Monte Carlo with a Positive Trend

Simulating Changes in Portfolio Values

In [13]:
```
import pandas_datareader.data as web
import datetime

start = datetime.datetime(2005, 1, 1)
end = datetime.datetime.today()

data_dict = {}
stocks = "MSFT", "AAPL", "IBM", "GOOG"
for name in stocks:
    data_dict[name] = web.DataReader(name, "yahoo", start, end)

# make a list of the close data for each stock
close_data_dict = {key:val["Close"] for key, val in data_dict.items()}
close_data_df = pd.DataFrame(close_data_dict)
```

In [14]:
```
close_data_pct_change = close_data_df.pct_change()
close_data_pct_change
```

Out[14]:

Date	MSFT	AAPL	IBM	GOOG
2005-01-03	NaN	NaN	NaN	NaN
2005-01-04	0.003740	0.010270	-0.010742	-0.040501
2005-01-05	-0.002235	0.008758	-0.002068	-0.005090
2005-01-06	-0.001120	0.000775	-0.003109	-0.025632
2005-01-07	-0.002991	0.072811	-0.004366	0.028109
...	...	...	...	...
2021-08-16	0.005976	0.013548	0.002864	0.003685
2021-08-17	-0.005160	-0.006154	-0.008148	-0.011629
2021-08-18	-0.008018	-0.025501	-0.020713	-0.005320
2021-08-19	0.020775	0.002323	-0.010396	0.002515
2021-08-20	0.025575	0.010157	0.007897	0.011127

4188 rows × 4 columns

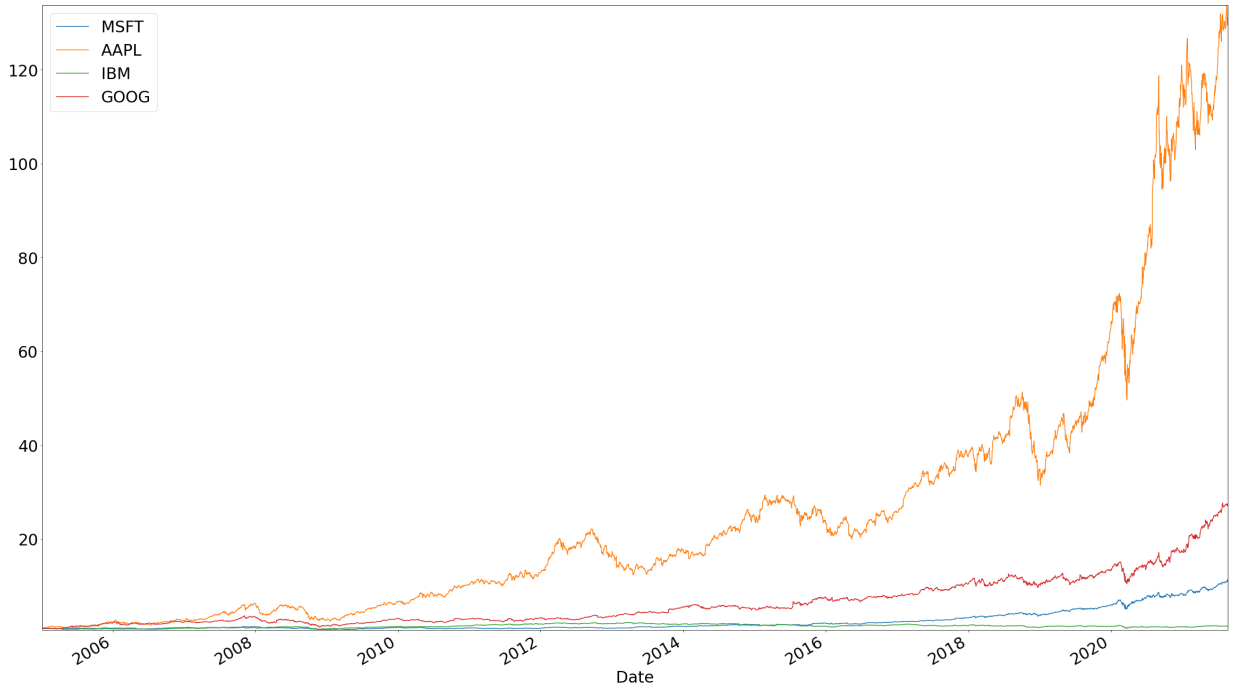
In [15]:
```
close_data_normalized = close_data_pct_change.add(1).cumprod()
close_data_normalized.fillna(1, inplace =True)
close_data_normalized
```

Out[15]:

Date	MSFT	AAPL	IBM	GOOG
2005-01-03	1.000000	1.000000	1.000000	1.000000
2005-01-04	1.003740	1.010270	0.989258	0.959499
2005-01-05	1.001496	1.019118	0.987212	0.954615
2005-01-06	1.000374	1.019908	0.984143	0.930146
2005-01-07	0.997382	1.094169	0.979847	0.956292
...	...	...	...	...
2021-08-16	11.017203	133.713322	1.468951	27.514517
2021-08-17	10.960359	132.890450	1.456982	27.194541
2021-08-18	10.872476	129.501605	1.426803	27.049853
2021-08-19	11.098354	129.802439	1.411969	27.117889
2021-08-20	11.382199	131.120819	1.423120	27.419643

4188 rows × 4 columns

In [16]:
```
fig, ax = plt.subplots(figsize=(40,24))
close_data_normalized.plot.line(ax = ax)
plt.show()
plt.close()
```



Daily Rate of Return

Derive the daily rate of return from the present value equation.

$$PV = \frac{FV}{(1+r)^t}$$

Overall rate of return: $R = \frac{FV}{PV}$

Discount the overall rate of return to find the average rate of return. We will use this to define the mean of the standard normal distribution:

$$r = (R)^{\frac{1}{t}} - 1$$

t is measured in days

In [17]:
```
mean_var_df = pd.DataFrame({"mean": close_data_normalized.iloc[-1].div(close_data
                                    1 / close_data_normalized["MSFT"].count()) - 1,
                            "sigma" : close_data_pct_change.std()})
mean_var_df
```

Out[17]:

	mean	sigma
MSFT	0.000581	0.017127
AAPL	0.001165	0.020875
IBM	0.000084	0.014578
GOOG	0.000791	0.018679

In [18]:
```
num_sims = 500
dates = list(close_data_df.index)
monte_carlo_sim_dict = {}
for stock in stocks:
    monte_carlo_sim_dict[stock] = {}
    mean = mean_var_df["mean"][stock]
    sigma = mean_var_df["sigma"][stock]
    run_monte_carlo(mean, sigma, num_sims, monte_carlo_sim_dict[stock], dates)
#    monte_carlo_sim_dict[stock] = pd.DataFrame(monte_carlo_sim_dict[stock]).add(
monte_carlo_sim_dfs = {stock: pd.DataFrame(monte_carlo_sim_dict[stock]).add(1).cu
```

In [19]:
```
monte_carlo_sim_dfs
```

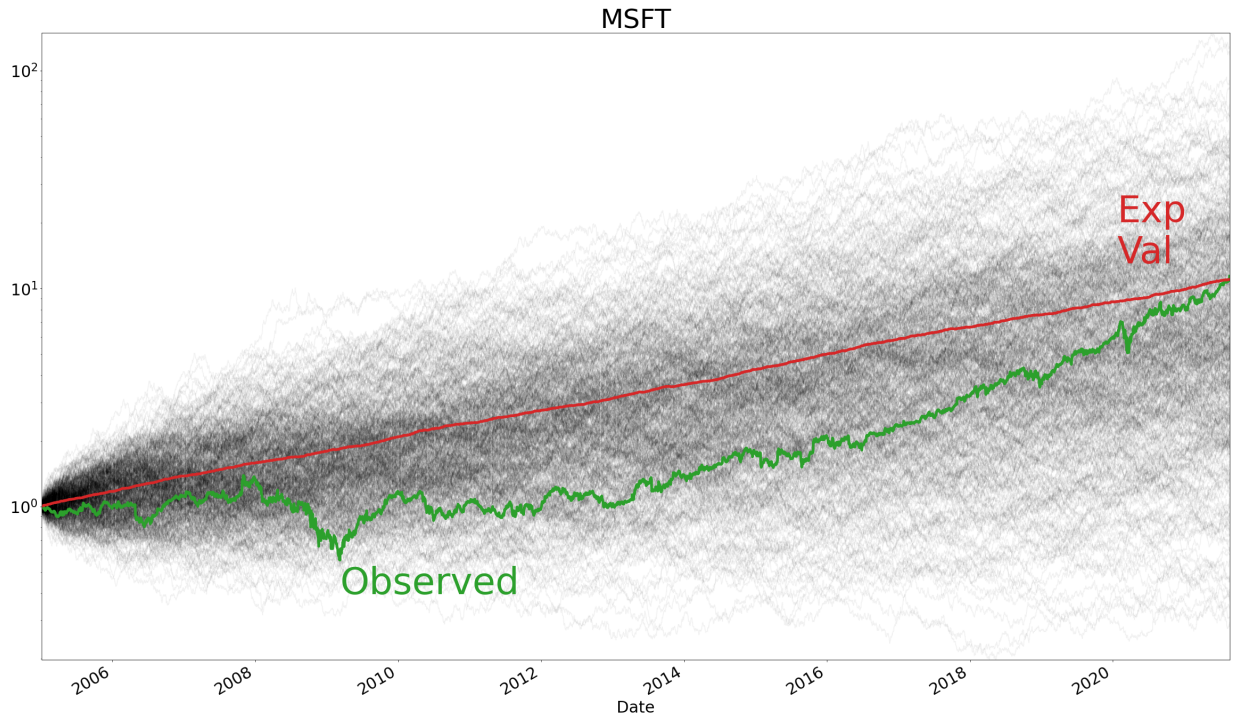
Out[19]:
```
{'MSFT':                 0         1         2         3         4
5    \
 2005-01-03  1.053563  1.003726  0.984080  1.009708   1.041833   1.003781
 2005-01-04  1.052441  1.021892  0.964191  1.017376   1.051140   1.019242
 2005-01-05  1.059702  1.018319  0.966042  1.050270   1.012380   1.026631
 2005-01-06  1.054814  1.024562  0.965205  1.084795   1.073637   1.037407
 2005-01-07  1.065333  1.029700  0.981678  1.069120   1.038813   1.052714
 ...              ...       ...       ...       ...        ...        ...
 2021-08-16  4.239206  7.681533  2.201181  8.085827  21.114711  40.516364
 2021-08-17  4.359741  7.704332  2.156041  7.936421  21.419464  41.130207
 2021-08-18  4.352891  7.869818  2.114742  7.946706  21.323160  40.631783
 2021-08-19  4.314441  7.743833  2.108032  8.028576  20.933080  41.339213
 2021-08-20  4.415739  7.417982  2.135793  7.924041  20.878240  40.742088

                   6         7         8         9    ...       490        49
1  \
 2005-01-03   0.982811  0.975780  1.004706  0.966495  ...  1.012325   0.99815
6
 2005-01-04   0.967031  0.973529  1.015386  0.972135  ...  1.019464   1.03356
```

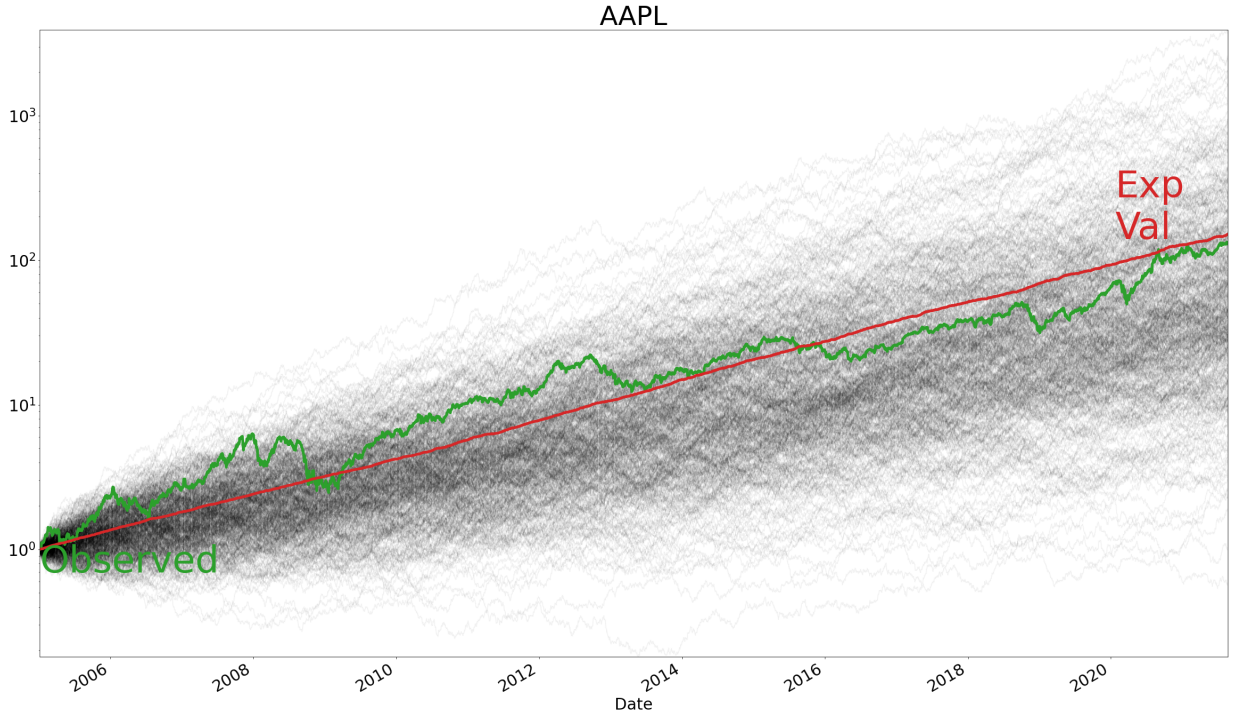
In [20]:
```
def plot_monte_carlo_sim(sim_data, observed_data = None, title = None, logy = Tru
    sim_data["mean"] = sim_data.mean(axis=1)
    index = sim_data.index
    fig, ax = plt.subplots(figsize = (40, 24))
    sim_data.drop(["mean"], inplace = False, axis = 1).plot.line(
            legend=False, marker =".", markersize = .1, color = "k", alpha = .05,
    if observed_data is not None:
        observed_data.plot.line(legend = False, color = "C2", linewidth = 5, logy

        # find x coordinate of lowest value observed
        obs_text_x = observed_data[observed_data == observed_data.min()].index
        plt.text(obs_text_x, observed_data.loc[obs_text_x] * .7,
                 "Observed", fontsize = 70, color = "C2")
    sim_data["mean"].plot.line(legend = False, color = "C3",
                               linewidth = 5, logy = logy, ax = ax)
    plt.text(index[-400], sim_data["mean"].iloc[-400] * 1.5, "Exp\nVal",
            fontsize = 70, color = "C3")



    plt.title(title, fontsize = 50)
    plt.show()
    plt.close()
```

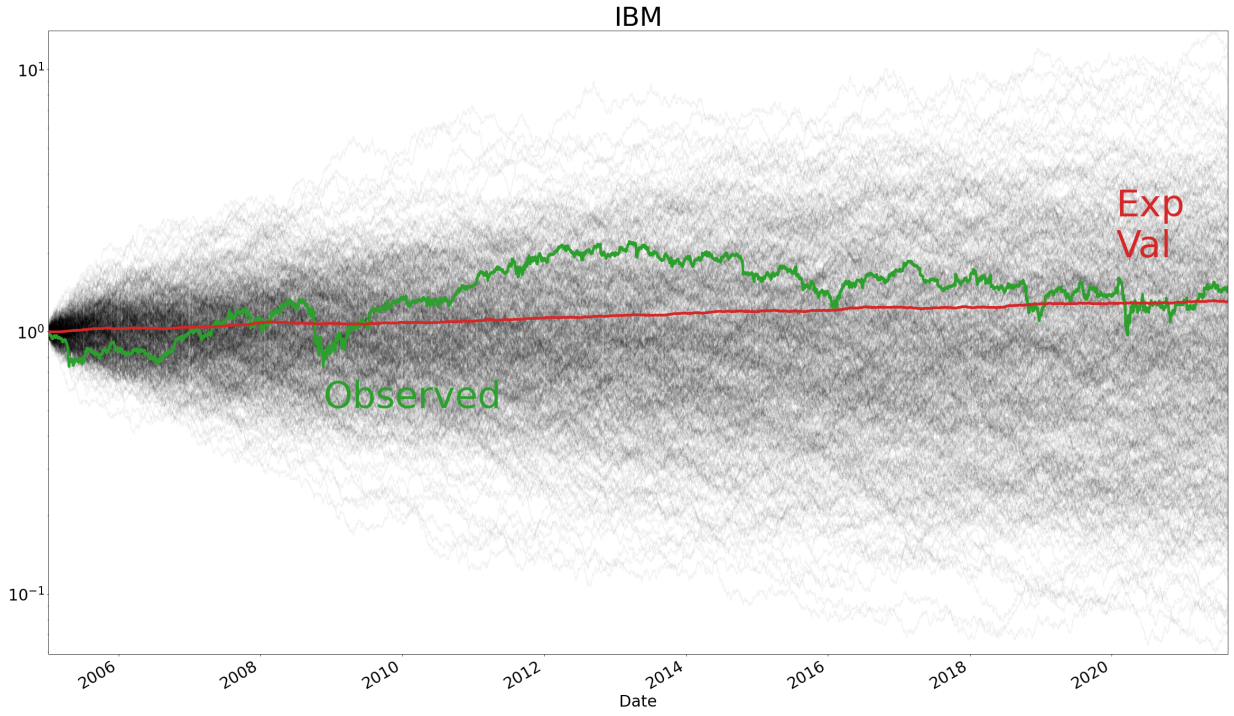
In [21]:
```
monte_carlo_sim_dfs[stock].median(axis = 1)
```

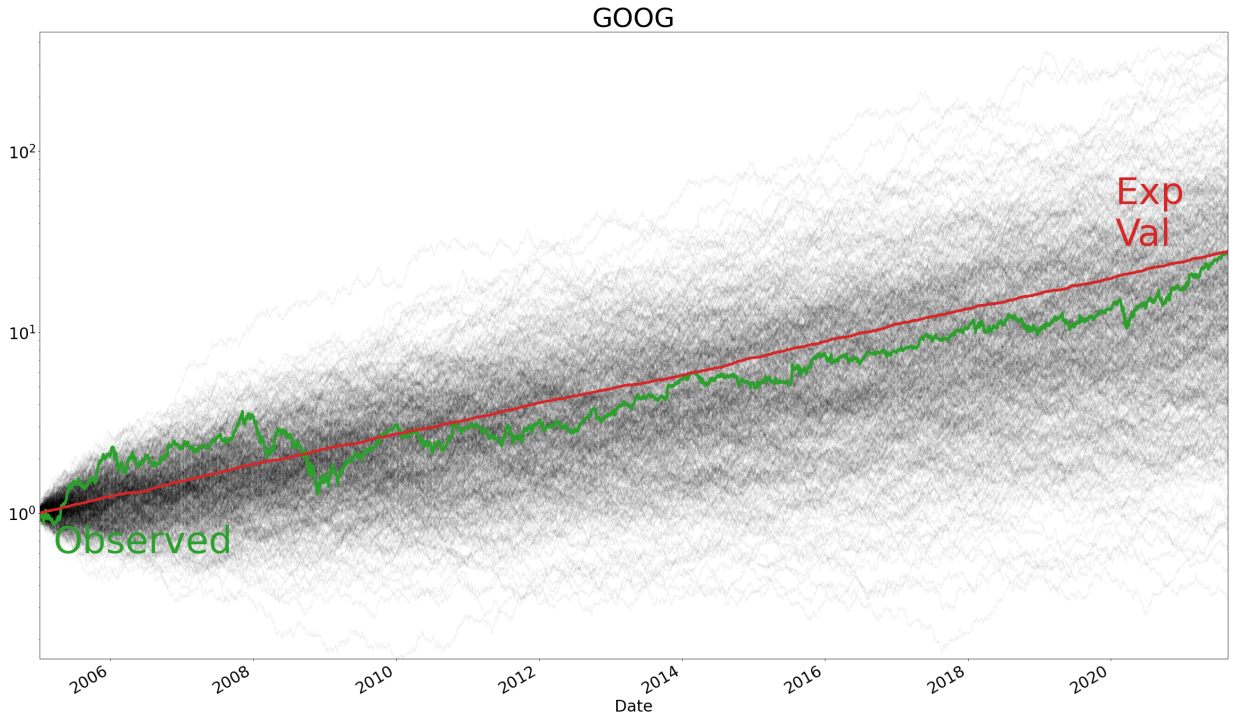
Out[21]:
```
2005-01-03     1.000026
2005-01-04     1.000996
2005-01-05     1.002078
2005-01-06     1.002986
2005-01-07     1.004279
                 ...
2021-08-16    15.428141
2021-08-17    15.379355
2021-08-18    15.372501
2021-08-19    15.429274
2021-08-20    15.436144
Length: 4188, dtype: float64
```

In [22]:
```
for stock in stocks:
    plot_monte_carlo_sim(monte_carlo_sim_dfs[stock], close_data_normalized[stock]
```




AAPL
10^3
10^2
10^1
10^0
Exp
Val
Observed
2006
2008
2010
2012
2014
2016
2018
2020
Date


IBM
10^1
10^0
10^{-1}
Exp
Val
Observed
2006
2008
2010
2012
2014
2016
2018
2020
Date



Calculate Portfolio Return

In [23]:
```
portfolio_weights = {stock: 1 / len(stocks) for stock in stocks}
portfolio_weights
```

Out[23]:
```
{'MSFT': 0.25, 'AAPL': 0.25, 'IBM': 0.25, 'GOOG': 0.25}
```

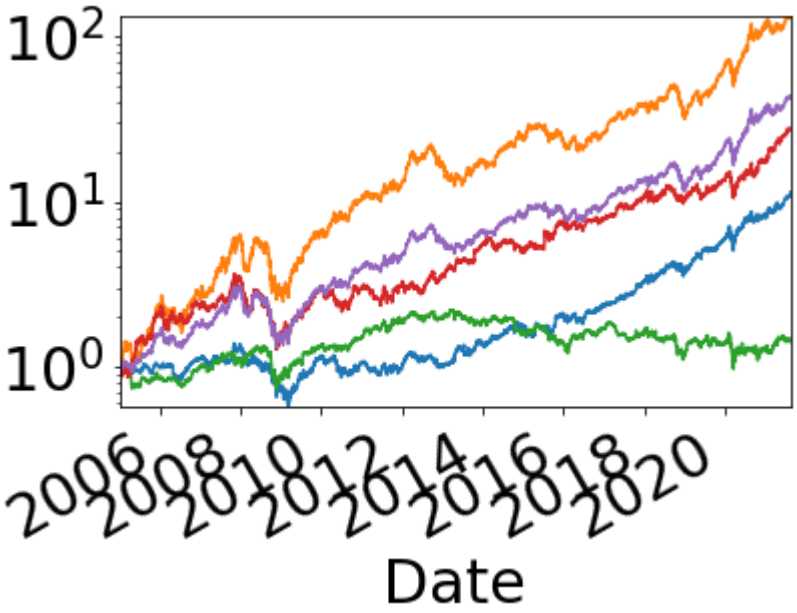
In [24]:
```
ex_return = pd.DataFrame(close_data_normalized.apply(lambda x :x.mul(portfolio_we
                         columns = ["Portfolio Return"])
close_data_normalized["Portfolio Return"] = ex_return
close_data_normalized
```

Out[24]:

Date	MSFT	AAPL	IBM	GOOG	Portfolio Return
2005-01-03	1.000000	1.000000	1.000000	1.000000	1.000000
2005-01-04	1.003740	1.010270	0.989258	0.959499	0.990692
2005-01-05	1.001496	1.019118	0.987212	0.954615	0.990610
2005-01-06	1.000374	1.019908	0.984143	0.930146	0.983643
2005-01-07	0.997382	1.094169	0.979847	0.956292	1.006923
...	...	...	...	...	...
2021-08-16	11.017203	133.713322	1.468951	27.514517	43.428498
2021-08-17	10.960359	132.890450	1.456982	27.194541	43.125583
2021-08-18	10.872476	129.501605	1.426803	27.049853	42.212684
2021-08-19	11.098354	129.802439	1.411969	27.117889	42.357663
2021-08-20	11.382199	131.120819	1.423120	27.419643	42.836445

In [25]:
```
close_data_normalized.plot.line(legend=False, logy = True)
```

Out[25]:
```
<AxesSubplot:xlabel='Date'>
```



In [26]:
```
ex_return_pct_change = ex_return.pct_change()
ex_return_pct_change.dropna(inplace = True)
ex_return_pct_change
```

Out[26]:

Date	Portfolio Return
2005-01-04	-0.009308
2005-01-05	-0.000082
2005-01-06	-0.007033
2005-01-07	0.023667
2005-01-10	0.001298
...	...
2021-08-16	0.011400
2021-08-17	-0.006975
2021-08-18	-0.021168
2021-08-19	0.003434
2021-08-20	0.011303

4187 rows × 1 columns

In [27]:
```
mean_var_df = pd.DataFrame({"mean": (ex_return.iloc[-1] / ex_return.iloc[0]) **(1
                            "sigma" : ex_return_pct_change.std()})
mean_var_df
```

Out[27]:

	mean	sigma
Portfolio Return	0.000898	0.016538

In [28]:
```
num_sims = 1000
dates = ex_return.index
monte_carlo_sim_dict = {}
mean = mean_var_df["mean"]["Portfolio Return"]
sigma = mean_var_df["sigma"]["Portfolio Return"]
run_monte_carlo(mean, sigma, num_sims, monte_carlo_sim_dict, dates)
monte_carlo_sim_df = pd.DataFrame(monte_carlo_sim_dict).add(1).cumprod()
monte_carlo_sim_df
```

Out[28]:

	0	1	2	3	4	5	6	7	
2005-01-03	1.000498	1.001900	1.005910	1.000265	0.992145	1.021637	1.003311	0.985248	0.9
2005-01-04	1.007038	1.006432	0.998970	0.986136	0.985194	1.037900	1.008015	0.947983	0.9
2005-01-05	1.011888	0.966786	0.992157	0.971938	0.995614	1.051474	1.011967	0.954512	0.9
2005-01-06	1.011141	0.968888	0.988026	0.958616	0.997806	1.048722	1.019211	0.948955	0.9
2005-01-07	1.011129	0.978532	1.017629	0.957952	0.997970	1.052190	1.047014	0.956386	0.9
...	...	...	...	...	...	...	...	...	
2021-08-16	3.012508	17.309580	18.521044	29.262178	43.720489	219.758607	47.910968	43.271354	33.
2021-08-17	3.000861	17.595527	18.845198	29.662866	42.532594	220.404568	48.317963	42.472307	33.
2021-08-18	3.054657	17.265137	18.446986	29.107579	41.582188	218.235596	48.240630	42.930195	33.
2021-08-19	3.065793	17.602233	18.689565	28.466499	40.507098	214.347665	47.931794	43.082565	32.
2021-08-20	3.088672	17.736901	18.695999	28.338543	40.518245	212.366909	47.680987	42.584755	32.

4188 rows × 1000 columns

In [29]:
```
plot_monte_carlo_sim(monte_carlo_sim_df, ex_return["Portfolio Return"], "Portfoli
```