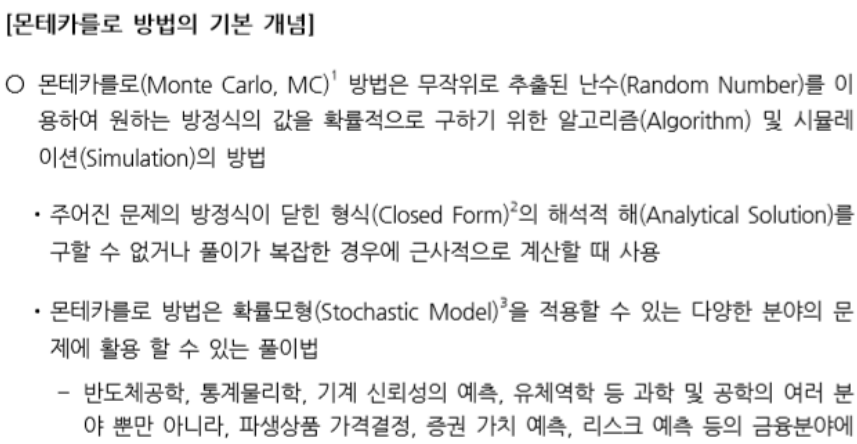
Section 6

Date	
Property	
Weeks	3주차

1. Monte Carlo Simulation

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/b47c7e62-2e8f-44a0-890d-2a74eb9e49ce/____.pdf



[몬테카를로 방법의 기본 개념]

○ 몬테카를로(Monte Carlo, MC)[1] 방법은 무작위로 추출된 난수(Random Number)를 이용하여 원하는 방정식의 값을 확률적으로 구하기 위한 알고리즘(Algorithm) 및 시뮬레이션(Simulation)의 방법

- 주어진 문제의 방정식이 닫힌 형식(Closed Form)[2]의 해석적 해(Analytical Solution)를 구할 수 없거나 풀이가 복잡한 경우에 근사적으로 계산할 때 사용
- 몬테카를로 방법은 확률모형(Stochastic Model)[3]을 적용할 수 있는 다양한 분야의 문제에 활용 할 수 있는 풀이법
  - 반도체공학, 통계물리학, 기계 신뢰성의 예측, 유체역학 등 과학 및 공학의 여러 분야 뿐만 아니라, 파생상품 가격결정, 증권 가치 예측, 리스크 예측 등의 금융분야에

쉽게 말하면 현재 시점에서 미래의 가격을 예상 해보는 기법.

예상 할 수 있게끔 만드는건 바로 난수(Random number) 생성.

* 우리는 내일 주가를 예상 할 수 없다.

그래서 모델링을 한다. (수학을 이용해서)

즉, 주가의 움직임을 어떻게 모델링 할 수 있을까?

1900년 '**루이 바슐리에**'는 박사 학위논문 <<투기이론>> 에서

"주식가격의 정규분포를 가정"

"주식가격이 시장의 모든 합리적인 예측 및 정보를 반영한다면

미래 가격변화는 브라운 운동을 따른다"고 주장.

cf) 브라운 운동 : 액체나 기체 속에서 미소입자들이 불규칙하게 운동하는 현상.

즉, 금융시장의 가격변동을 브라운 운동으로 모델링 했다.

⇒ 하지만 주식가격은 마이너스가 될 수 없어서 저 주장은 말이 안됨.

이후 미국의 경제학자 **폴 사무엘슨**이 바슐리에의 주장을 바탕으로

주가수익률과 **위너 프로세스** 도입.

주가의 움직임은 정규분포를 따른다 (바슐리에)

⇒ 주가의 수익률이 정규분포를 따른다 (사무엘슨)

$$\frac{\Delta S}{S} = \mu\Delta t + \sigma\Delta w \left(\Delta w = \sqrt{\Delta t}\varepsilon,\ \varepsilon \sim N(0,1)\right)$$

<이 식을 가지고 주가의 미래 가격 분포를 만들어 볼 수 있음. (노란색이 난수 만드는거)>

2. Lab Session-Random Walks and Monte Carlo

<윗 식 그대로 표현>

```
def gbm(n_years=10, n_scenarios=1000, mu=0.07, sigma=0.15, steps_per_year=12,s_0=100):
    # 10년 기간 동안 월 기준으로 나눠 1000개의 가격 시나리오를 만든다. (평균은 7%, 변동성은 15%라 하자. 초깃값은 100)
    dt=1/steps_per_year
    n_steps=int(n_years*steps_per_year)
    xi=np.random.normal(size=(n_steps,n_scenarios))
    rets=mu*dt + sigma*np.sqrt(dt)*xi # 식 그대로 사용 ds/s=udt+sigma*dw
    rets=pd.DataFrame(rets)
    # to prices
    prices=s_0*(1+rets).cumprod()
    return prices
```

<통계적 방법을 통해서, 주가의 수익률이 정규분포를 따른다>

```
def gbm1(n_years=10, n_scenarios=1000, mu=0.07, sigma=0.15, steps_per_year=12,s_0=100):
    dt=1/steps_per_year
    n_steps=int(n_years*steps_per_year)
    rets_plus_1=np.random.normal(loc=mu*dt,scale=(sigma*np.sqrt(dt)),size=(n_steps,n_scenarios)) #통계 분포로 표현
    rets_plus_1[0]=1
    # to prices
    prices=s_0*pd.DataFrame(rets_plus_1).cumprod()
    return prices
```

<계산 속도>

```
%timeit gbm(n_years=5,n_scenarios=1000)
```

```
7.89 ms ± 133 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

```
%timeit gbm1(n_years=5,n_scenarios=1000)
```

```
7.06 ms ± 351 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

당연히 내장한거 사용하는게 더 빠름.

3. Analyzing CPPI strategies

<몬테카를로 시뮬레이션 하여 나온 값들의 움직임에 CPPI 전략을 이용해본다>

일반적인 몬테카를로 시뮬레이션 형태 //

오른쪽 히스토그램은 왼쪽의 시뮬레이션 가격의 최종값이 몇개 나왔는가 보여주는거.

<원가격의 70% 미만이 안되도록 floor를 설정한 형태>

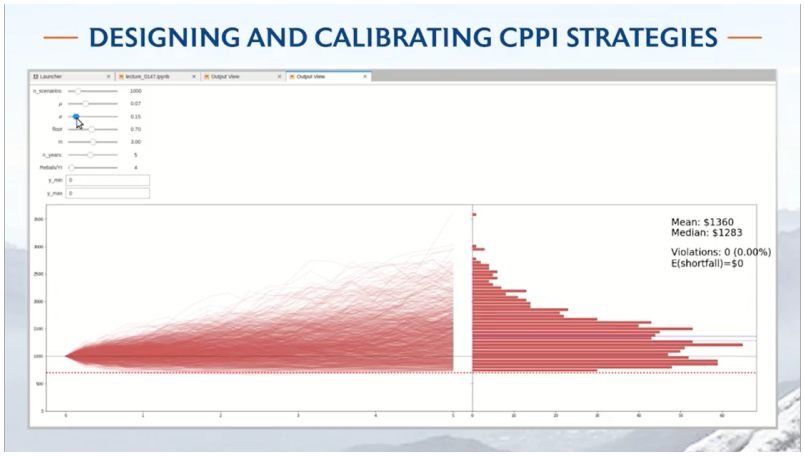
4. Lab Session-Installing IPYWIDGETS

<하란대로 따라 하면 됩니다.>

저게 일일이 손으로 변수값 바꾸기 힘드니까, 아예 콘솔 형태로 왔다갔다 할수 있게끔 하는

라이브러리?를 설치하는거 같음.

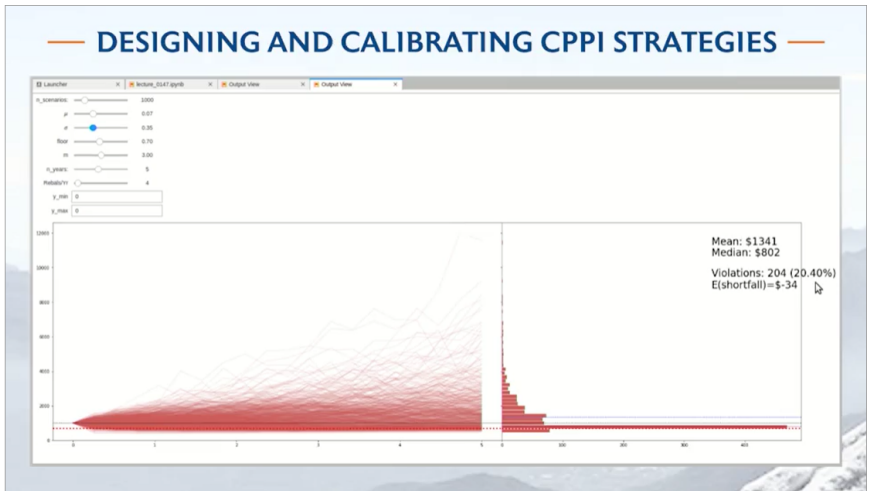
5. Designing and calibrating CPPI strategies



1) floor 설정한 상태 = violation 없는 상태

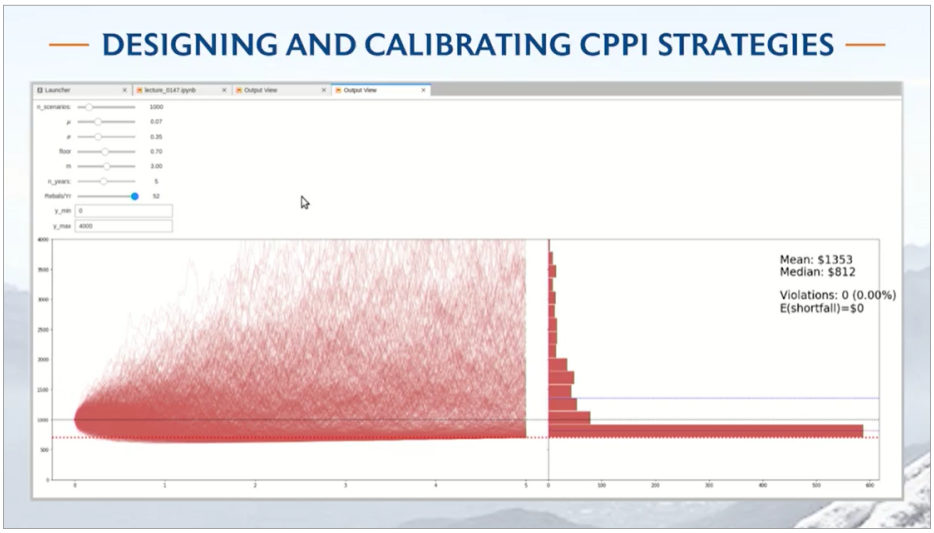
왜 violation 이 없을까? 그건 우리가 처음부터 변동성을 15%로 설정했기 때문!

그렇다면 변동성을 더 높여보자 (이게 더 현실적이니까)



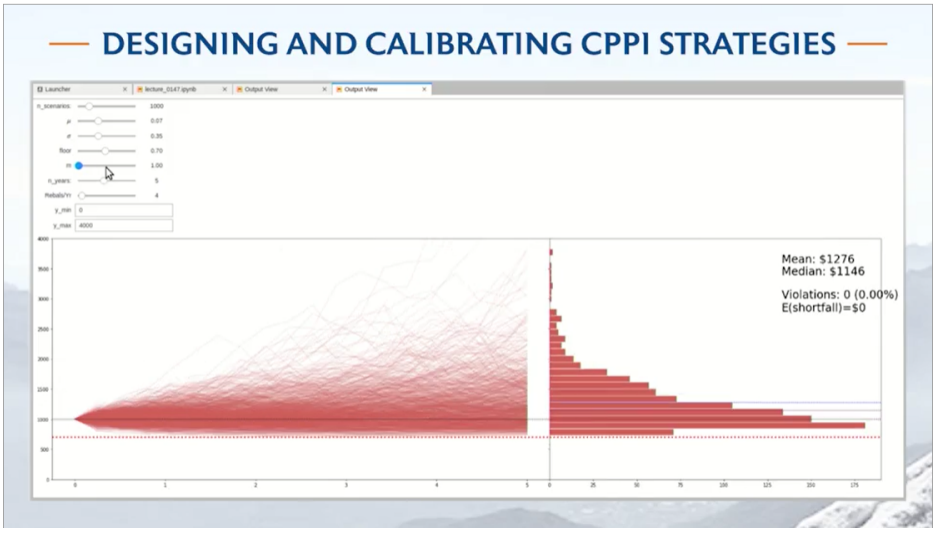
2) 변동성을 높였을 때 :: floor보다 더 내려간다.

violation 줄이는 방법 :: 그건 매주 rebalancing을 하는 것

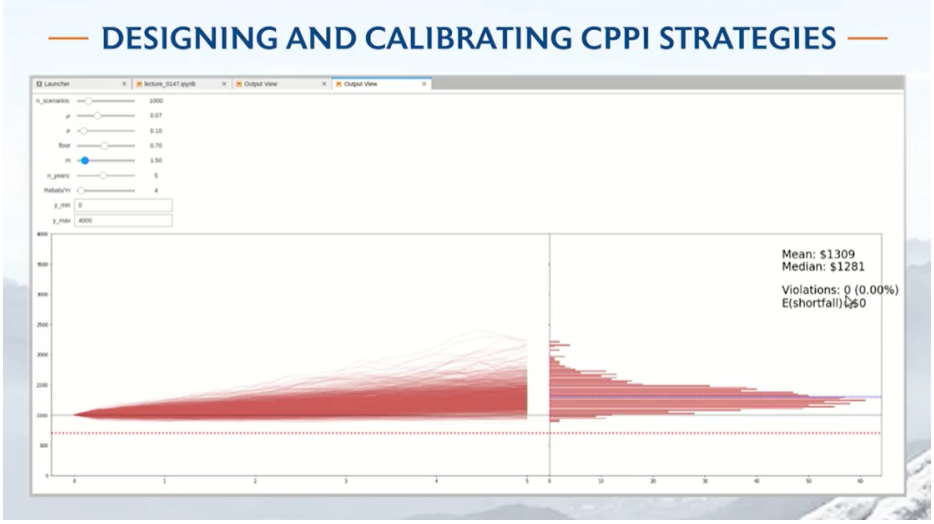


3) rebalancing 횟수를 늘린다.

다른 방법으로는 rebalancing은 4분기 마다 + CPPI 모델 사용하기



4-1) m(쿠션에 곱해주는 승수)을 줄인다.



4-2) 변동성을 줄인다.

이렇게 되면 10%의 변동성은 바닥을 통과할 일이 없게 된다.

그러나 내 위치가 floor보다 약간 위라고 하면 이 10%의 변동성도 위험할 수 있다.

결론 ::

변동성이 작은 시장이라면 큰 m을 쓰고, 변동성이 큰 시장이라면 작은 m을 쓰자

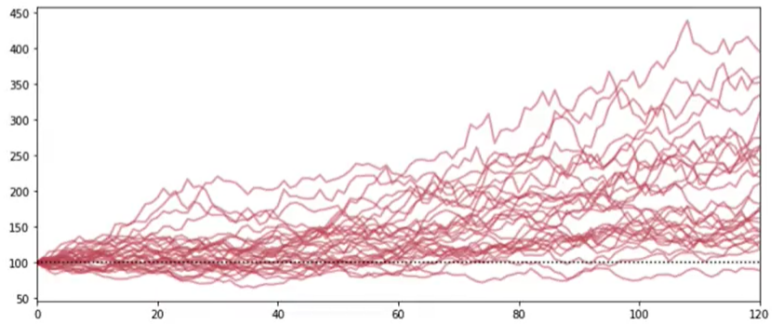
6. Lab Session - interactive plots of monte Carlo Simulations of CPPI and GBM-Part1

```
def gbm(n_years = 10, n_scenarios=1000, mu=0.07, sigma=0.15, steps_per_year=12, s_0=100.0, prices=True):
    """
    Evolution of Geometric Brownian Motion trajectories, such as for Stock Prices through Monte Carlo
    :param n_years:  The number of years to generate data for
    :param n_paths: The number of scenarios/trajectories
    :param mu: Annualized Drift, e.g. Market Return
    :param sigma: Annualized Volatility
    :param steps_per_year: granularity of the simulation
    :param s_0: initial value
    :return: a numpy array of n_paths columns and n_years*steps_per_year rows
    """
    # Derive per-step Model Parameters from User Specifications
    dt = 1/steps_per_year
    n_steps = int(n_years*steps_per_year) + 1
    # the standard way ...
    # rets_plus_1 = np.random.normal(loc=mu*dt+1, scale=sigma*np.sqrt(dt), size=(n_steps, n_scenarios))
    # without discretization error ...
    rets_plus_1 = np.random.normal(loc=(1+mu)**dt, scale=(sigma*np.sqrt(dt)), size=(n_steps, n_scenarios))
    rets_plus_1[0] = 1
    ret_val = s_0*pd.DataFrame(rets_plus_1).cumprod() if prices else rets_plus_1-1
    return ret_val
```

```
def show_gbm(n_scenarios, mu, sigma):
    """
    Draw the results of a stock price evolution under a geometric brownian motion model
```

```
    """
    s_0=10000
    prices=erk.gbm(n_scenarios=n_scenarios, mu=mu, sigma=sigma, s_0=s_0)
    ax=prices.plot(legend=False, color="indianred", alpha=0.5, linewidth=2, figsize=(12,5))
    ax.axhline(y=s_0, ls=":", clolor="black")
    #draw a dot at the origin
    ax.plot(0,s_0,marker='o', color='darkred', alpha=0.2)
```

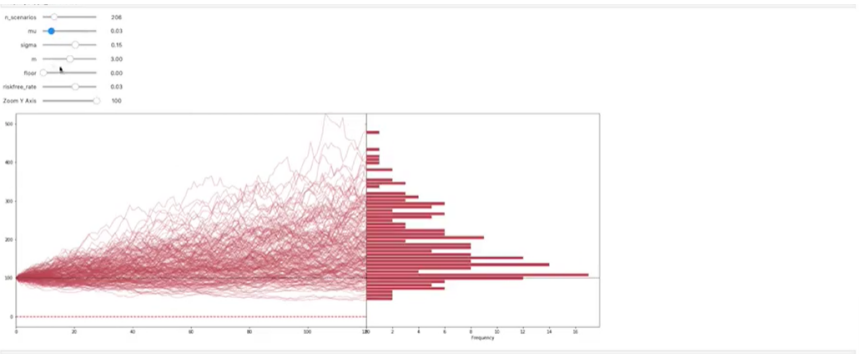
```
show_gbm(30, 0.07, 0.15)
```



7. Lab Session - interactive plots of monte Carlo Simulations of CPPI and GBM-Part2

```
import matplotlib.pyplot as plt
import pandas as pd

def show_cppi(n_scenarios=50, mu=0.07, sigma=0.15, m=3, floor=0, riskfree_rate=0.03, y_max=100):
    """
    plot the results of a Monte Carlo Simulation of CPPI
    """
    start=100
    sim_rets=erk.gbm(n_scenarios=n_scenarios, mu=mu, sigma=sigma, prices=False, steps_per_year=12)
    risky_r=pd.DataFrame(sim_rets)

    #run the "back test"
    btr=erk.run_cppi(risky_r=pd.DataFrame(risky_r), riskfree_rate=riskfree_rate, m=m, start=start, floor=floor)
    wealth=btr["wealth"]

    #calculate terminal wealth stats
    y_max=wealth.values.max()*y_max/100
    terminal_wealth = wealth.iloc[-1] #last price를 의미.

    #plot # one row two colums, y축 공유
    fig, (wealth_ax, hist_ax) = plt.subplots(nrows=1, ncols=2, sharey=True, gridspec_kw={'width_ratios':[3,2]}, figsize=(24,9))
    plt.subplot_adjust(wspace=0.0) # 두 그래프 간 간격 = 0
    wealth.plot(ax=wealth_ax, legend=False, alpha=0.3, color="indianred")
    wealth_ax.axhline(y=start, ls=":", color="black")
    wealth_ax.axhline(y=start*floor, ls="--", color="red")
    wealth_ax.set_ylim(top=y_max)

    terminal_wealth.plot.hist(ax=hist_ax, bins=50, ec='w', fc='indianred', orientation='horizontal')  #ec=edge color
    hist_ax.axhline(y=start, ls=":", color="black")

cppi_controls = widgets.interactive(show_cppi,
                                    n_scenarios=widgets.IntSlider(min=1, max=1000, step=5, value=50)
                                    mu=(0.,+.2, .01),
                                    sigma=(0,.3,.05),
                                    floor=(0, 2, .1),
                                    m=(1,5,.5),
```

```
                                riskfree_rate=(0,.05,.01),
                                y_max=widgets.IntSlider(min=0, max=100, step=1, value=100)
)
display(cppi_controls)
```



앞과 비슷한게 나온다.

8. Merits and limits of portfolio insurance strategies

😋

포트폴리오 보험이 포트폴리오의 가치가 하락될 것을 우려하여 헷지를 하는거라면,

현 포트폴리오의 가치와 반대방향을 가지는 선물을 매도하거나, 풋옵션을 매수하면 될듯.

따라서 한방향만 가지던 포트폴리오가 잠잠해 질수 있는 좋은 상품.

Merit ⇒ 1. 가격이 일정 범위 내에서 하락하더라도 그만큼의 상쇄분을 보상 받을 수 있다.

Limits ⇒1. 올해 3월과 같은 panic장이 나오게 되면 (서킷브레이커, 사이드카, 마이너스 유가 등등) 소용 없음. 팔자팔자 하는 사람만 있을테니까