

# Section 5

<input type="checkbox"/> Date	
<input type="checkbox"/> Property	
<input checked="" type="checkbox"/> Weeks	3주차

## 2. Limits of diversification

risk management(hedging & insurance)에 대해 얘기할 것

- Diversification 포트폴리오 다각화  
: 포트폴리오의 특정 위험을 제거함으로써 프리미엄 얻음

The slide has a blue header bar with the text 'LIMITS OF DIVERSIFICATION'. Below it, the main title is 'LIMITS OF DIVERSIFICATION'. The central text reads:  
DIVERSIFICATION WILL FAIL YOU WHEN  
YOU NEED IT THE MOST  
TRYING TO ADDRESS THIS LIMITATION BY "IMPROVING"  
DIVERSIFICATION SIMPLY WON'T WORK  
ALMOST BY DEFINITION ONE CANNOT  
DIVERSIFY AWAY SYSTEMATIC RISK

On the right side of the slide, there is a photograph of a man in a suit speaking. He is gesturing with his hands while holding a small object. In the bottom left corner, there is a logo for EDHEC-RISK.

At the bottom of the slide, there is a dark bar with the text 'the risk factor that'.

시장 상황에 따라 평균적으로 적용 가능, but 체계적인 위험을 처리하는 데 적용 불가능

ex) 2008년, 단일 주식 포트폴리오를 보유하든 여러 주식에 걸쳐 다양한 포트폴리오를 보유하든 모두 20 % 감소  
→ 체계적인 위험은 모든 자산에 동시에 영향을 미치는 위험 요소를 의미. 따라서 포트폴리오 다각화의 한계 있음.

- liberty hedging portfolio  
: 체계적인 위험 요소가 있으면 부채가 증가하기 때문에 해정을 통해 극복 (위험 분산)

## DIVERSIFICATION VERSUS HEDGING

HEDGING (I.E., AVOIDING RISK TAKING)  
IS THE ONLY EFFECTIVE WAY TO  
OBTAIN DOWNSIDE PROTECTION

THE PROBLEM WITH HEDGING  
IS THAT INVESTORS GIVE UP ON THE  
UPSIDE AT THE SAME TIME AS THEY  
GIVE UP ON THE DOWNSIDE



[사진]

hedging is the one effective way to obtain downside protection

the problem with hedging is that investors give up on the upside at the same time as they give up on the downside

포트폴리오의 부채 = 자산 은 동일하기 때문에 큰 영향 x

but 해지는 양면적(symmetric) 이기 때문에 단점을 보완하는 동시에 또 그만큼 상승을 기대할 수 없다 ( $(-) + (+) = 0$ )

100퍼 자금 조달 위해 해지하면 비싸기 때문에 부자들만 할 수 있다..😢(소수의 거액투자자) > 보험(insurance)을 통해 비용 위험 감소 가능

- Hedging vs Insurance

: 위험을 무시하고, 수익(성능) 추구형 포트폴리오에 투자할 것인가?

⇒ Insurance 를 통해 동적 해징(dynamic hedging)으로 위험 완화 + 수익 보장 가능

<포트폴리오 보험 portfolio insurance>

위험 없을 때 → hedging 해지 전략 그대로 써서 수익 추구 포트폴리오

위험 있을 때 → 안전 추구 포트폴리오

두 가지 경우에 대비한 할당 가능

다음 시간 시간 경과 & 예산 소비에 따른 할당 방법 알려줄게!

### 3. Lab session- Limits of Diversification-Part1

목표 : 시장의 주식에 대한 상관 관계 측면에서 시장이 하락을 볼 것

→ 가중 가중 시장 지수 (cap-weighted market index) 을 통해

총 시장 규모. 해당 포트폴리오의 모든 구성 요소, 해당 포트폴리오의 모든 개별 주식의 크기&가치에 비례하여 포트폴리오를 구축하고 시가 총액을 기준으로 무게를 측정하면 → 유동성에 대한 미세한 수정으로 플로트 조정 가능

```

import edhec_risk_kit as erk
import pandas as pd
import numpy as np

ind_return = erk.get_ind_returns()
ind_nfirms = erk.get_ind_nfirms()
ind_size = erk.get_ind_size()

# 개별 산업의 시가총액 계산
ind_mktcap = ind_nfirms * ind_size # 산업 내 기업 수 x 크기
# 전체 산업의 시가총액 계산
tot_mktcap = ind_mktcap.sum(axis="columns") # 개별 산업 시가총액의 합

# 월별 개별 산업의 비중 구하기 (개별 market cap의 비중)
ind_capweight = ind_mktcap.divide(tot_mktcap, axis = "rows") # 개별 시총/전체 시가총액

# 전체 시장 수익률은 개별 산업의 가중 평균
total_market_return = (ind_capweight * ind_return).sum(axis="columns") # 개별 산업의 market cap 비중 x 개별 산업 수익

# 전체 시장 수익에서 파생 된 자산 전체 시장 지수 구하기
total_market_index = erk.drawdown(total_market_return).Wealth

```

## 4. Lab Session-Limits of diversification - Part 2

```
total_market_index["1980":].plot(figsize=(12,6))
total_market_index["1980":].rolling(window=36).mean().plot(figsize=(12,6))
#여기서 rolling 함수는 이동평균을 계산해주는 함수로, 위 코드에서는 1980년 이후 전체 시장 지수의 이동평균을 계산한다.

tmi_tr36rets = total_market_return.rolling(window=36).aggregate(erk.annualize_rets, periods_per_year=12)
#total market index trailing 36-months returns
#aggregate는 각각의 window에 적용된다.

ts_corr = ind_return.rolling(window=36).corr(); ts_corr.tail() #multi Index 발생

ind_tr36corr = ts_corr.groupby(level='date').apply(lambda cormat: cormat.values.mean())
# cf)apply(lambda x: pd.series({'key':function(x)}): lambda 함수 적용한 컬럼들로 Pd.DataFrame 생성

tmi_tr36rets["2007":].plot(label = "Tr36 Mo Rets", figsize = (12,6), legend = True)
ind_tr36corr["2007":].plot(label = "Tr36 Mo Corr", figsize = (12,6), legend = True, secondary_y = True)
# 36 months trailing mean 과 correlation이 반대로 움직이는 것을 알 수 있다.

tmi_tr36rets.corr(ind_tr36corr)
#compute correlation of average correlation of industries and returns -> 이게 핵심
#when market is crashing diversification is not a good strategy.
```



### ▼ Reference

- <https://www.thebalance.com/rolling-returns-versus-average-annual-returns-2388654>
- <https://www.r-bloggers.com/tidy-time-series-analysis-part-3-the-rolling-correlation/>

## 5. An introduction to CPPI - Part 1

CPPI에 알아보기에 앞서 포트폴리오 보험에 대해 살펴보자.

포트폴리오 보험은 동적자산배분전략(dynamic allocation strategies) 중 하나이다.

### ▼ 포트폴리오 보험

포트폴리오 보험 전략은 투자기간 말에 최저유지수준(floor) 이상으로 포트폴리오의 가치를 보호하면서 주가가 상승하는 경우에는 이에 편승하여 이익을 획득하고자 하는 자산 운용기법이다.

포트폴리오 보험 전략은 다양한 방법이 있다. 풋옵션을 이용한 보호풋 전략과 합성풋옵션을 이용한 동적자산배분전략 등이 있다.

포트폴리오보험  
= 기초자산 1개 보유 + 풋옵션 1개 매입  
= 기초자산 1개 보유 + delta 풋옵션 개의 기초자산을 공매 + 무위험대출  
= 기초자산(1+ delta 풋옵션)개 매입 + 무위험대출  
= 기초자산 delta 콜옵션 개 매입 + 무위험대출

- 풋옵션을 이용한 보호풋전략

기초자산의 가격이 행사가격 이하가 된 경우 그 손실을 보전해 주는 보험의 효과를 제공한다. ( $S + P$ ) 단, 보유한 자산에 대한 풋옵션이 존재하지 않거나 투자기간과 풋옵션의 만기가 일치하지 않는 경우가 많아 현실적이지 않다.

- 합성풋옵션을 이용한 동적자산배분전략

위에서 살펴본 보호풋 전략의 한계를 보완하고자 합성풋옵션을 풋옵션 대신 이용하는 전략이 있다. 풋옵션 델타만큼 기초자산(주식)을 공매하고 무위험대출(채권 보유) 함으로써 풋옵션을 매입한 효과를 얻을 수 있다.

이 때, 시간의 흐름에 따라 풋옵션 델타가 변하기 때문에 기초자산 투자액과 무위험 대출액을 지속적으로 조정해야 한다. 이렇게 지속적으로 조정해야하는 전략을 **동적자산배분전략**이라고 한다.

reference) 종길이 형님의 재무관리연습 6판 / <https://blog.naver.com/headformat/50013208894>

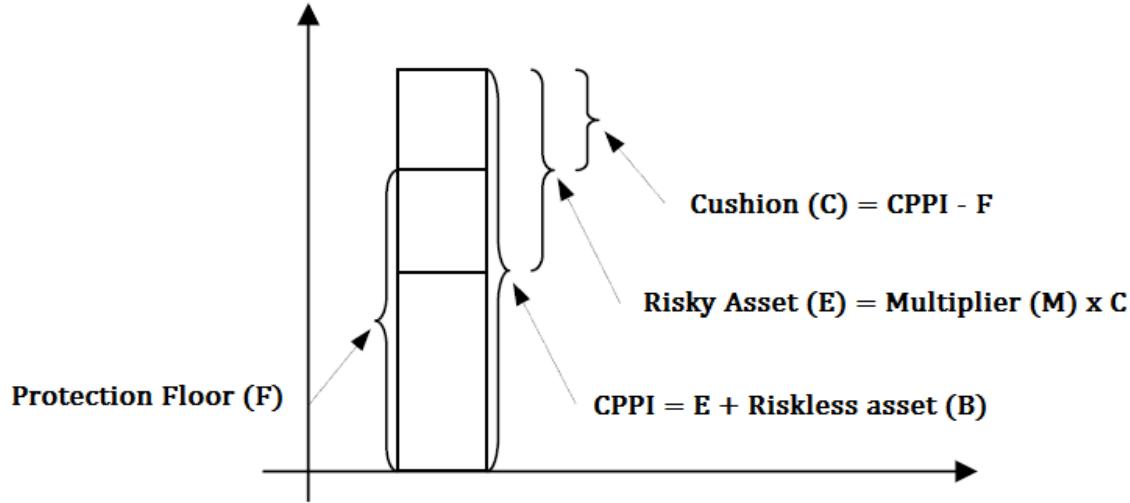
### 1. 고정비율 포트폴리오 보험 전략 CPPI(Constant Proportion Insurance)

위에서 자세히 살펴본 포트폴리오 보험 전략 중 하나가 CPPI이다.

주식과 채권으로 포트폴리오를 구성하는 전략으로 유지하고자 하는 포트폴리오의 최저수준(floor)를 설정하여 포트폴리오에서 주식 투자액이 차지하는 비중(Risky Asset(E))을 Cushion에 승수(m)배로 일정하게 유지하는 전략이다.

최저수준만 초기에 설정하면 되는 간단한 전략이다.

주식 가격이 상승하면 포트폴리오 가치가 증가하고 cushion 또한 증가하므로 주식에 투자하는 비중이 늘어나는 반면, 주식 가격이 하락하면 주식에 투자하는 비중은 줄어든다. 초기에 정한 Protection Floor는 시간 경과에 따라 Rf(무위험 이자율)로 증가한다.



- GAP Risk

GAP Risk란, Rebalancing date 사이에서 위험 자산 가치의 급격한 하락으로 최저수준(Floor)보다 포트폴리오의 가치가 낮아지게 되는 위험을 말한다.

이는 Cushion에 승수(M)를 곱한 만큼 위험자산에 투자하기 때문이다. 승수의 역수를 취한 값을 GAP Size라고 한다. 예를 들어 승수가 4인 경우 포트폴리오의 가치는 25%까지 떨어질 수 있음을 의미한다.

<http://www.deltaquants.com/Introduction-to-risks-in-CPPI-products>

## 6. An introduction to CPPI - Part 2

지난시간에 MDD(maximum drawdown, 전고점대비하락비율)를 배웠다.

MDD는 위험관리 기법으로서, 심리적으로 얼마만큼의 리스크를 감당할수있는지 판단 기준이 된다.

CPPI(Constant proportion portfolio insurance strategies)는 mdd의 마지노선을 보호하기 위한 기법이다.

$V(t)$  : t 시점에서의 포트폴리오 가치

$M(t)$  : 0~t 시간에서 포트폴리오가 가질수있는 최대 가치

$V(t)$ 와  $M(t)$ 의 관계



예시) MDD를 20퍼센트 아래로 한다 =  $M(t)$ 가 항상 80% 이상으로 유지한다.

즉, mdd를 a라고 생각할때, 우리는 포트폴리오의 가치를  $1-a$  만큼 지키려고 하는 것임.

결론:

투자자들은 이익이 극대화되기를 바라면서도 최소한 원금만이라도 건질수 있길 바람.

그 전략으로 cippi가 고안되었는데

투자기간을 기준으로

투자 상품의 "가격이 얼마나 떨어질 수 있느냐"를 추정한 후

그 범위를 전제로 일정액의 배수에 해당하는 액수를 직접 투자하는 전략

그 일정액의 배수를 어떻게 정하느냐? → MDD의 하한과, cap과의 차이를 생각함

$$F_t \leq A_t \leq T_t \Rightarrow E_t = m(A_t - F_t)$$

$$T_t \leq A_t \leq C_t \Rightarrow E_t = m(C_t - A_t)$$

floor값과 cap값과 현재 자산가치 차이 중 작은것으로 계산하면 됨.

여기서  $T(t)$ 는 임계값으로,  $(F(t)+C(t))/2$  로 많이 계산함.

예시)

What is the initial allocation to the risky asset for a portfolio starting at \$100, with a multiplier equal to 4, a wealth preservation floor set at 90% and a cap set at 105%?

- 20%

**맞습니다**

The portfolio initial value is closer to the cap than it is to the floor. In this case, the investment in risky asset is  $4 \times (105\%-100\%) = 20\%$ .

## 7. Lab session-CPPI and Drawdown Constraints-Part1

```
In [7]: def compound1(r):
    return np.expm1(np.log1p(r).sum()) #
    # = np.exp(np.log(1+r).sum()-1)
def compound2(r):
    return (r+1).prod()**1

# 결과적으로 compound는 루프의 형태로 사용될 것이기에 효율적으로 짜야한다.

compound1(ind_return[["Steel", "Fin", "Beer"]])
```

```
Out[7]: Steel      435.580153
Fin        5778.433650
Beer       35820.527452
dtype: float64
```

```
In [8]: compound2(ind_return[["Steel", "Fin", "Beer"]])
```

```
Out[8]: Steel      435.580153
Fin        5778.433650
Beer       35820.527452
dtype: float64
```

They both produce the same result, but are they identical? The second one might appear a bit more clear to some programmers, which of these is more efficient? We can find out by using the %timeit magic.

```
In [9]: %timeit compound1(ind_return)
744 µs ± 18.6 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

```
In [10]: %timeit compound2(ind_return)
499 µs ± 6.79 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

```
#####
이게 우리가 파이썬에서 루프를 사용하지 말아야 하는 이유
#####
Astonishingly, on my computer, compound1 took around 0.8 millisecond, while compound2 took more than 8 ms per loop, which is an entire order of magnitude! Let's add this to our toolkit, which we'll use later:
```

```
def compound(r):
    """
    returns the result of compounding the set of returns in r
    """
    return np.expm1(np.log1p(r).sum())
```

And now, back to CPPI ...

```
for step in range(n_steps):
    # 1. Compute the cushion (asset value minus floor)
    cushion = (account_value - floor_value)/account_value

    # 2. Compute the allocation (based on the multiplier)
    risky_w = m*cushion
    risky_w = np.minimum(risky_w, 1) # 1 0/상 까지 -> 1
    risky_w = np.maximum(risky_w, 0) # 0 0/상 까지 -> 0
    safe_w = 1-risky_w
    risky_alloc = account_value*risky_w
    safe_alloc = account_value*safe_w

    # 3. Compute the new asset value
    account_value = risky_alloc*(1+risky_r.iloc[step]) + safe_alloc*(1+safe_r.iloc[step])

    # save the histories for analysis and plotting
    cushion_history.iloc[step] = cushion
    risky_w_history.iloc[step] = risky_w
    account_history.iloc[step] = account_value
    risky_wealth = start*(1+risky_r).cumprod()
```

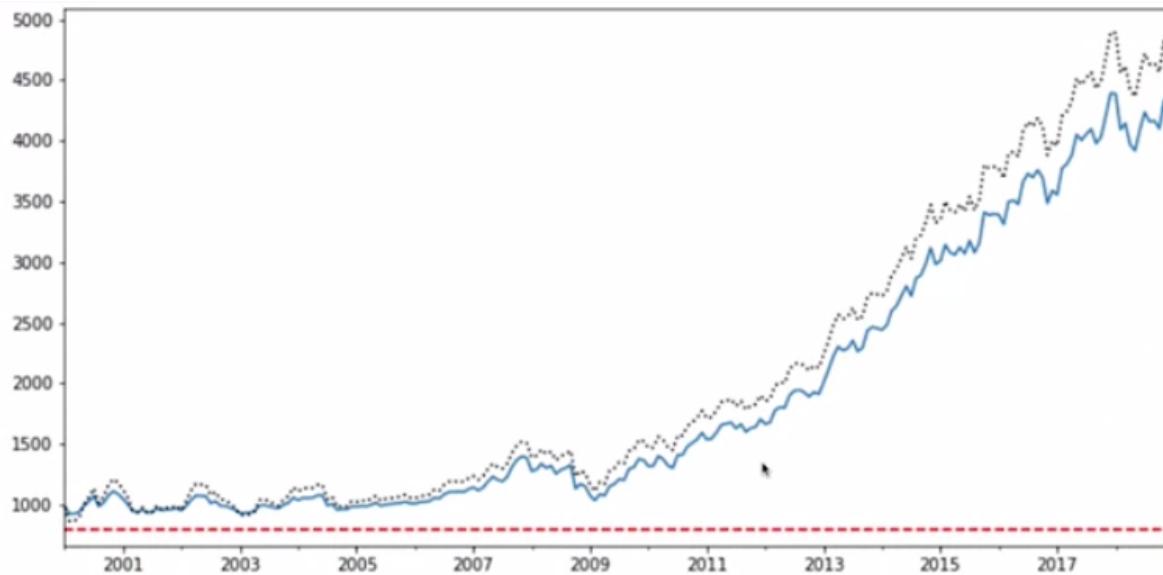
## 8. Lab session-CPPI and Drawdown Constraints-Part2

오개미 화이팅

네....

```
ax = account_history["Beer"].plot(figsize=(12,6))
risky_wealth["Beer"].plot(ax=ax, style="k:")
ax.axhline(y=floor_value, color='r', linestyle="--")
```

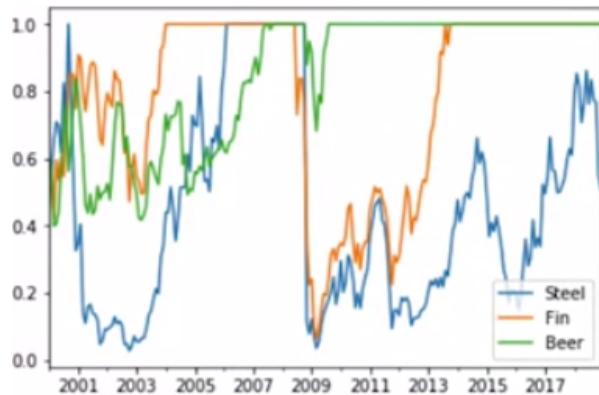
plot 을 해봅시다



예쁜 결과가 나오네요

```
risky_w_history.plot()
```

CPPI에 사용한 weights를 plot 해봅시다

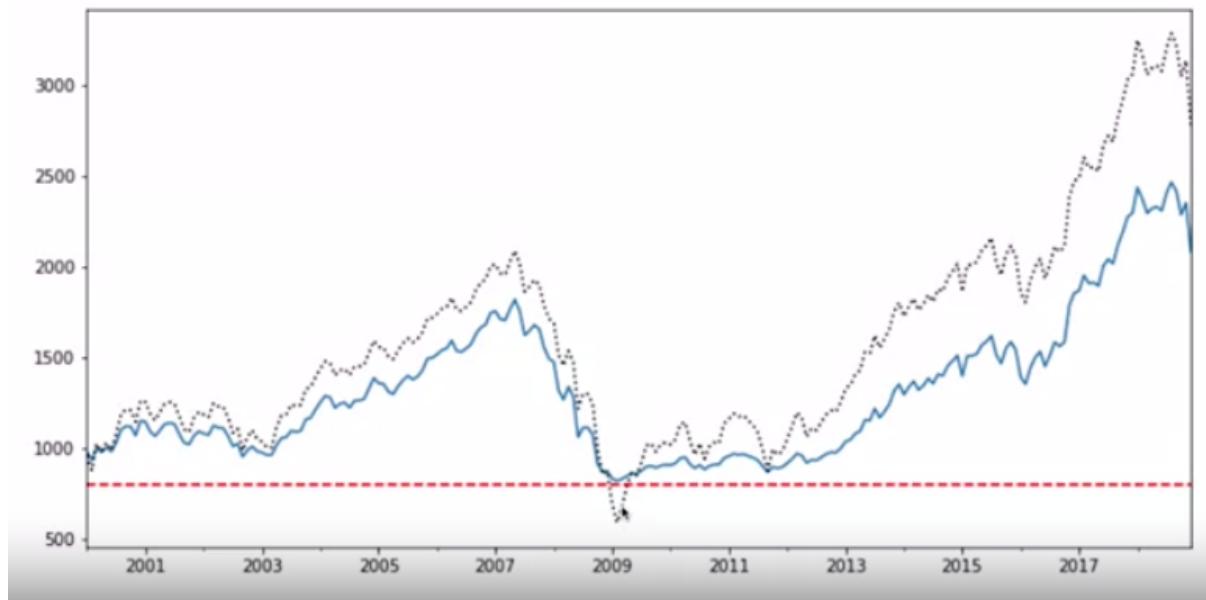


예쁘네요

특정 순간이 지나면 쿠션이 너무 커져 weight가 100%를 바꿔네요

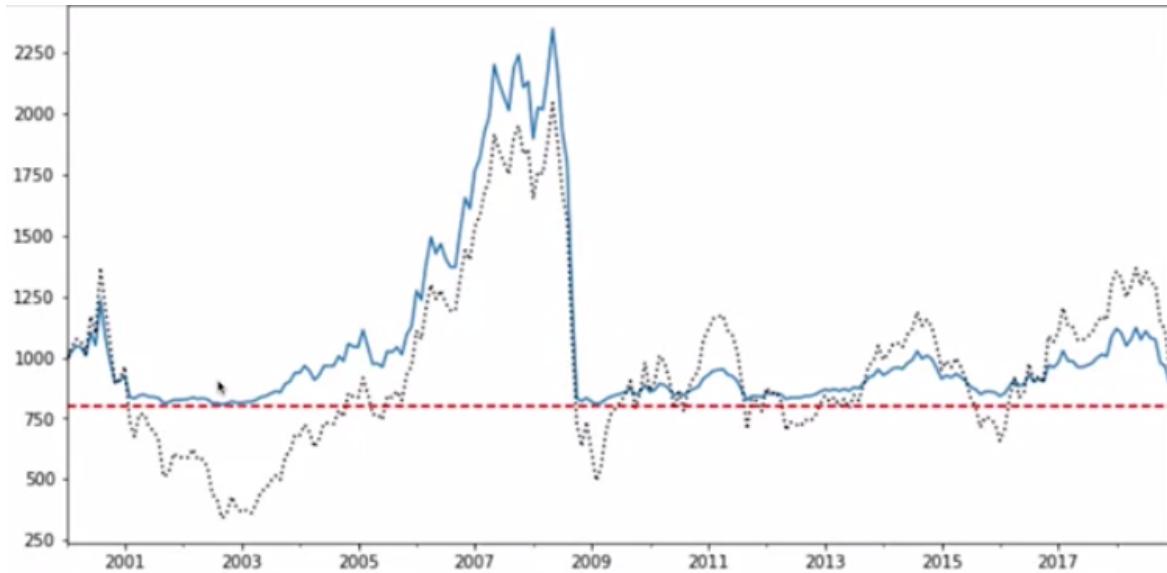
그래서 real-life에서는 floor을 지속 적으로 업데이트 해야 되지요

그게 Drawdown 을 이용하는 거지요



CPPI를 Finance 에 적용해 보면 2009년에 확인 할수 있듯이 망했네요

단점으로는 증가 구간에서 증가 폭이 별루에요....



Steel에 적용해 봅시다..."Oh My God that is brutal"

더 망했네요.....

역시 술은 좋은거에요 인정

그래도 손해는 아니니까 행복한거...

이번에는 CPPI를 자동 업데이트 시킬꺼래요

재밌겠네요

```

risky_w = np.minimum(risky_w, 1)
risky_w = np.maximum(risky_w, 0)
safe_w = 1-risky_w
risky_alloc = account_value*risky_w
safe_alloc = account_value*safe_w
# recompute the new account value at the end of this step
account_value = risky_alloc*(1+risky_r.iloc[step]) + safe_alloc*(1+safe_r.iloc[step])
# save the histories for analysis and plotting
cushion_history.iloc[step] = cushion
risky_w_history.iloc[step] = risky_w
account_history.iloc[step] = account_value
risky_wealth = start*(1+risky_r).cumprod()
backtest_result = {
    "Wealth": account_history,
    "Risky Wealth": risky_wealth,
    "Risk Budget": cushion_history,
    "Risky Allocation": risky_w_history,
    "m": m,
    "start": start,
    "floor": floor,
    "risky_r":risky_r,
    "safe_r": safe_r
}
return backtest_result

```

I

이전 시간에 했던 코드에다 risky\_weakth = 일부분을 더해 줍니다.

그리고 예쁘게 사전으로 포장을 해줍니다. 정말 아름답네요

```
def summary_stats(r, riskfree_rate=0.03):
    """
    Return a DataFrame that contains aggregated summary stats for the returns in the columns of r
    """
    ann_r = r.aggregate(annualize_rets, periods_per_year=12)
    ann_vol = r.aggregate(annualize_vol, periods_per_year=12)
    ann_sr = r.aggregate(sharpe_ratio, riskfree_rate=riskfree_rate, periods_per_year=12)
    dd = r.aggregate(lambda r: drawdown(r).Drawdown.min())
    skew = r.aggregate(skewness)
    kurt = r.aggregate(kurtosis)
    cf_var5 = r.aggregate(var_gaussian, modified=True)
    hist_cvar5 = r.aggregate(cvar_historic)
    return pd.DataFrame({
        "Annualized Return": ann_r,
        "Annualized Vol": ann_vol,
        "Skewness": skew,
        "Kurtosis": kurt,
        "Cornish-Fisher VaR (5)": cf_var5,
        "Historic CVaR (5)": hist_cvar5,
        "Sharpe Ratio": ann_sr,
        "Max Drawdown": dd
    })
```

이 코드는 편의를 위해 추가 되었습니다.

Back test 결과를 한눈에 보여 주는 거에요

	Annualized Return	Annualized Vol	Skewness	Kurtosis	Cornish-Fisher VaR (5%)	Historic CVaR (5%)	Sharpe Ratio	Max Drawdown
Steel	-0.002790	0.312368	-0.326334	4.144381	0.150139	0.208117	-0.102567	-0.758017
Fin	0.055166	0.192909	-0.533218	4.995534	0.091224	0.132175	0.126718	-0.718465
Beer	0.080598	0.138925	-0.493545	4.173881	0.063015	0.091442	0.354314	-0.271368

이렇게 유용하지요

```
btr = erk.run_cppi(risky_r)
erk.summary_stats(btr["Wealth"].pct_change().dropna())

-----
AttributeError                                 Traceback (most recent call last)
<ipython-input-23-7bdaeb9cbb90> in <module>
      1 btr = erk.run_cppi(risky_r)
----> 2 erk.summary_stats(btr.Wealth.pct_change().dropna())

AttributeError: 'dict' object has no attribute 'Wealth'
```

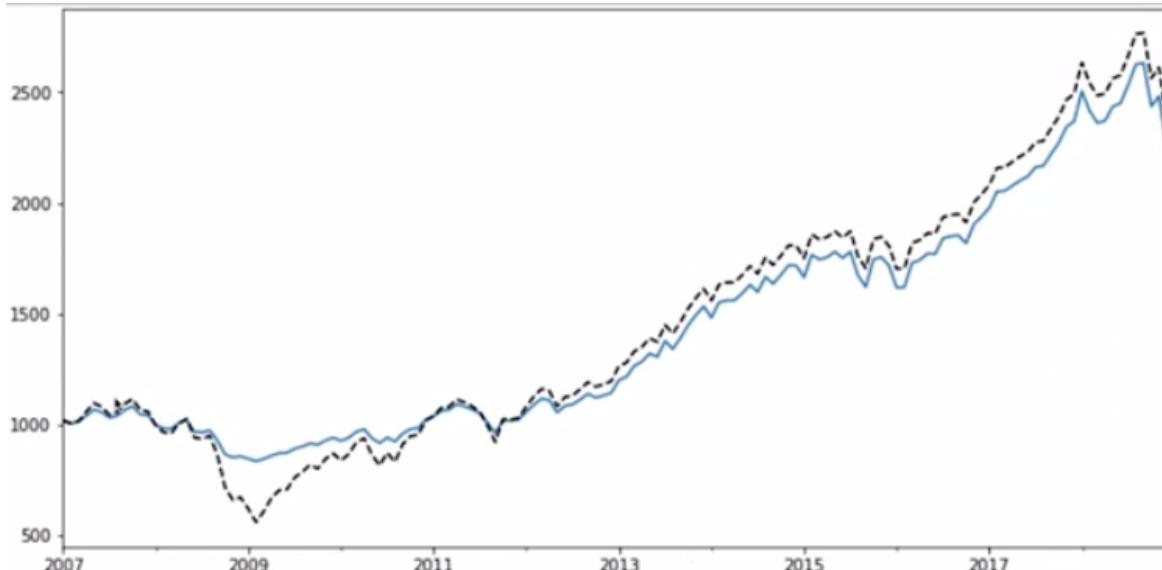
이런 오류가 났네요....

```
btr = erk.run_cppi(risky_r)
erk.summary_stats(btr["Wealth"].pct_change().dropna())
```

	Annualized Return	Annualized Vol	Skewness	Kurtosis	Cornish-Fisher VaR (5%)	Historic CVaR (5%)	Sharpe Ratio	Max Drawdown
Steel	-0.005167	0.174180	-1.995143	17.110190	0.091995	0.130153	-0.196750	-0.655198
Fin	0.040894	0.131678	-0.946504	6.051414	0.065535	0.091621	0.080352	-0.549673
Beer	0.075544	0.115462	-0.669250	4.760879	0.052923	0.074908	0.383772	-0.259582

```
btr = erk.run_cppi(tmi_return["2007":])
ax = btr["Wealth"].plot(figsize=(12,6), legend=False)
btr["Risky Wealth"].plot(ax=ax, style="k--")
```

또 plot을 해봅시다



```
erk.summary_stats(btr["Risky Wealth"].pct_change().dropna())
```

	Annualized Return	Annualized Vol	Skewness	Kurtosis	Cornish-Fisher VaR (5%)	Historic CVaR (5%)	Sharpe Ratio	Max Drawdown
R	0.073411	0.150463	-0.734939	4.523488	0.071592	0.096315	0.280618	-0.499943

```
erk.summary_stats(btr["Wealth"].pct_change().dropna())
```

	Annualized Return	Annualized Vol	Skewness	Kurtosis	Cornish-Fisher VaR (5%)	Historic CVaR (5%)	Sharpe Ratio	Max Drawdown
R	0.069416	0.100381	-0.588292	3.740932	0.045678	0.062953	0.382052	-0.229683

Drawdown이 줄었쥬

행복하세요

Drawdown constraint

```
# set up the CPPI parameters
dates = risky_r.index
n_steps = len(dates)
account_value = start
floor_value = start*floor
peak = start

if isinstance(risky_r, pd.Series):
    risky_r = pd.DataFrame(risky_r, columns=["R"])

if safe_r is None:
    safe_r = pd.DataFrame().reindex_like(risky_r)
    safe_r.values[:] = riskfree_rate/12 # fast way to set all values to a number

# set up some DataFrames for saving intermediate values
account_history = pd.DataFrame().reindex_like(risky_r)
risky_w_history = pd.DataFrame().reindex_like(risky_r)
cushion_history = pd.DataFrame().reindex_like(risky_r)

for step in range(n_steps):
    if drawdown is not None:
        peak = np.maximum(peak, account_value)
        floor_value = peak*(1-drawdown)
        cushion = (account_value - floor_value)/account_value
        risky_w = m*cushion
```

여기 if 문을 집어 넣어 주시면 됩니다~~

요약하면 만약 drawdown을 20%로 설정하면 floor value 는 peak의 eighty percent 가된다는 말입니다.

그리고 위에 peak = start\_value를 까먹으면 NameError: name 'peak' is not defined 이런거 나오니까 꼭 넣어 주세요! 부탁드립니다.

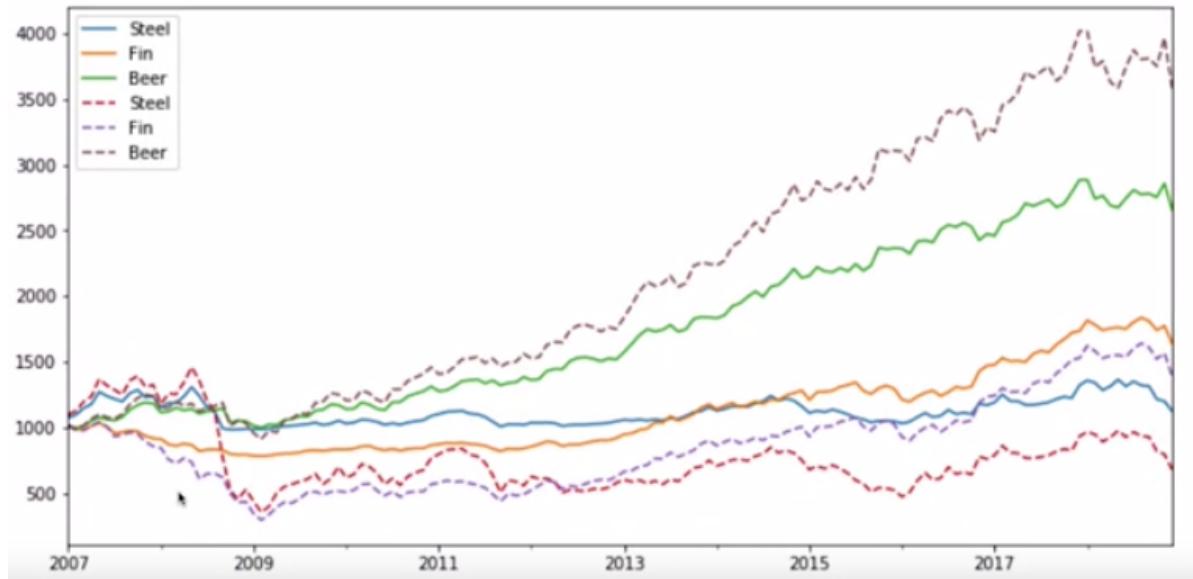
자자자 이제 끝이라네요

실수를 안했다면요....

```
btr = erk.run_cppi(ind_return["2007":][["Steel", "Fin", "Beer"]], drawdown = 0.25)

ax = btr["Wealth"].plot(figsize=(12,6))
btr["Risky Wealth"].plot(ax=ax, style="--")
```

plot을 합니다 조금 많습니다



```
erk.summary_stats(btr[["Risky_Wealth"]].pct_change().dropna())
```

	Annualized Return	Annualized Vol	Skewness	Kurtosis	Cornish-Fisher VaR (5%)	Historic CVaR (5%)	Sharpe Ratio	Max Drawdown
Steel	-0.039660	0.306407	-0.459951	4.782828	0.152288	0.203837	-0.221642	-0.758017
Fin	0.027364	0.212204	-0.695200	4.621401	0.105744	0.149862	-0.012370	-0.718465
Beer	0.111554	0.127971	-0.670797	4.650878	0.056497	0.077388	0.620132	-0.271368

```
erk.summary_stats(btr[["Wealth"]].pct_change().dropna())
```

	Annualized Return	Annualized Vol	Skewness	Kurtosis	Cornish-Fisher VaR (5%)	Historic CVaR (5%)	Sharpe Ratio	Max Drawdown
Steel	0.003784	0.097073	-0.441089	5.220481	0.047371	0.066991	-0.262958	-0.248059
Fin	0.041975	0.085028	-0.355163	4.153860	0.038342	0.054111	0.136964	-0.243626
Beer	0.084375	0.086263	-0.744111	4.571533	0.037937	0.051189	0.613413	-0.161186

drawdown을 25프로로 했으니까... MAX Drawdown이 25이하여야겠쥬

끝!

## 9. Simulating asset returns with random walks

주식 투자와 같은 위험한 asset 투자를 있다고 가정.

$S(t)$  : t시점에서 위험자산 가치

$B(t)$ : t시점에서의 risk-free 자산의 가치

아주 작은 시점 이후에서의 자산의 가치 변화를 알고 싶다면 미분하면 된다

$$\begin{cases} \frac{dS_t}{S_t} = \underbrace{(r + \sigma\lambda)}_{=\mu} dt + \sigma dW_t \\ \frac{dB_t}{B_t} = rdt \Leftrightarrow B_t = B_0 e^{rt} \end{cases}$$

첫번째 식에서 첫번째 component는 average 수익. 두번째 component는 변동값(stochastic)

두번째 모델은 riskfree이므로 일정

예시)

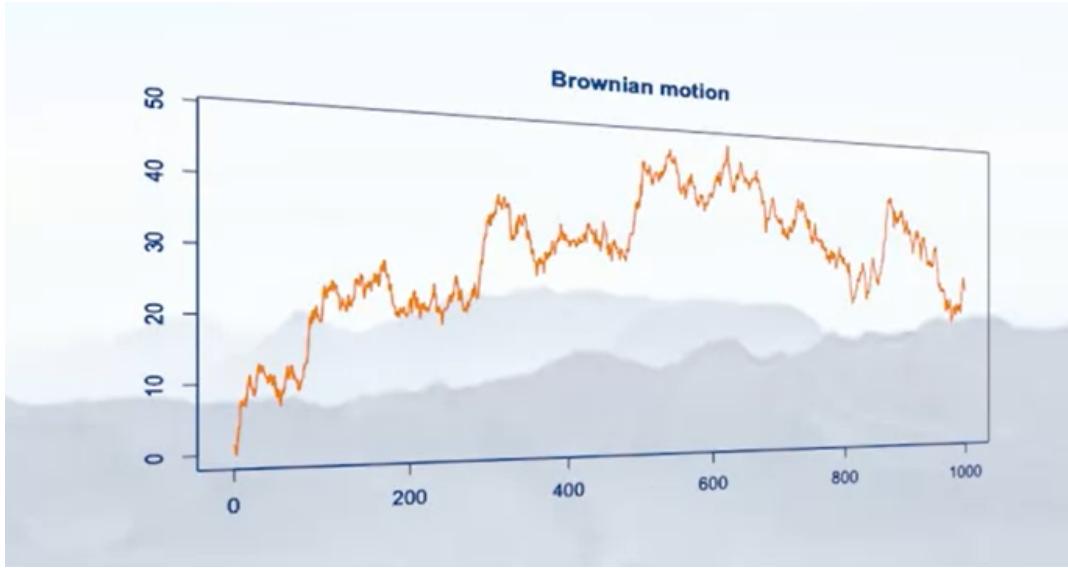
Assume that a stock index has a Sharpe ratio of 0.5 and a volatility of 20%. If the risk-free rate is 2%, what is the drift of the stock index return process?

- 2%
- 10%
- 12%

맞습니다

The drift of the stock index process is the expected return of the stock index return, which is given by  $2\% + 0.5 \times 20\% = 12\%$ .

<Brownian motion process>



Brownian motion은 random walk한 모형을 가지고있지만 stochastic하여 예측가능

brownian motion을 알기 위해서 discrete한 time을 가정하고 미분해보자

$$\frac{S_{t+dt} - S_t}{S_t} = (r + \sigma\lambda)dt + \sigma\sqrt{dt}\xi_t$$

앞서 살펴봤던 것 처럼 첫번째 component는 기대 수익.

두번째 컴포넌트는 시그마\*brownian motion임. 랜덤한 값

위 식을 통해서 dt기간 후의 S의 분포를 도출해 낼 수 있다

$$\left. \begin{aligned} \mathbf{E}_t \left[ \frac{dS_t}{S_t} \right] &= \mu dt \\ \mathbf{Var}_t \left[ \frac{dS_t}{S_t} \right] &= \sigma^2 dt \end{aligned} \right\} \Rightarrow \left. \begin{aligned} \frac{\mathbf{E}_t \left[ \frac{dS_t}{S_t} \right]}{dt} &= \mu \\ \frac{\mathbf{Var}_t \left[ \frac{dS_t}{S_t} \right]}{dt} &= \sigma^2 \end{aligned} \right\}$$

$dt$ 는 어떤 시간을 의미하고 있다고 했다.

연간 수익에 대한 내용이라면  $dt=12$ 로 나누면 된다.

정리하자면

brownian motion은 mean이 0이고  $dt$ 인 분산을 따름.

$m$ 은 연간 기대수익, 시그마는 연간 분산임.

예시)

If you estimate volatility of a portfolio return to be 2% over the monthly horizon, what would be your estimate for annual volatility?

- 2%
- 24%
- 6.9%

**맞습니다**

The square-root of time scaling law implies that annual volatility is  $\sqrt{12}$  times higher than monthly volatility. So the estimate for annual volatility is  $\sqrt{12} \times 2\% = 6.9\%$ .