Griffin Arnone
Anhua Cheng
Bailey Scoville
MSDS 460
February 25, 2024

Homework Assignment 4: Monte Carlo Methods---Performance Benchmark

Abstract

R and Python, two popular programming languages in the financial sector, rival each other in performance and accessibility. This benchmark test attempts to compare the performance of R and Python in conducting Monte Carlo simulations of the performance of four mutual funds over one year. The findings of this study can be applied to discussions of the value and effectiveness of investment fund managers. Quantitative metrics for the benchmark test include execution time, measured in seconds, and memory usage, measured in megabytes. Qualitative metrics include ease of code and code efficiency. Based on these metrics, few significant differences were observed in the performance of Python and R in this simulation.

Introduction

R and Python are two of the most commonly used programming languages given their ease of access and availability of a vast amount of libraries. Verma (2023) and Sharma (2023) both listed R and Python as the top programming languages for the financial sector. Being proficient in one of these two programming languages is often required when applying for a data scientist position in the finance industry. As such, our team embarked on the journey to compare the performance of these two programming languages while implementing Monte Carlo simulations over financial data. The underlying study we took was to understand if active investment managers outperform the passive index in general and if it is justified to pay higher management fees to invest in active mutual funds.

We selected three of the highest-rated, actively managed mutual funds and one passive Exchanged-Traded Fund (ETF) and simulated 10,000 different daily price paths for each of the funds. With the simulated data, we computed the average expected returns and volatility to compare the results. Given the amount of data involved and the large number of simulation trials, we believe we can effectively compare the performance of R and Python.

Literature Review

Many scholarly articles compare Python and R's performance in a variety of contexts. Gallagher and Trendafilov (2018) compared the normalization of Python and R in the business sector due to the increasing demand for programming skills among recent business school graduates. Though this study did not compare the languages' simulation capabilities, the authors note that both Python and R accurately performed complex statistical calculations. The authors argued any slight differences in language performance observed in their study could be mitigated by the programming skill of the data professional (Gallagher and Trendafilov 2018).

Brittain et al. (2018) evaluated the performance of Python, R, and SAS, emphasizing the critical role of tool selection for data scientists. After comparing the three languages on a variety of metrics, the authors concluded Python and R outperformed each other situationally, but neither language was declared more capable overall (Brittain et al. 2018).

Methods

The fund data was sourced from Morningstar's list of large blend funds. To select comparable portfolios, we looked up the highest-rated mutual funds by Morningstar that invest mainly in large capitalized U.S. companies. Appendix A includes the search results. Among the list, we selected three actively managed mutual funds, American Funds Washington Mutual (RWMGX), Parnassus Core Equity (PRILX), T.Rowe Price U.S. Equity Research (PCCOX), and

one passive ETF (WFSPX). Once we identified the mutual funds, we downloaded the historical fund closing prices since fund inception from Yahoo Finance.

With downloaded historical fund closing prices, we conducted the simulation in a series of steps in Python and R. First, we calculated the historical daily returns and reviewed the distribution of the logarithm of the daily returns. The logarithmic distribution is close to normal distribution for all four funds and distribution plots are included in Appendix B. With that information and the standard deviation of the logarithm of the daily returns, we simulated 10,000 different daily price paths for a year, assuming normal distribution of the variables. To simulate a year of fund performance, the code calculated stock prices over 251 days, the average number of trading days in one year. For each of the simulations, we then projected the next year's price paths based on the fund closing price as of Feb 16, 2024. We created a dataframe for each of the four selected funds which contained 10,000 simulations of next year's potential price fluctuations. We then computed the average expected returns for each of the four funds and associated volatility.

The simulation code was created using Jupyter Notebook for Python and RStudio for R. The visualizations were created using Python. The group collaborated on the code but ran the simulation on one member's MacbookPro. Various stages of the simulation process included lines of code to measure the quantitative metrics of execution time and memory usage. The qualitative metrics of ease of code and code efficiency were determined based on the judgment of the team.

Results

The average annual returns calculated for each fund were similar in Python and R and are detailed in Appendix C. The fund with the highest annual return was the American Funds

Washington Mutual (RWMGX), which is an actively managed fund. The average annual returns were 13.55% and 13.05% in the Python and R simulations, respectively. The average annual returns for the ETF (WFSPX) were 11.67% and 11.20% in the Python and R simulations, respectively, which implies that actively managed mutual funds outperform the passive index fund based on the scope of our analysis. Variability plots for the Python simulation are included in Appendix D, and simulated daily return distributions are included in Appendix E.

Overall, R demonstrated more consistency in memory usage and Python recorded faster execution times across a variety of data inspection, statistical calculations, and simulation tasks. The task with the longest execution time was exporting all four large price path dataframes, each with 2,510,000 observations, to CSV files. This final step took approximately 5 seconds in R and Python. A full list of execution times and memory usage is included in Appendix F.

Both Python and R efficiently executed the stock price simulation with relatively similar lines of code. While the team observed more ease of coding in Python, it should be noted the team has more experience with that programming language.

Conclusions

While we noticed differences in memory usage and execution time between R and Python, the differences observed in this benchmark study are minor. However, the differences in these metrics may become more consequential with a larger simulation. Based on our study, both programming languages are capable tools for performing simulation analysis.

If our team conducted the study again, we would generate simulations over longer periods, such as 5, 10, and 20 years, to determine how the performance differences scale. We might also work with variables of different distributions to understand how different variables impact execution time and memory usage in R and Python.

References

Brittain, Jim, Mariana Cendon, Jennifer Nizzi, and John Pleis. 2018. "Data Scientist's Analysis Toolbox: Comparison of Python, R, and SAS Performance." *SMU Data Science Review* 2 (2): https://scholar.smu.edu/datasciencereview/vol1/iss2/7/.

Gallagher, Michael, and Rossen Trendafilov. 2018. "R Vs. Python: Ease of Use and Numerical Accuracy." *Journal of Business and Accounting* 11, no. 1 (October): 117-126.

Melul, Elias. 2020. "Monte Carlo Simulations for Stock Price Predictions [Python]." Medium. https://medium.com/analytics-vidhya/monte-carlo-simulations-for-predicting-stock-prices-python-a64f53585662.

Morningstar. n.d. "Large Blend Funds." Morningstar. Accessed February 19, 2024. https://www.morningstar.com/large-blend-funds.

Sharma, Gaurav. 2023. "Best Programming Languages for Finance & Fintech in 2024." Bankers by Day. https://www.bankersbyday.com/programming-languages-banking-finance-fintech/.

Verma, Rakesh. 2023. "Top Programming Languages for Finance and FinTech: Which One Should You Choose?" LinkedIn. https://www.linkedin.com/pulse/top-programming-languages-finance-fintech-which-one-should-verma.

Yahoo! Finance. n.d. "Trending Tickers." Yahoo! Finance. Accessed February 18, 2024. https://finance.yahoo.com/lookup.

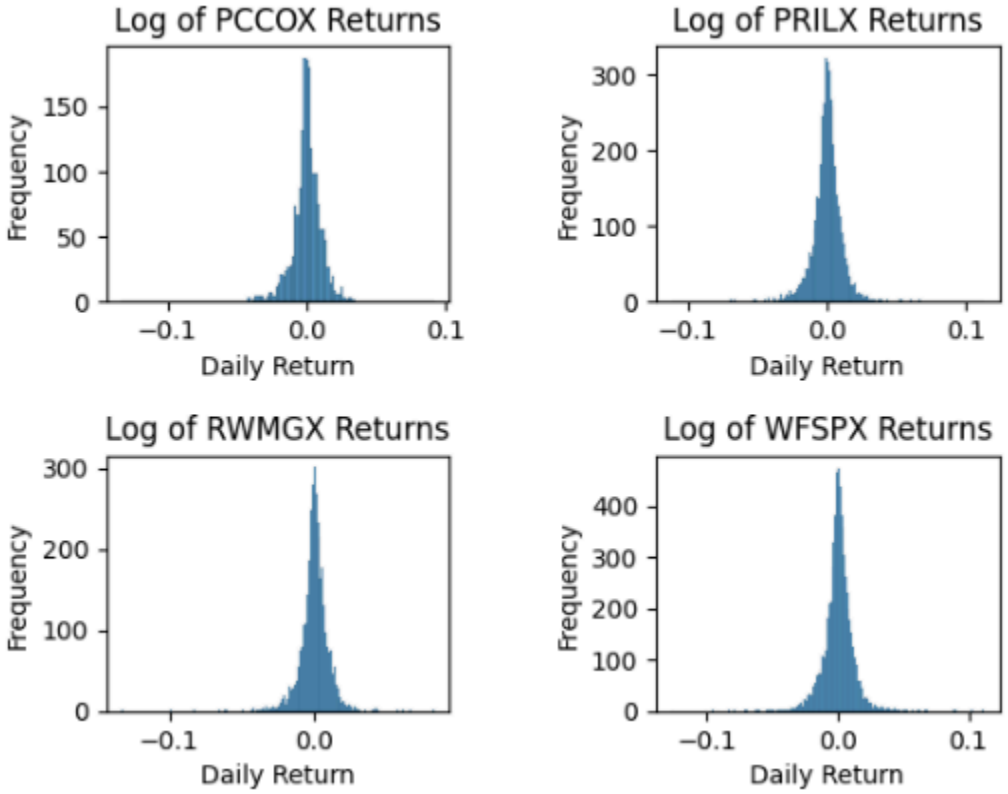
Appendix A

Morningstar Screener Summary: US Large Blend

Ticker	Name	Medalist Rating - Overall	Morningstar Category	Fund Size
FWMMX	American Funds Washington Mutual 529-F-2	Gold	Large Blend	173,272,637,071.00
FWWMX	American Funds Washington Mutual 529-F-3	Gold	Large Blend	173,272,637,071.00
FXAIX	Fidelity 500 Index	Gold	Large Blend	484,415,114,433.00
GQLIX	GMO Quality I	Gold	Large Blend	9,392,746,376.00
GQETX	GMO Quality III	Gold	Large Blend	9,392,746,376.00
GQEFX	GMO Quality IV	Gold	Large Blend	9,392,746,376.00
GQESX	GMO Quality R6	Gold	Large Blend	9,392,746,376.00
GQLOX	GMO Quality VI	Gold	Large Blend	9,392,746,376.00
GQEIX	GQG Partners US Select Quality Eq Instl	Gold	Large Blend	2,022,082,790.00
GQEPX	GQG Partners US Select Quality Eq Inv	Gold	Large Blend	2,022,082,790.00
GQERX	GQG Partners US Select Quality Eq R6	Gold	Large Blend	2,022,082,790.00
NWAVX	Nationwide GQG US Quality Eq R6	Gold	Large Blend	111,066,069.00
PRILX	Parnassus Core Equity Institutional	Gold	Large Blend	28,731,237,712.00
PRBLX	Parnassus Core Equity Investor	Gold	Large Blend	28,731,237,712.00
PBFDX	Payson Total Return	Gold	Large Blend	256,371,002.00
SWPPX	SchwabÂ® S&P 500 Index	Gold	Large Blend	85,924,055,963.00
PRCOX	T. Rowe Price U.S. Equity Research	Gold	Large Blend	11,074,174,440.00
PACOX	T. Rowe Price U.S. Equity Research Adv	Gold	Large Blend	11,074,174,440.00
PCCOX	T. Rowe Price U.S. Equity Research I	Gold	Large Blend	11,074,174,440.00
VFIAX	Vanguard 500 Index Admiral	Gold	Large Blend	398,447,915,120.00
VINIX	Vanguard Institutional Index I	Gold	Large Blend	269,620,835,548.00
VIIIX	Vanguard Institutional Index Instl Pl	Gold	Large Blend	269,620,835,548.00
WFSPX	iShares S&P 500 Index K	Gold	Large Blend	37,834,588,937.00

Appendix B

Distribution of Logarithm of the Historical Daily Returns



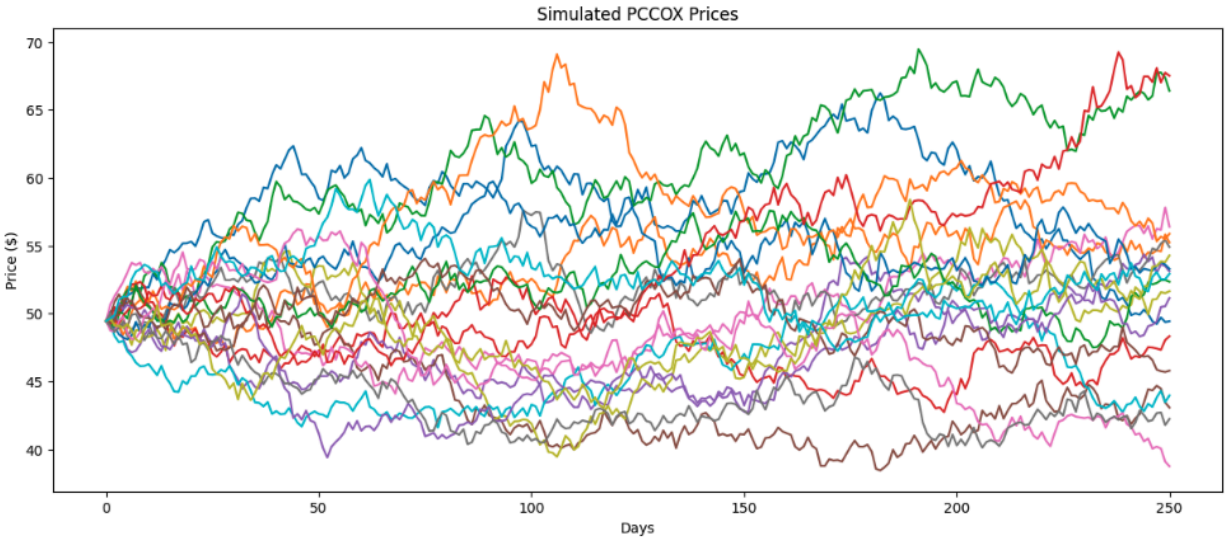
Appendix C

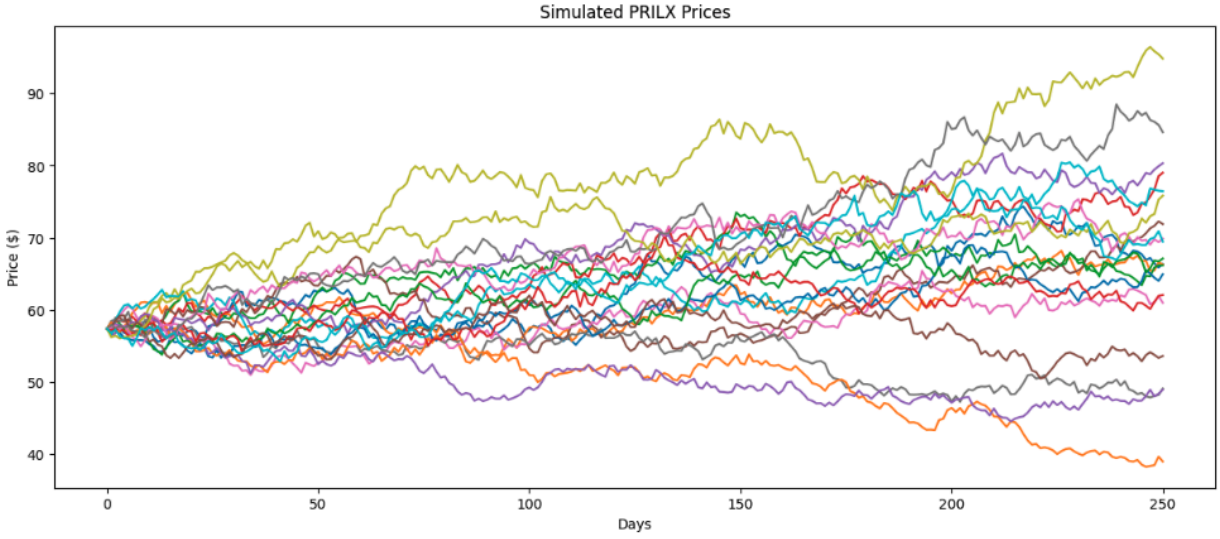
Comparison of Simulated Average Annual Returns between Python and R

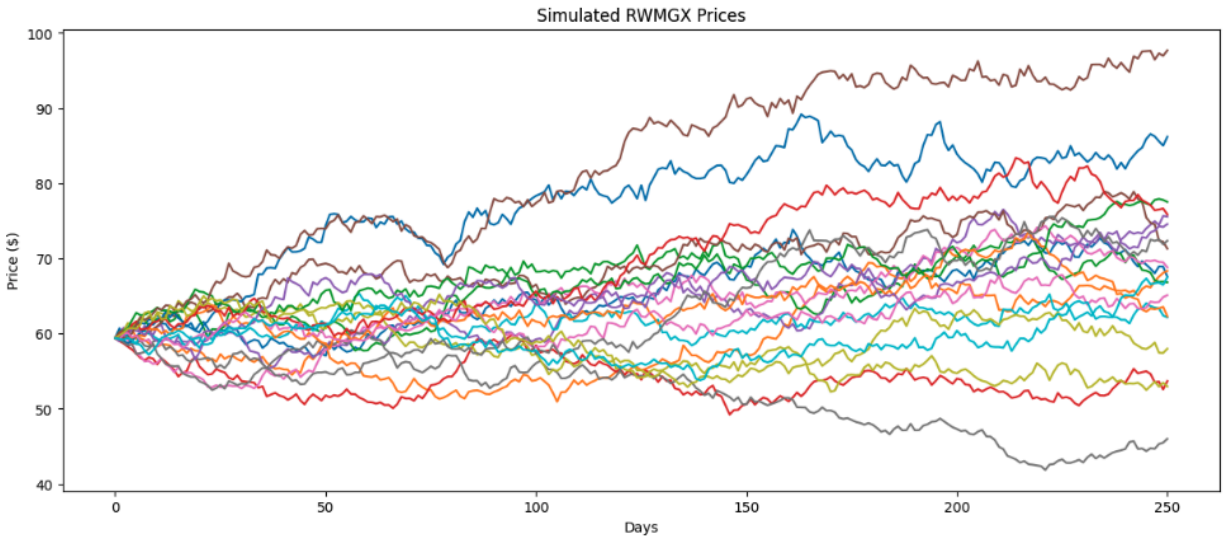
Fund Name	Ticker	Average Annual Return in Python	Average Annual Return in R
T.Rowe Price U.S. Equity Research	PCCOX	13.29%	12.71%
Parnassus Core Equity	PRILX	10.55%	10.12%
American Funds Washington Mutual	RWMGX	13.55%	13.05%
iShare S&P 500 Index	WFSPX	11.67%	11.20%

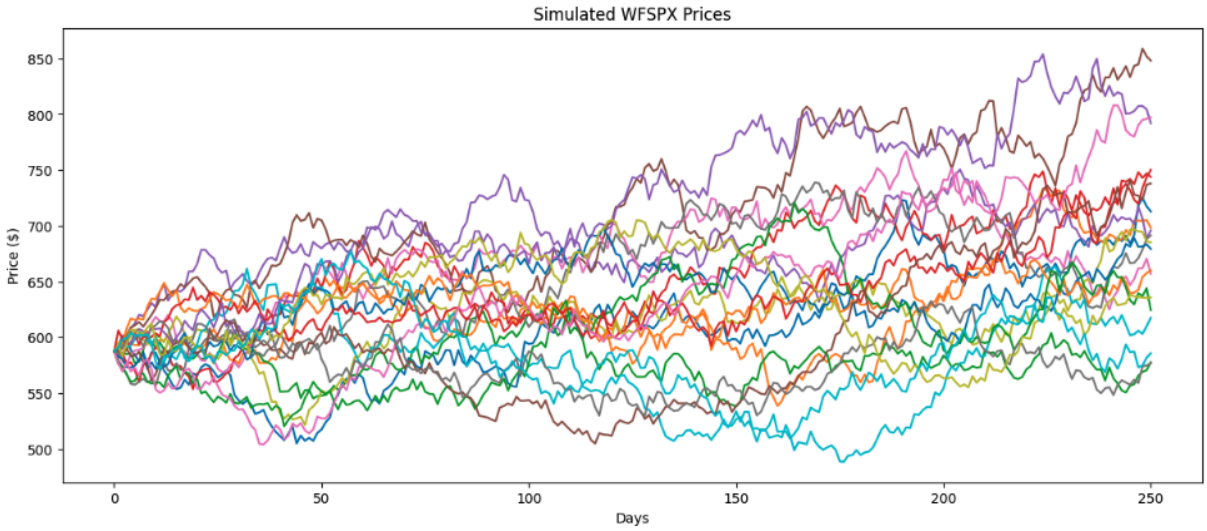
Appendix D

Python Fund Simulation Plots - Price Paths 1-20



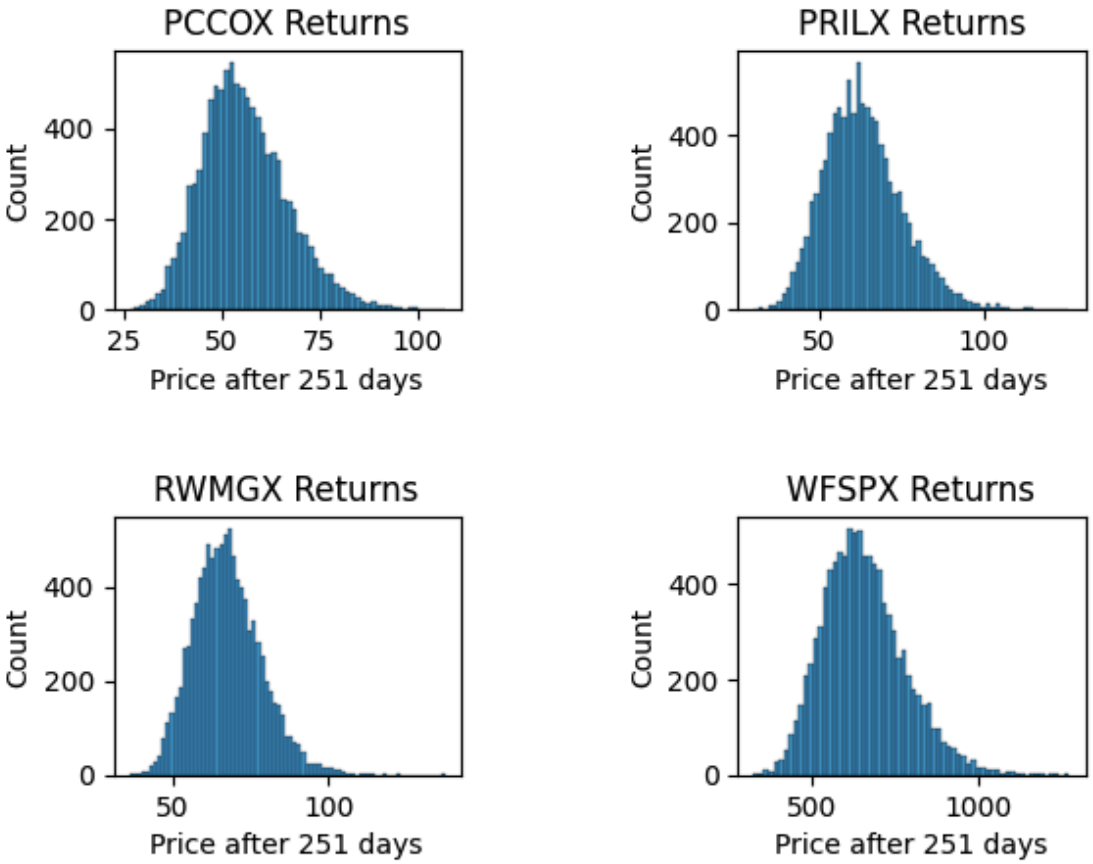



Simulated RWMGX Prices
Price ($)
Days
100
90
80
70
60
50
40
0
50
100
150
200
250


Simulated WFSPX Prices
Price ($)
Days
850
800
750
700
650
600
550
500
0
50
100
150
200
250

Appendix E

Python Fund Simulation Return Distributions



Appendix F

Performance Comparison Between Python and R

	Execution Time (seconds)	Memory Usage (MB)
Load & Inspect Data		
R	0.1739468575	409.0445328
Python	0.05571985245	201.046875
Fund Log Transformation		
R	0.02582097054	409.0982437
Python	0.005049705505	201.21875
Fund Transformation Plots		
R	0.03252100945	409.7133942
Python	0.7213199139	214.15625
Compute Drift & Variance		
R	0.02247190475	409.3455887
Python	0.002657175064	214.375
Compute Variance & Daily Returns		
R	0.3507380486	409.3457947
Python	0.3926398754	499.3125
Simulate Price Path for Trials		
R	0.2052550316	409.3777313
Python	0.02635383606	474.515625
Price Paths to Dataframes		
R	0.2396790981	408.7008591
Python	0.009378910065	459.8125
Calculate Return & Volatility		
R	0.5472159386	409.8740997
Python	0.05669927597	410.640625
Calculate Average Return & Create Table		
R	0.02333402634	409.8772125
Python	0.005060195923	411.3125
Export Dataframes to CSV		
R	4.930419922	409.9670715
Python	5.008857965	488.421875
Bring in data		
R	0.1739468575	409.0445328
Python	0.05571985245	201.046875
Export data		
R	4.930419922	409.9670715
Python	5.008857965	488.421875
Benchmark study implementation		
R	1.588461876	3274.664063
Python	0.5535588264	2872.234375
Graph Codes		
R	0.03252100945	409.7133942
Python	0.7213199139	214.15625