# We need to convert data frames to xts (extensible time series)
Ra<-
as.xts(eod_ret[,c('BA','BAC','BMY','EBAY','EBIX','ECOL','ECPG','
EEFT','ELS','HA','HAE','HON'),drop=F])
Rb<-as.xts(eod_ret[, 'SP500TR',drop=F]) #benchmark

Ra2013_2017 = Ra["2013/2017"]
Rb2013_2017 = Rb["2013/2017"]
Ra2018 = Ra["2018"]

```



```

Rb2018 = Rb["2018"]

head(Ra2013_2017)
head(Rb2013_2017)
tail(Ra2013_2017)
tail(Rb2013_2017)

# And now we can use the analytical package...

# Stats
table.Stats(Ra2013_2017)

# Distributions
table.Distributions(Ra2013_2017)

# Returns
table.AnnualizedReturns(cbind(Rb2013_2017,Ra2013_2017),scale=252
)

# Accumulate Returns
acc_Ra2013_2017<-Return.cumulative(Ra2013_2017)
acc_Ra2013_2017

acc_Rb2013_2017<-Return.cumulative(Rb2013_2017)
acc_Rb2013_2017

# Capital Assets Pricing Model
table.CAPM(Ra2013_2017,Rb2013_2017)

# Graphical Return Data Analytics -----
-----

# Cumulative Returns Chart (Q1)
library("RColorBrewer")

chart.CumReturns(Ra2013_2017,legend.loc = 'topleft', colorset =
brewer.pal(n = 12, name = "Paired"))
chart.CumReturns(Rb2013_2017,legend.loc = 'topleft')
chart.CumReturns(cbind(Rb2013_2017, Ra2013_2017),legend.loc =
'topleft', main = "Q1: Cumulative Return Chart for Selected
Tickers",
                    colorset = c(brewer.pal(name="Paired", n = 8),
brewer.pal(name="Dark2", n = 6)))

#Box plots

```

```

chart.Boxplot(cbind(Rb2013_2017,Ra2013_2017), sort.by =
"variance",
               colorset = c(brewer.pal(name="Paired", n = 8),
brewer.pal(name="Dark2", n = 6)))

#Drawdown
chart.Drawdown(Ra2013_2017,legend.loc = 'bottomleft')

# MV Portfolio Optimization -----
-----

# specificy entries in 2013 to 2017
Ra_training<-Ra["2013/2017"]
Rb_training<-Rb["2013/2017"]

head(Ra_training)
tail(Ra_training)

# use available 2018 data
Ra_testing<-Ra["2018"]
Rb_testing<-Rb["2018"]

head(Ra_testing)
tail(Ra_testing)

#optimize the MV (Markowitz 1950s) portfolio weights based on
training
table.AnnualizedReturns(Rb_training)
mar<-mean(Rb_training) #we need daily minimum acceptabe return

require(PortfolioAnalytics)
require(ROI) # make sure to install it
require(ROI.plugin.quadprog) # make sure to install it
pspec<-portfolio.spec(assets=colnames(Ra_training))
pspec<-add.objective(portfolio=pspec,type="risk",name='StdDev')
pspec<-add.constraint(portfolio=pspec,type="full_investment")
pspec<-
add.constraint(portfolio=pspec,type="return",return_target=mar)

#optimize portfolio
opt_p<-
optimize.portfolio(R=Ra_training,portfolio=pspec,optimize_method
= 'ROI')

#extract weights & check if sum = 1 (Q2)
opt_w<-opt_p$weights

```

```

opt_w
sum(opt_w)

#apply weights to test returns
Rp<-Rb_testing # easier to apply the existing structure
#define new column that is the dot product of the two vectors
Rp$ptf<-Ra_testing %*% opt_w

#check
head(Rp)
tail(Rp)

# Annualized Returns for portfolio and SP500TR index for
available 2018 (Q4)
table.AnnualizedReturns(Rp)

# Value & Chart Hypothetical Portfolio Returns (Q3) -----
-----
acc_Rp = Return.cumulative(Rp)
acc_Rp

#chart.CumReturns(cbind(Rb, Ra),legend.loc = 'topleft')
chart.CumReturns(Rp,legend.loc = 'topleft', main = "Q3:
Cumulative Return Chart for Optimized Portfolio")

# End of Group Project

```

REFERENCES (ENDNOTES)

ⁱ Statman, M., 2019. *Behavioral Finance: Past Battles And Future Engagements*. [online] Taylor & Francis. Available at: <<https://www.tandfonline.com/doi/abs/10.2469/faj.v55.n6.2311>> [Accessed 10 May 2020].

ⁱⁱ Investopedia. 2020. *Portfolio*. [online] Available at: <<https://www.investopedia.com/terms/p/portfolio.asp>> [Accessed 10 May 2020].

ⁱⁱⁱ Finance.yahoo.com. 2020. *Yahoo Is Now A Part Of Verizon Media*. [online] Available at: <<https://finance.yahoo.com/quote/%5ESP500TR/history?p=%5ESP500TR>> [Accessed 10 May 2020].

Kalczynski, P. J., 2020. *SHARED Dropbox Folder. ISDS 570 – Business Data Transformation* [online] Available at: https://www.dropbox.com/sh/t67rvul12z1hwok/AABx7NiUgOE0_Br1e-2pDPmla?dl=0. [Accessed 10 May 2020].