Project 1

Lorenzo Ausiello, Stephan Bilyk, Fabrizio Dimino

2023-10-23

Excercise 1

Table 1: ETFs (4 out of 30) Prices from Jan 2010 to Sept 2023

IVV.Adjusted	IJH.Adjusted	IYM.Adjusted	IYZ.Adjusted
87.73411	60.18383	48.38065	14.38381
87.98888	60.36364	48.79496	14.47481
88.05833	60.68238	49.70958	14.23683
88.44434	60.98487	49.43597	14.18083
88.74538	61.36905	50.12391	14.07584
88.86890	61.25461	49.99100	14.10384

Table 2: ETFs (4 out of 30) Log-Returns from Jan 2010 to Sept 2023

IVV	IJH	IYM	IYZ
0.0028997478	0.002983092	0.008527177	0.006306601
0.0007889972	0.005266453	0.018570554	-0.016577554
0.0043739827	0.004972429	-0.005519281	-0.003940930
0.0033978643	0.006279868	0.013819741	-0.007431469
0.0013908783	-0.001866477	-0.002655108	0.001987510
-0.0092502357	-0.012623363	-0.019580697	-0.016008292

Table 3: Summary table: Mean Return, Volatility and Sharpe Ratio

	Mean.Return.annualized	Volatility.Annualized	SharpeRatio.Annualized
Mean	0.1009057	0.2011773	0.5155555
25%	0.0922163	0.1771209	0.4222272
50%	0.1029094	0.1946887	0.5169379
75%	0.1140247	0.2236544	0.6539740

```
## `geom_smooth()` using formula = 'y ~ x'
```

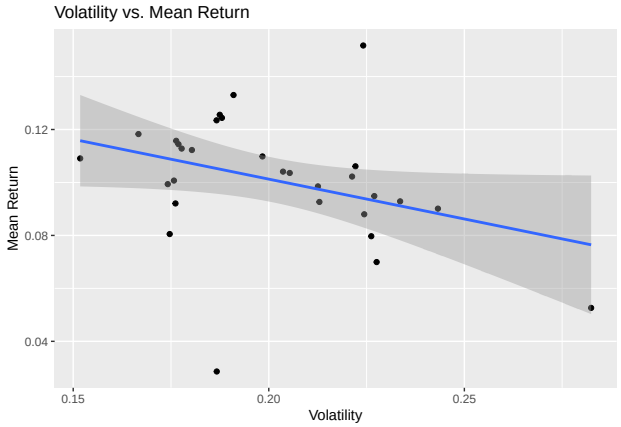


Table 4: Summary table: Beta, Jensen's Alpha, Trey. Ratio, Track. Error, Inf. Ratio

	Beta	Jens.Alpha	Trey.Ratio	Track.Err	Inf.Ratio
Mean	1.0266810	-0.0184291	0.0992870	0.0845243	-0.1490396
25%	0.9828148	-0.0344435	0.0843686	0.0492396	-0.2698441
50%	1.0652492	-0.0191073	0.0968018	0.0903056	-0.1662575
75%	1.1018332	0.0042799	0.1198593	0.1093885	-0.0910654

Table 5: Statistics matrix (Expense ratio included)

	Beta	Jens.Alpha	Trey.Ratio	Track.Err	Inf.Ratio	Expense_Ratio
IJH	1.0811782	-0.0215430	0.0958100	0.0776654	-0.1564114	0.05
IWF	1.0476347	0.0117800	0.1269799	0.0492396	0.3512019	0.19
IJR	1.0998310	-0.0250583	0.0929517	0.1080424	-0.1249908	0.06

	Beta	Jens.Alpha	Trey.Ratio	Track.Err	Inf.Ratio	Expense_Ratio
IWM	1.1398033	-0.0439287	0.0771949	0.1028314	-0.2698441	0.19
IWD	0.9606481	-0.0191073	0.0958454	0.0488214	-0.4846599	0.19
IVW	1.0348209	0.0058026	0.1213428	0.0434500	0.2262972	0.18
IWB	1.0007483	-0.0013881	0.1143484	0.0127184	-0.1023358	0.15
IVE	0.9493078	-0.0104979	0.1046770	0.0489690	-0.3341871	0.18
IUSG	1.0378448	0.0042799	0.1198593	0.0436634	0.1983327	0.04
IUSV	0.9542086	-0.0097202	0.1055489	0.0514325	-0.2920304	0.04
IWV	1.0183388	-0.0055966	0.1102396	0.0170947	-0.2032318	0.20
IYW	1.1569288	0.0178303	0.1311473	0.0969206	0.3713613	0.40
IWN	1.1110163	-0.0488919	0.0717290	0.1147751	-0.3140345	0.24
IWO	1.1730486	-0.0428911	0.0791717	0.1127302	-0.2028138	0.24
IJK	1.0753425	-0.0203602	0.0968018	0.0754465	-0.1542864	0.17
IJJ	1.0824392	-0.0267594	0.0910141	0.0946995	-0.1818195	0.18
IJS	1.0652704	-0.0284228	0.0890542	0.1279434	-0.1631087	0.18
IJT	1.1214016	-0.0236877	0.0946122	0.1035117	-0.0931028	0.18
IYH	0.8121640	0.0242841	0.1456359	0.0915080	0.0278095	0.40
IYF	1.0980859	-0.0344435	0.0843686	0.0903056	-0.2557038	0.40
IYY	1.0029340	-0.0032782	0.1124668	0.0176754	-0.1662575	0.20
IYK	0.7420929	0.0231938	0.1469901	0.0893544	-0.0744810	0.40
IYE	1.1301779	-0.0781745	0.0465654	0.2014229	-0.3133122	0.40
IYG	1.2216119	-0.0512688	0.0737673	0.1195628	-0.2142844	0.40
IYJ	1.0652492	-0.0134580	0.1031018	0.0648583	-0.0910654	0.40
IDU	0.6388584	0.0065807	0.1260363	0.1479065	-0.2380974	0.40
IYC	0.9828148	0.0097242	0.1256297	0.0694039	0.1114530	0.40
IYM	1.1018332	-0.0575822	0.0634751	0.1198624	-0.3820755	0.40
IYZ	0.8681139	-0.0718608	0.0329574	0.1093885	-0.7964700	0.40

```
## `geom_smooth()` using formula = 'y ~ x'
```

Question 1.3.b Once computed the 5 different measures for each ETF, we can highlight that IYH has the highest Alpha. This measure show that this ETF has the highest ability to outperform the index. So, IYH is the best ETF if we consider only the excess returns of the funds over their capability to track the index. However, IYK shows the highest Treynor ratio, thanks to its high alpha and low beta. Therefore, IYK is the best if we consider the amount of expected return for unit of systematic risk. On the other side, if we consider excess returns of the funds over the market index, compared to their idiosyncratic risk, we can notice that IYW is the winner, with the highest Information Ratio. This ratio could be a little bit tricky, since funds with very low betas can have a low information ratio even if they show high alphas and low tracking error volatility. Then, we can highlight that IYZ has the lowest Alpha. This measure show that this ETF underperform the index by 7.19%. So, IYZ is the worst ETF if we consider only the excess returns of the funds over their capability to track the index. IYZ shows also the lowest Treynor ratio, due to its very negative alpha. And finally, if we consider excess returns of the funds over the market index compared to their idiosyncratic risk, we can notice that IYZ is again the loser, with the most negative Information Ratio.

Question 1.3.c The expense ratio is a measure of ETFs operating costs relative to assets. Passive index funds tend to have lower expense ratios than actively-managed funds or those in less liquid asset classes. In this case, we can notice that Expense ratio presents a positive relationship with tracking error volatility. The higher the tracking error volatility, the higher the expense ratio, maybe due to the fact that ETFs showing an higher tracking error are the funds that replicate/track the index less. However, expense ratio present a negative relationship with Alpha, Treynor ratio and Information ratio. This could imply that funds that deviate less from the market index have performed better.

Question 1.4.b CAPM assumes a perfect linear relationship between Expected returns of funds and funds' betas. CAPM considers alphas equal to 0, but in the real-world CAPM doesn't hold because alphas exist and are not 0. However, roughly speaking, assets with higher betas tend to have higher expected mean returns (when expected return of the market is positive), because the higher the beta the more the asset amplify the expected return. However, the linear relationship

is not respected because different assets show different alphas. Data points should cluster around a linear trendline.

Exercise 2

Table 6: Main statistics of IVV, IYW, IYF

	Mean	Volatility	SR
IVV	0.116	0.176	0.656
IYW	0.152	0.224	0.677
IYF	0.093	0.213	0.435

Table 7: Covariance Matrix

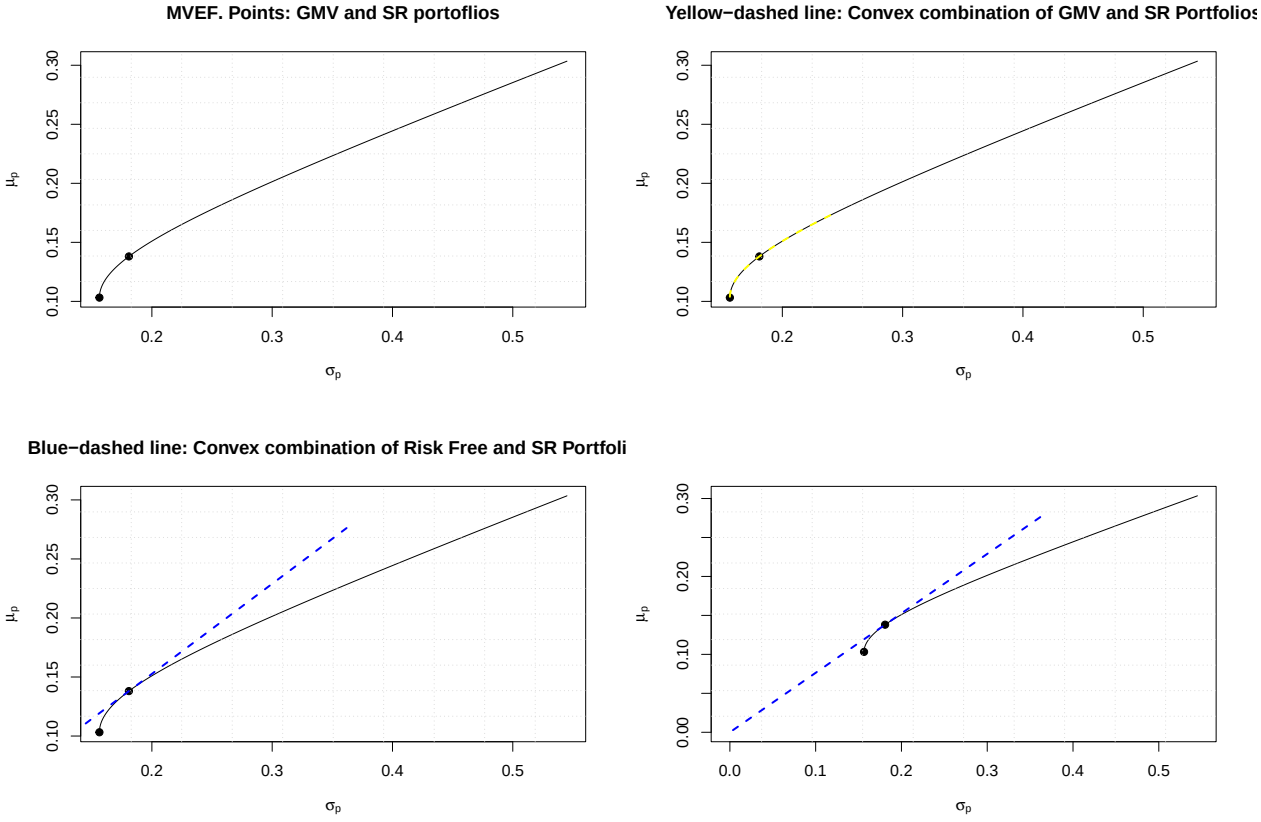
	IVV	IYW	IYF
IVV	0.0310920	0.0359712	0.0341417
IYW	0.0359712	0.0502441	0.0350051
IYF	0.0341417	0.0350051	0.0453465

Table 8: Weights of GMV pfolio (w0), w1 and SR pfolio

	w_0	w_1	SR_pfolio
IVV	2.6580963	-2.123383	2.1547284
IYW	-0.8604302	3.320027	-0.0733866
IYF	-0.7976662	-1.196644	-1.0813418

Table 9: Main statistics of GMV pfolio (w0), w1 and SR pfolio

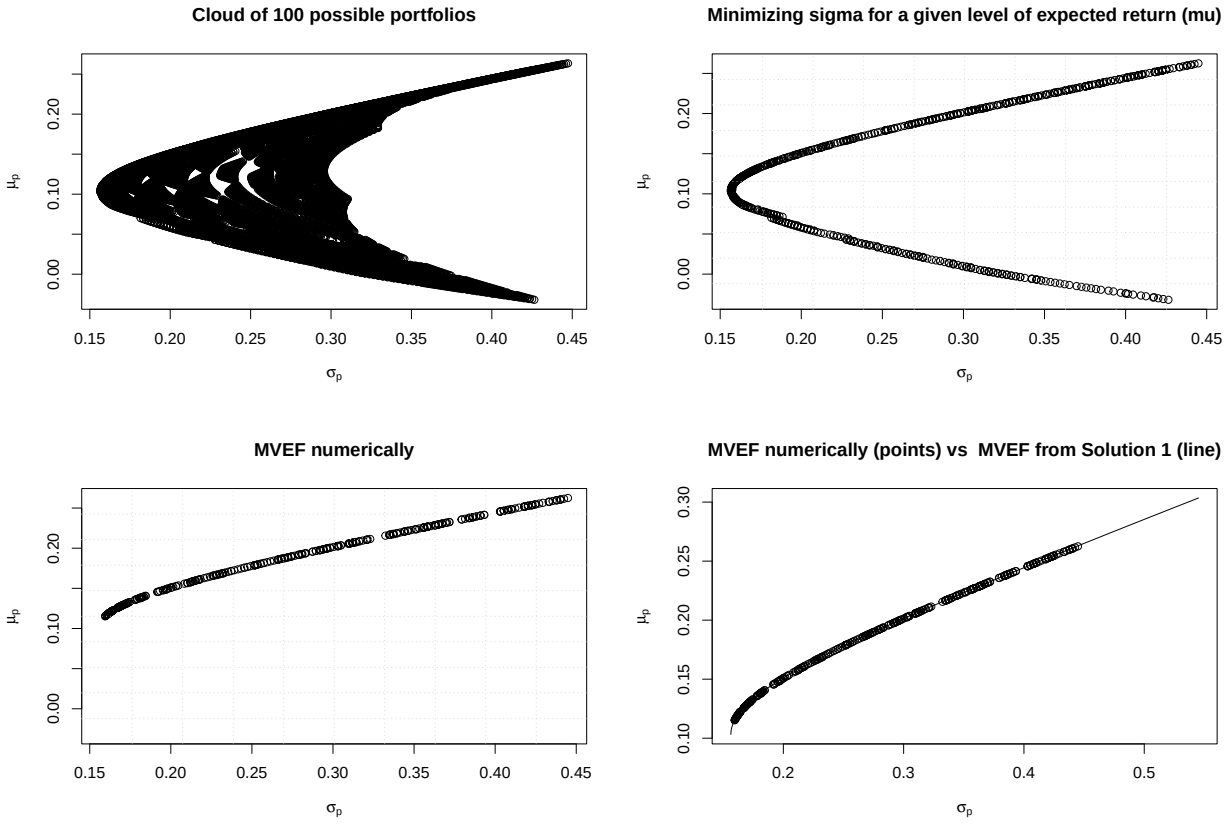
	w_0	w_1	SR_pfolio
mu	0.1031857	0.1471285	0.1380639
sig	0.1564006	0.3835734	0.1809125
SR	0.6597527	0.3835734	0.7631528




```
##
## Call:
## lm(formula = mu_p ~ sig_p, data = metrics_MVEF)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -2.019e-15 -2.060e-17 -1.200e-18  1.680e-17  5.093e-15
##
## Coefficients:
##               Estimate Std. Error    t value Pr(>|t|)
## (Intercept) -2.383e-16  5.721e-18 -4.165e+01   <2e-16 ***
## sig_p        7.632e-01  2.738e-17  2.787e+16   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.28e-16 on 1999 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 7.768e+32 on 1 and 1999 DF,  p-value: < 2.2e-16
```

Table 10: Sharpe Ratio of SR Porfolio (equal to the slope of the regression above)

x
0.7631528



Question 2.3.b When is between -1 and 0, it implies a short position in the GMV portfolio (w0) and a long position in the Sharpe Portfolio (wSR). This would mean that you are borrowing funds in the GMV portfolio and investing them in the Sharpe Portfolio. The negative values represent leveraging your investments, which can amplify both potential gains and potential losses. Moreover, negative values (leverage) imply increasing risk: the resulting portfolio will show higher expected return but also higher volatility respect to SR portfolio (that is riskier than GMV by definition).

Question 2.4.c The straight line of the graph is called the capital allocation line (CAL). It show all the risk–return combinations available mixing the Risk Free asset (0% volatility and 0% expected return) and the Sharpe Ratio Portfolio (risky asset). If we regress mean returns on volatility using all the feasible combinations (complete portfolios from now on), the *Slope* represents the increase in the mean return of the complete portfolio per unit of additional standard deviation. For this reason, the slope is equal to the Sharpe ratio of the Sharpe ratio portfolio: when volatility of complete

portfolio (independent variable) will be equal to volatility of SR portfolio, the mean return will be equal to the mean return of the SR portfolio. The *intercept* is equal to the mean return of the Risk Free (0%), that is the mean return of the complete portfolio when volatility is equal to 0% (so, when allocation in SR portfolio is 0 and allocation in Risk Free is 1).

Excercise 3

Breaking Even

Table 11: Expected number of tosses to get 3 heads in a row

x
13.97701

Table 12: $k to play a fair game (E[PNL] = 0)

x
0.0715461

Turtles

Table 13: Expected number of groups. Number of turtles = 10

x
2.92808

Trees

Table 14: True values 2 steps

	x
Exp_P2	102
Var_P2	198

Table 15: Simulated values 2 steps

	x
Exp_P2	101.9796
Var_P2	197.5952

Table 16: Simulated values 10 steps

	x
Exp_P10	110.0912
Var_P10	985.5535

Breaking Even The game consists in a fair coin that is repeatedly tossed. You win a $1 when you get three heads in a row for the first time. However, you have to pay k for each toss. In this task, we will use a Monte Carlo simulation to determine the maximum price (k) you would be willing to pay per coin toss that makes the game break even. To find the value of k that makes the game break even, you need to determine the expected value of the profit and loss (PNL), which should be equal to zero. Therefore, we will simulate the game 10^5 times, by repeatedly tossing a fair coin until three heads in a row are achieved. For each game, we will calculate the number of tosses needed to get three heads in a row. Once obtained 10^5 simulations, we perform the mean of the number of tosses needed in each experiment to find the expected number of tosses needed to get three heads in a row. We will use this value to find the value of k that results in a break-even game: $E[PNL] = \$1 - E[X] \times \$k = 0$. So, k should be equal to 1/E[X] to have a fair game.

Code explained: the function called simulate_game takes one argument n. This argument n represents the number of times you want to simulate the game. In the function we create a vector named options containing two elements: "head" and "tail". These represent the possible outcomes of a fair coin toss. Then, we start a while loop, that continues as long as the consecutive_heads variable is less than 3, which means it will keep looping until you get three consecutive "head" outcomes. In each iteration of the loop, the tosses variable is incremented by 1, representing the current toss number. We use the sample function to simulate a fair coin toss. It randomly selects one element from the options vector with equal probability (0.5 for "head" and 0.5 for "tail"). If the result is "tail," it enters the if block. If the outcome of the coin toss is "tail," it sets consecutive_heads back to 0 because the consecutive streak of "head" outcomes is broken. If the outcome of the coin toss is "head," the code enters the else block. In this case, it increments the consecutive_heads variable by 1, representing a consecutive "head" outcome. his closing brace ends the while loop. The loop continues until you achieve three consecutive "head" outcomes in a row. Finally, this line returns the total number of coin tosses it took to achieve three consecutive "head" outcomes. We set n=10^5 and then we compute the mean of the resulting vector to get E[X] and then k=1/E[X].

Turtles In this game, we have n turtles crawling in the same direction in a one way street. Their initial velocity is different (from 1 to n cm/s), and the initial position is completely random. We will simulate 10^5 experiments (in each experiment turtles have different velocities and positions) to determine how many groups the turtles will divide into after enough time. The logic is the following: if the front turtle is slower than the rear turtle, the latter will slow down until it reaches the velocity of the front one. The two turtle will form the group. Then, if the third turtle presents an higher velocity than the first one, it will join the group, otherwise it will form another different group. An so on for each of the n turtles. To find the expected number of groups formed by the turtles (given a certain number n of turles partecipating), we will create multiple experiments. In each experiment, we will simulate the movement of the turtles with different initial velocities and positions. We will keep track of how the turtles form groups and calculate the average number of groups formed in the different experiments.

Code Explained: We define a function named turtle_experiment that takes one argument, n, which represents the number of turtles in the experiment. Inside the function, it generates a random sample of integers representing the turtles (remember different postions and different velocities: value is the velocity and index is the position). The line groups <- unique(cummin(turtles)) calculates the unique cumulative minimum of the turtles vector. The cummin function computes the cumulative minimum, meaning it keeps track of the minimum value encountered as it moves through the vector. The unique function then extracts the unique values, which represent the unique groups of turtles formed as the cumulative minimum changes. In other words, it identifies how many groups of turtles were formed over time. The function returns the length of the groups vector, which indicates the number of unique groups formed during the experiment. This represents the final result of the simulation. Then we uses sapply to apply the turtle_experiment function to simulate the experiment 100,000 times. It records the number of unique groups formed in each simulation run and stores these values in the num_groups vector. Finally, we calculate the mean (average) of the num_groups vector. This represents the expected number of unique groups formed over a large number of simulations, providing insight into how the turtles tend to organize themselves into groups on average.

Trees In this task, we will work with a binomial model to simulate the price process and estimate the expected value and variance at a specific step. First of all we create a vector P2 containing 3 possible values for a variable and then we create a vector prob with probabilities corresponding to each value in P2. Therefore, we calculate the expected (mean) value by summing the product of each value in P2 and its corresponding probability. The variance is computed by summing the squared differences between each value in P2 and the expected value, weighted by their probabilities. These are the true values for variance and mean.

Simulated_p2 is a function that simulates 2 steps of a random process. It starts with an initial value s = 100. It then simulates two steps where s1 is randomly selected from s + 10 with a 55% probability or s - 10 with a 45% probability, and s2 is randomly selected from s1 + 10 with a 55%

probability or s1 - 10 with a 45% probability. Then we simulate this process 100,000 times, resulting in a vector of simulated values. Fianlly, we calculate and display the mean and variance of the simulated values in a table. Similar to the previous section, we simulate 10 steps instead of 2 using the simulated_p10 function. Then we simulates the 10-step process 100,000 times and calculates the mean and variance of the simulated values. The code essentially compares the expected and variance values for a true 2-step process with the values obtained through simulation for both 2 and 10-step processes. It uses random sampling to simulate the process and then computes statistics based on those simulations.

Excercise 4

```
## [1] "IVV"
```

Table 17: Calibrated Values: mu hat and sigma hat

mu_hat	sigma_hat
0.1309752	0.1470194

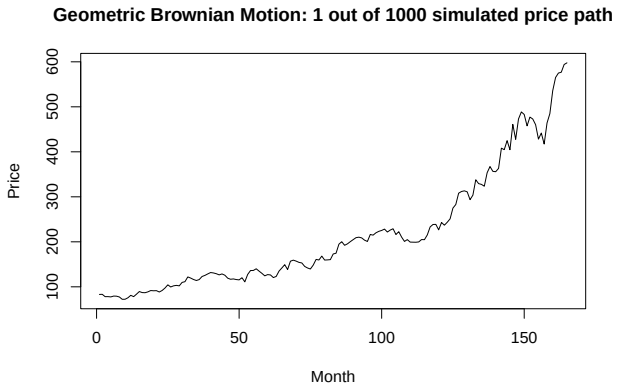


Table 18: Simulated Expectation of S(t) after 164 month

x
501.3902

Table 19: True Value: Expectation of S(t) after 164 month

x
497.7805

Table 20: Simulated Variance of S(t) after 164 month

x
283.3827

Table 21: True value: Variance of S(t) after 164 month

x
291.8135

Table 22: True values vs Simulated Values

	Mean	Sigma
Simulation	501.3902	283.3827
True Value	497.7805	291.8135

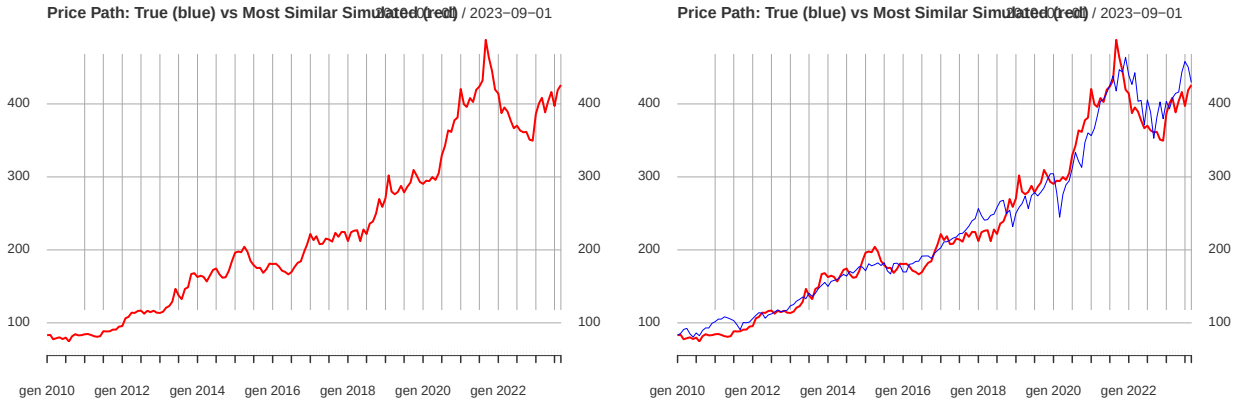


Table 23: 99% Pfolio position VaR based on Simulated path: 100 IVV ETFs

	x
1%	37065.34

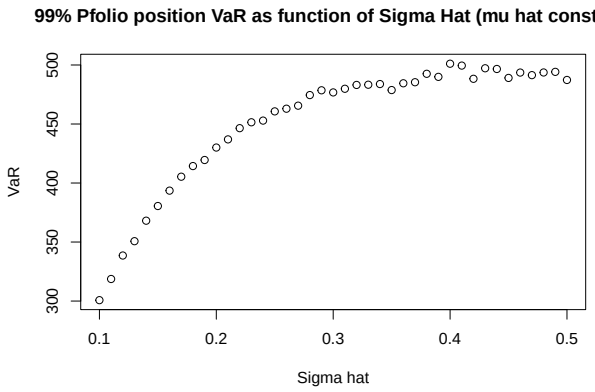


Table 24: 99% Returns VaR: historical approach

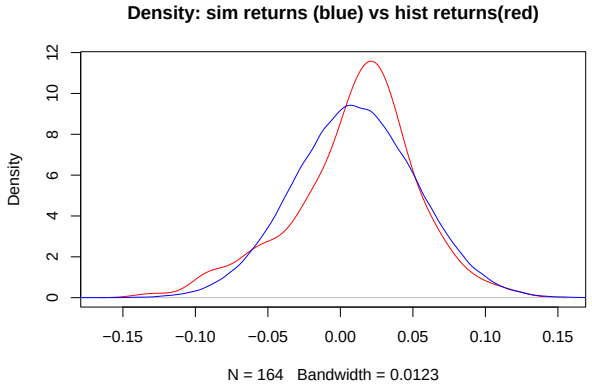
	x
1%	0.1044194

Table 25: 99% Returns VaR: parametric approach

	x
1%	0.0989902

Table 26: 99% Returns VaR: true value

x
0.0987322



Bonus Question 2

Table 27: 99% pfolio position VaR from closed form solutionh: 100 IVV ETFs

x
37650.43

Table 28: 99% Pfolio position VaR based on Simulated path: 100 IVV ETFs

	x
1%	37065.34

Question 4.e.ii The graph that shows 99% VaR as funcion of demonstrates that VaR is very sensitive to the accuracy of calibrated parameters. An incorrect choice of calibrated could lead to incorrect assessments of portfolio risk. In fact, different parameters estimation involves a completely different consideration of the value that the pfolio will present in extremely negative cases, that is "wrong". Therefore, inaccurate parameters lead to a sub-optimal risk management. This is typically called *Model Risk*, a risk that can be evaluated and mitigated using stress testing and back-testing. If VaR is extremely sensitive to changes in model parameters, this indicates the need for accurate calibration and the most appropriate model to manage risk effectively.

Question 4.e.iii If we look at the shape of the function (VaR as a function of sigma) we can notice that the relationship between VaR of the portfolio position and its volatility (sigma) is nonlinear. Particularly, the VaR does not increase linearly, but in a decreasing way, with increasingly slower growth as sigma increases. This is due to the fact that the returns are normally distributed, and therefore portfolio position is lognormally distributed. As proof, the VaR will be function of sigma

as follows: vd formula *Bonus question 2*. So, keeping mu hat constant, we will have VaR equal to a constant minus esponential of the negative sigma, that assumes the shape in the plot above. This form is in contrast to the case of a Gaussian IID (Independent and Identically Distributed) process in which VaR grows linearly with volatility. In the case of a Gaussian IID process, VaR is proportional to standard deviation and follows a linear relationship. Particularly, in this case we have VaR = -Z*sigma.

Question 4.f.iii We can observe that 99% VaR of the portfolio return computed with Historical and Parametric approach are different. In fact, VaR Historical is the VaR computed considering only one realization of the price path (and therefore returns). So, considering only 165 realizations of the monthly returns (that are normally distributed with mu and sigma parameters), the VaR could be different from the expected one. The VaR Parametric, instead, is computed considering 165000 realizations (1000 price path) of returns drawn from the same normal distribution with the same parameters than before. However, the sample is very large in this case and VaR will be very close to -Z*sigma.

Question 4.f.iv We can observe that the density functions of historical returns and simulated returns have different shapes. The density function of simulated returns is very very close to a normal distribution with mu and sigma as parameters (we are considering 165000 realizations drawn from that distribution), The historical distribution presents the same mu and the same sigma (parameters used as inputs to compute calibrated values) but the distribution is slightly different, with a fatter tail on the left side (therefore VaR will be higher).

Question 4.f.v Using two different approaches (historical and parametric) to calculate VaR provides a more comprehensive perspective on risk management. The parametric VaR may underestimate the risk compared to historical VaR, but it is essential to evaluate the trade-offs between the two methods, including data sensitivity and underlying assumptions. The model's ability to replicate the distribution of observed returns is a crucial indicator of its effectiveness. In general, this exercise highlights the importance of understanding the limitations and implications of VaR calculation methods and the significance of proper model calibration. Different approaches can yield varying results, and considering multiple perspectives can help mitigate model risk.The discrepancy between historical and parametric VaR emphasizes the sensitivity of risk models to their underlying assumptions. Understanding and critically evaluating these assumptions is crucial for robust risk management.