Risk Analysis

This report showcases that a fictional investment team has the skills necessary to beat the 60:40 benchmark through tactical asset allocation (TAA). TAA is an investment strategy that involves actively adjusting the weighting of a portfolio, such as re-balancing holdings, in order to benefit from changing market conditions. In this scenario we assume that this team is able to implement a tactical asset allocation that achieves an average skill of 0.55. Through calculating the confidence interval (via moving block bootstrapping) for the average skill required to produce an annualised out-performance of the benchmark exceeding 1%, we can demonstrate this team's average skill falls within this this region highlighting the capabilities of this team to beat the benchmark. Additionally, we can use this average skill of 0.55 to find the average annualised outperformance we could have achieved over the last 20 years, with varying asset weighting strategies. Moreover, using Monte Carlo simulations we can show how our skill of 0.55 could fair over the predicted future return paths.

Question 1

This report initially assumes that our tactical asset allocation is bound by choosing a weighting of equity and fixed income each month of either 70:30 or 50:50. Through this TAA it is found that the average skill required to achieve an annualised outperformance greater than 1% using the original data set is 0.53 and using a bootstrapped method it is 0.549 with a 95% confidence interval of 0.515 to 0.581. This annual outperformance is defined as the excess return obtained over and above the 60:40 benchmark. The 60:40 strategy is a well renowned benchmark within investing as it allows investors to utilise a level of diversification that provides "equity-like returns with bond-like volatility" (quantstart 2022), which can be shown from our data in the following figure:

Benchmark annualised returns	0.0778
Equity annualised returns	0.0989
Fixed Income annualised returns	0.0349
Benchmark annualised volatility	0.1281
Equity annualised volatility	0.2001
Fixed Income annualised volatility	0.0689

Therefore, we have created a function that will determine the skill level required to obtain the necessary annualised outperformance. But before we go into detail about the function that achieves this we want to know if the data provided is simple returns or log returns as this would impact the equation used to calculate annualised performance. To accomplish this, we test the properties of the returns data. This is carried out by creating three returns datasets:

1) the original (returns_data)
2) a log returns dataset under the assumption the original returns are simple returns (log_transformed_data)
3) a simple returns dataset assuming the original returns are log returns (log_reversed_data)

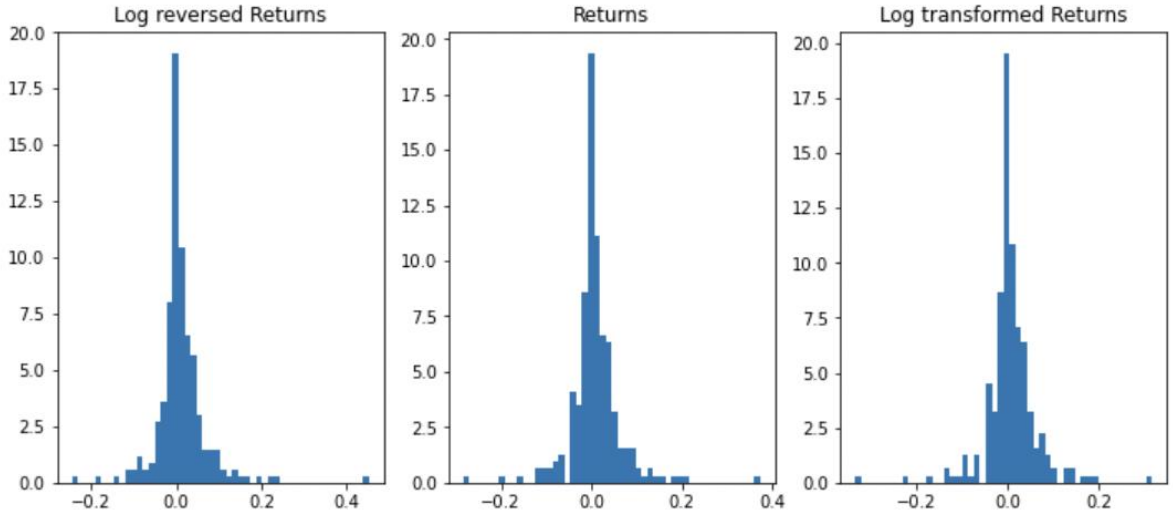
This is achieved through the following transformations:

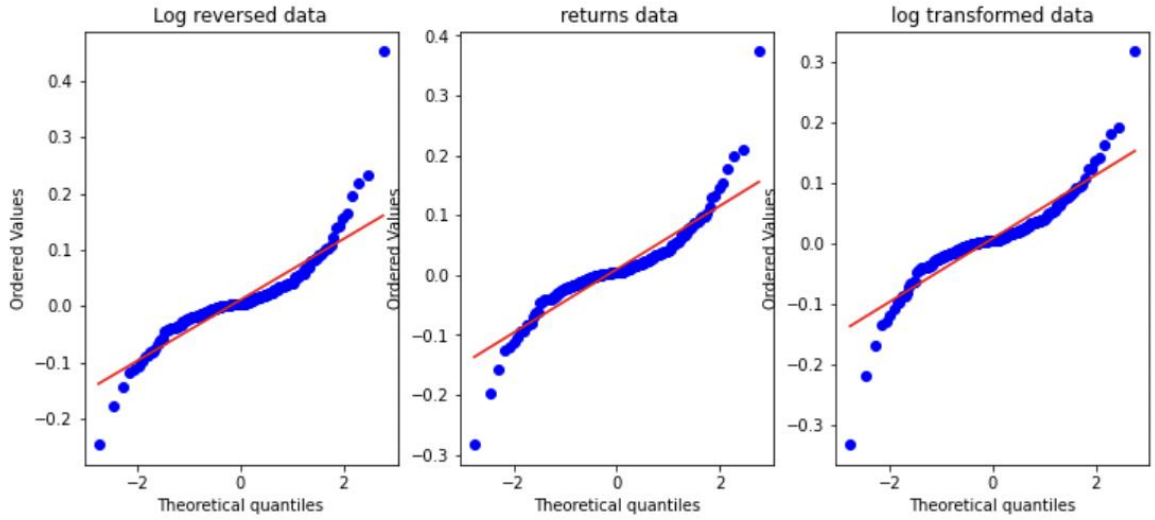
```
# First we create these three return data sets:


log_reversed_data = np.exp(returns_data) - 1
returns_data
log_transformed_returns = np.log(returns_data +1)
```

The hypothesis here is if the original dataset consists of simple returns, we should observe that it exhibits properties of simple returns, while the log-transformed dataset exhibits properties of log returns. Similarly, if the original dataset consists of log returns, we expect to observe properties of log returns in the original dataset, while the log-reversed dataset exhibits properties of simple returns. By comparing these different datasets, we should gain insights into the properties of the original dataset.

Firstly, we create histograms of each dataset as simple returns will likely form a skewed distribution with higher variance and thus more extreme values hence will not necessarily follow a normal distribution. Alternatively, log normal returns should follow a normal distribution. This results from the central limit theorem for sums, which states taking sums of many large samples will tend towards a normal distribution. Therefore, as log returns are additive over time it is equivalent to taking the many sums of independent variables and thus will converge to a normal distribution. Normality is further tested through a QQ plot and calculating each distribution's moments. Moreover, we use an ADF to test for the stationarity of the distributions. We would expect that the log returns would show greater signs of stationarity in comparison to the simple returns. The results of these tests are shown below:






```
log reversed data:
1st Moment, Mean: 0.011277963177313497
2nd Moment, Variance: 0.0036076477839145265
3rd Moment, Skewness: 1.7330139355701897
4th Moment, Kurtosis: 14.449795466060232
-----------------------------------
returns data:
1st Moment, Mean: 0.009527083333333334
2nd Moment, Variance: 0.0033228301414930553
3rd Moment, Skewness: 0.6801179596613416
4th Moment, Kurtosis: 10.320265597991279
-----------------------------------
log transformed data:
1st Moment, Mean: 0.007860595632165762
2nd Moment, Variance: 0.0032533323582690944
3rd Moment, Skewness: -0.3099036422780205
4th Moment, Kurtosis: 9.984359807889017


log reversed data:
ADF Statistic: -6.799957713941085
p-value: 2.2502357085097493e-09
Critical Values:
        1%: -3.459
        5%: -2.874
        10%: -2.573
returns data:
ADF Statistic: -16.536304472923145
p-value: 2.0024417545906982e-29
Critical Values:
        1%: -3.458
        5%: -2.874
        10%: -2.573
log transformed data:
ADF Statistic: -16.638246896227173
p-value: 1.6432485466614778e-29
Critical Values:
        1%: -3.458
        5%: -2.874
        10%: -2.573
```

However, as shown in the figures none of these tests provided evidence of the expected properties of the distributions. This is likely due to an insufficient number of observations; therefore, we are not able to conclude with any certainty if the returns are log returns or simple returns without knowledge of the price dataset from which they originated. In this case we will assume that we have simple returns as this is more intuitive and the tests lean towards the properties of simple returns.

We will now comeback to the function that finds the average skill required to create an annualised outperformance of the benchmark by >1%. This is achieved through the "avg_skill_for_outperformance" function in the attached code. This function randomly chooses a weighting each month for the full 240 months in the dataset resulting in a sequence of which the skill and annual outperformance are calculated. The calculation of the annual outperformance is where the importance of the previous assumption lies. This is because simple returns are asset additive which means that we can multiply our weights across Equity and Fixed Income to find the weighted average return for our portfolio each month, which would not be possible with log returns. However simple returns are not additive across time, unlike log returns, instead they are multiplicative. Hence, to calculate the performance of our portfolio we use the following formula:

$$R = \prod_{m=1}^{n}(1 + r_i) - 1$$

R= total returns
r_i = each months return

This creates a good model for finding the return of a monthly rebalanced portfolio as each month only accounts for the weighted average returns and hence this formula will result in the portfolios total performance. To find the annualised performance we then implement the following formula:

$$\bar{R} = [\prod_{i=1}^{n}(1 + r_i)]^{\frac{k}{n}} - 1$$

$\bar{R}$ = annualised return
k = number of periods in a year
n = number of periods

This simulation is continuously run generating numerous sequences with corresponding annualised outperformances and skills. In each case where a sequence achieves an annualised outperformance greater than 0.01, specifically greater than or equal to 0.01 to capture the instances of the smallest increment above 0.01, the associated skill is appended to a list. These simulations will continuously be repeated till a desired number of skills have been appended to the list, which in our case is set at 10. Taking the average of this list will provide a point estimate of the required skill from this dataset.

However, confidence intervals provide a more dependable statistic than point estimates to base the required skill level off. Therefore, to obtain a confidence interval, we will need to use a sufficient number of samples to provide a large number of means of required skills. This is because as the number of average skills increases the distribution of these means will tend to a normal distribution as supported by the central limit theorem. This theorem states that if you have a population with mean μ and standard deviation σ and take sufficiently large random samples from the population with replacement then the distribution of these

sample means will be approximately normally distributed (LaMorte, W 2016). Thus, this allows us to calculate standard errors which can be used to construct confidence intervals.

A sufficiently large number of samples can be achieved through bootstrapping with replacement. Bootstrapping with replacement is a useful technique to generate such samples from an existing dataset. This is achieved by taking random samples from the original dataset with replacement, meaning each observation has an equal chance of being selected even if it has been selected before. However, when dealing with time series data, serial correlation can cause problems as standard bootstrapping becomes impaired as it assumes variables are independently and identically distributed, which is violated by the autocorrelation. Jegadeesh (1990) and Campbell, Lo and MacKinlay (1996) have found that monthly equity returns are significantly autocorrelated. Although there may not be continuous autocorrelation throughout the data, there will likely be pockets of serial correlation, such as during bull and bear markets. We confirm this in our data through checking for autocorrelation via an ACF test for a rolling window of 60 months with 20 lags. The resulting figures show in many cases autocorrelation is present.

Therefore, to address this issue, this report uses an augmented bootstrapping technique called moving block bootstrapping (MBB) as standard bootstrapping is no longer adequate. MBB overcomes this limitation by resampling blocks, of a constant size, of observations instead of individual observations preserving the temporal dependence in the data.

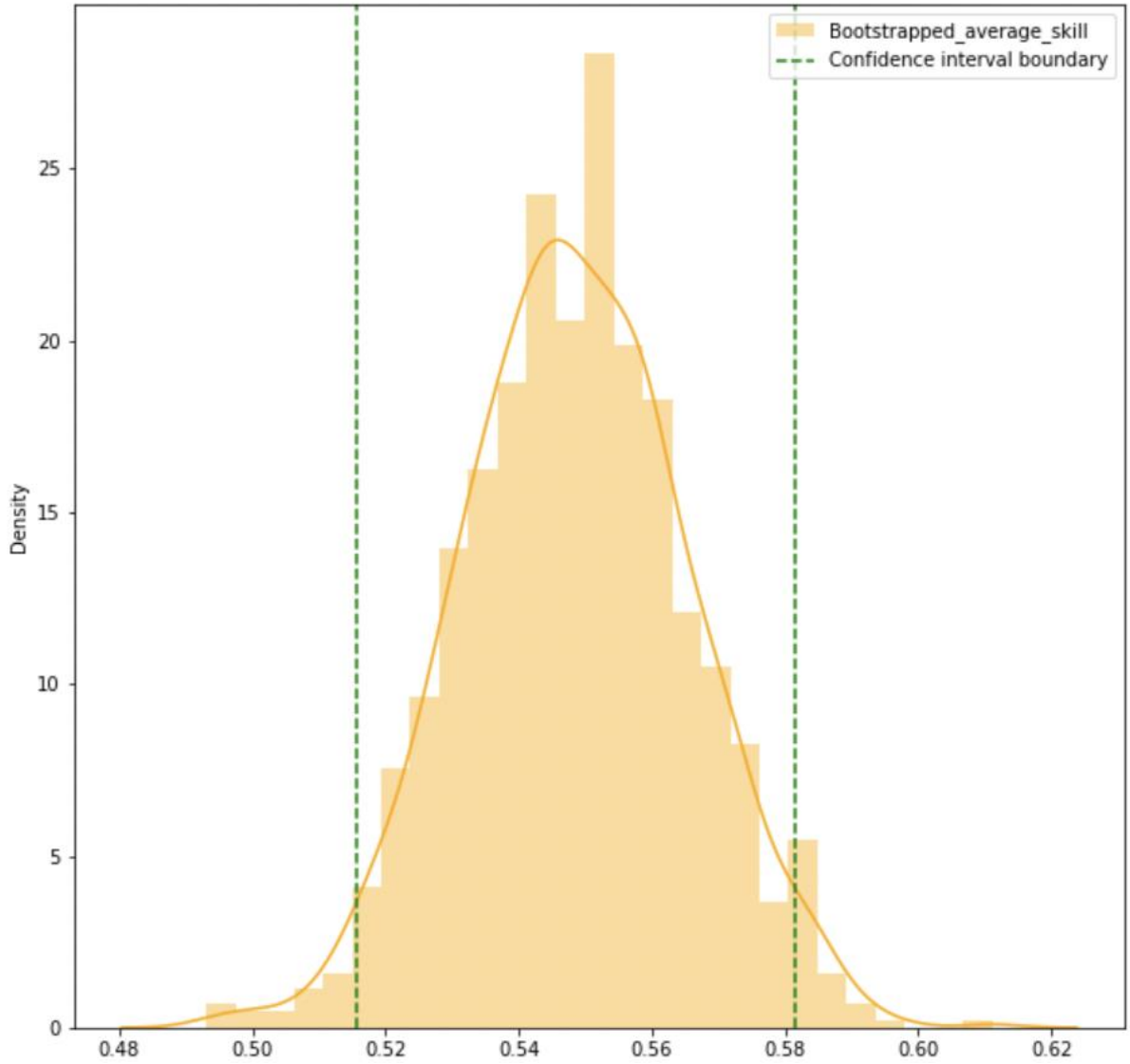
More specifically from the original sample of 240 months we randomly choose smaller subsamples of 150 months, this is because we are trying to avoid any potential bias that may persist in the full sample. Subsamples of 150 were chosen in line with the 0.632 rule put forward by Efron and Tisbshirani (1994). This involves taking subsamples by selecting 63.2% of the original sample set, often used to create training sets. This allows us to strike a balance between the bias and variance inherent in any single subsample. Therefore, allowing us to obtain more representative estimates of the average skill we are looking for. For the implementation of MMB within each subsample we find the possible blocks by taking the 1st observation + block size, then the 2nd observation + block size, and so on until all possible blocks are found, equal to:

$$subsample\ size - block\ size + 1.$$

Hall et al (1995) found that the optimal block size for bootstrapping with dependent data occurs at the balance between the error-about-the-mean and the bias in order to minimise the mean squared error. Moreover, it was found a block length of $n^{1/5}$ works well for estimating a two-sided distribution of an unknown parameter where n=sample size (in our case subsample). The optimal block length depends on the specific problem and sample size, nonetheless in our case we use 3 ($150^{1/5}$). Blocks are chosen until the number of observations are equal to the subsample size of 150, thus 50 blocks are chosen ($\frac{subsample\ size}{block\ size}$), these blocks are chosen at random and with replacement. Once all blocks are chosen, they are concatenated together to form a resampled dataset.

For each resampled dataset we then apply the aforementioned function to find the average skill required for an outperformance of >1%. We set the required number of instances

where outperformance is >1% to 10, meaning that for each resampled dataset we will take the mean of 10 required skill levels. We then set the number of resampled datasets to 1,000 meaning that we will achieve 1,000 average skill levels. The distribution of the means are plotted on the figure below:



From this distribution we find the mean average skill level required is 0.549 with a confidence interval of 0.515 to 0.582.

In this scenario we assume this team can implement a tactical asset allocation that obtains an average skill level of 0.55. This can be achieved through the framework proposed by MT Faber (2007) which suggests using relative strength, moving averages, value, or risk management. These strategies involve selecting assets based on their performance, using price trends, to determine when to hold or sell, selecting undervalued assets, or adjusting asset allocation based on market risk. As 0.55 falls within the confidence interval of the average skill level needed to produce an annual outperformance of >1% it corroborates the notion that this team can outperform the 60:40 benchmark.

Question 2

While having a skill level within the predetermined range is a positive sign it does not guarantee you will consistently achieve an annual outperformance of >1%. Even with our average skill of 0.55 the average annualised outperformance with the current weightings of 70:30 and 50:50 is 0.0046 on the original dataset – if we take the average outperformance of 50 instances where the skill level was 0.55 – and using the MBB method we get an average of 0.0049 with a 95% confidence interval of 0.0039 to 0.0060. This is obtained through the "avg_outperformance_for_skill" function in the accompanying code. This function works in a similar way to the "avg_skill_for_outperformance" function but in this function, we append the annual outperformance to a list if the skill is within the range 0.545 to 0.554. which results in collecting the outperformance of portfolios with an average skill 0.55. This is continuously repeated until we have appended a desired number of outperformances to our list of which we can take the average. For the previous confidence interval, we took 50 instances from 1000 resampled datasets. This shows that with our skill level we will still beat the benchmark but not necessarily by 1%.

By adjusting the possible weightings to 80:20 and 40:60, we can achieve a notable improvement in the average annualised outperformance. Running the "avg_outperformance_for_skill" with the new weights on the original dataset reveals an average outperformance of 0.0094 while using the BMM method, we obtain an average of 0.0096 and a confidence interval of 0.0075 to 0.0116.

Notably, the confidence intervals of the average outperformances for the different weighting options do not overlap. Therefore, we can assert with 95% confidence that the average outperformances are statistically different, and the new weightings indeed result in a greater average outperformance.

The arguments presented thus far demonstrate that, based on the current and resampled datasets, a skill of 0.55 can yield an average outperformance over the benchmark. However, these results are based on the returns path observed over the last 20 years. Future paths of returns may take a different course, which highlights the importance of further analysis to confirm the team's ability to beat the benchmark and what future average outperformance we might achieve with a skill of 0.55.

To this end, we will examine the level of outperformance achievable with a skill level of 0.55 on simulated future return paths. To generate these paths, we will leverage the Monte Carlo simulation approach. Specifically, we will incorporate an underlying Geometric Brownian Motion (GBM) with drift. This is a Markov process where the predicted returns are independent of the historical

The formula used for the Monte Carlo simulation is as follows:

$$r = \mu\Delta t + \sigma\epsilon\sqrt{\Delta t}$$

r = return
μ = drift
Δt = time interval

σ = volatility
$\epsilon\sqrt{\Delta t}$ = the Geometric Brownian Motion process scaled by the square root of the time increment

The GBM model predicts the return will be the expected return subject to a random positive or negative shock, created by the random variable generated by the Brownian motion process component. To parameterise this model, we estimate the drift as the mean of the original sample returns, and the standard deviation as the volatility. This approach ensures that the simulations generated are consistent with the statistical properties of the underlying assets, resulting in more accurate predictions.

Given that the GBM model assumes a lognormal distribution of returns, we use log returns instead of simple returns in our simulation. This approach better reflects the nature of a lognormal distribution and allows for more accurate modelling. By incorporating these techniques, we can generate simulations that are more closely align with the underlying asset's statistical properties.

Therefore, as we have assumed the data consists of simple returns, we will have to convert them to log returns to parameterise the model, this is achieved through:

$$\log returns = \log\left(simple\ returns + 1\right)$$

To generate these return paths two functions are created. The first function, "Simulation", predicts a single month return with only one iteration where inputs are expected return and expected volatility. Then the following function "generate_return_path" requires the expected return and volatility for both Equity and Fixed Income and then calls the previous function looping it 240 times to create a 20-year return path for both Equity and Fixed Income. In this case we have assumed there is no correlation between Equity and Fixed Income as their monthly returns are generated independently. This will partially reduce the accuracy of our predictions as these two assets are correlated, as supported by Marsh and Pfleiderer (2010), who found there is a statistically significant correlation between these assets that dynamically changes over time.

The simulation process only iterates once for each month, meaning we only generate one simulated return for each month. The reason for this is that if we were to take many iterations and take the average, the random component (or "shock") created from the GBM process would eventually converge to its mean of zero. This would result in the generated return converging to the drift or expected return. In both Equity and Fixed Income, the mean return is far smaller than the volatility, for Equity its 0.0952 to 0.057 and for Fixed Income its 0.003 to 0.0198. Therefore, the resulting returns would not be representative of the actual volatility in the market.

Therefore, to ensure our simulated returns accurately reflect the underlying asset's volatility, we need to keep the random component as a key part of the prediction process. Hence by only iterating once for each month we can maintain the appropriate level of randomness and generate more accurate simulations that reflect the true statistical properties of the asset.

To find the average outperformance of a skill of 0.55 we generate 1000 possible future paths and for each one we apply the “avg_outperformance_for_skill” function with a desired number of instances at 20, therefore returning 1000 average outperformances. After we have obtained these 1000 averages, we then take another average to give us the resulting average outperformance that can be achieved by this team with an average skill of 0.55 which we find to be 0.011.

In conclusion we have demonstrated that this team has the skills to beat a 60:40 benchmark through demonstrating that a skill of 0.55 would outperform the benchmark 0.39% to 0.60% with weighting options of 70:30 and 50:50 or 0.75% to 1.16% with weighting options of 80:20 and 40:60. Additionally, this skill comfortably sits in the confidence interval of the average skill required to beat the benchmark by 1% which is 0.51 to 0.58.

References:

www.quantstart.com. (n.d.). *The 60/40 Benchmark Portfolio | QuantStart*. [online] Available at: https://www.quantstart.com/articles/the-6040-benchmark-portfolio/ [Accessed 2 Mar. 2023].

Princeton (1996). *The Econometrics of Financial Markets*.

LaMorte, W. (2016). *Central Limit Theorem*. [online] sphweb.bumc.bu.edu. Available at: https://sphweb.bumc.bu.edu/otlt/MPH-Modules/BS/BS704_Probability/BS704_Probability12.html.

JEGADEESH, N. (1990). Evidence of Predictable Behavior of Security Returns. *The Journal of Finance*, [online] 45(3), pp.881–898. doi:https://doi.org/10.1111/j.1540-6261.1990.tb05110.x.

HALL, P., HOROWITZ, J.L. and JING, B.-Y. (1995). On blocking rules for the bootstrap with dependent data. *Biometrika*, 82(3), pp.561–574. doi:https://doi.org/10.1093/biomet/82.3.561.

Efron, B. and Tibshirani, R.J. (1994). *An Introduction to the Bootstrap*. [online] *Google Books*. CRC Press. Available at: https://books.google.co.uk/books?hl=en&lr=&id=gLlpIUxRntoC&oi=fnd&pg=PR14&dq=Efron+and+Tibshirani+(1994)&ots=AaBy_9OdA1&sig=GMy2nRZo0hEgN50r7y9UNsxQufk&redir_esc=y#v=onepage&q=Efron%20and%20Tibshirani%20(1994)&f=false [Accessed 2 Mar. 2023].

Marsh, T. and Pfleiderer, P.C. (2006). *The Relation between Fixed Income and Equity Return Factors*. [online] papers.ssrn.com. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=947877 [Accessed 2 Mar. 2023].

Faber, M.T. (2007). A Quantitative Approach to Tactical Asset Allocation. *The Journal of Wealth Management*, 9(4), pp.69–79. doi:https://doi.org/10.3905/jwm.2007.674809.