“비중 조절은 내가 할게, 돈은 누가 가져 올래?”

33기 권민경
33기 백원우
34기 조성윤
34기 김서연

수리적 모델을 통한
배당 포트폴리오 최적화 연구

33기 권민경
33기 백원우
34기 조성윤
34기 김서연

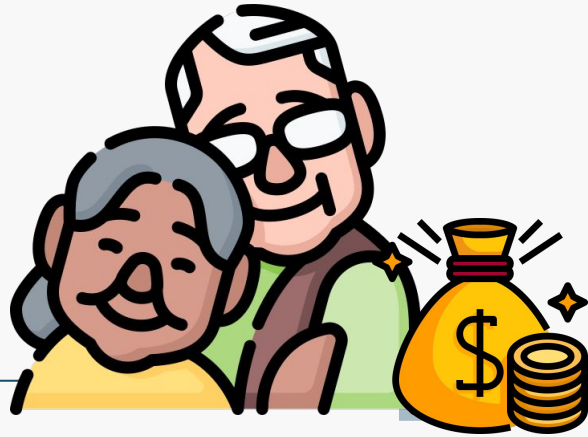
Table of Contents

01 개요

#재테크 #생애설계 #인플레이션 #노후 #은퇴 #초고령화

배당주?

- 초고령화 사회에 빠르게 도달하며 노후자산에 집중
- 수익성과 안전성: 높은 수익을 추구하는 포트폴리오보다 안전한 포트폴리오 추구, 낮은변동성
- 높은 현금유동성: 매달 지급받는 배당금, 현금화 쉬움
- 주가차익으로 인한 추가적인 자본이득
  : 배당금을 통한 매달 수익, 투자한 주식의 차익을 통한 수익도 기대해볼 수 있다.



배당포트폴리오의 특징

- 주가차익보다 배당을 통한 수익창출에 초점을 맞춘 포트폴리오
- 현금성이론 (Bird-in-Hand): 투자자들은 불확실한 자본이득보다 손 안의 배당(현금)을 선호한다는 이론

- 배당금
  : 기업의 이익잉여금의 일부를 주주에게 분배하는 것
  꾸준한 배당금의 지급은 기업에 대한 주주들의 신뢰성을 높여주는 근거

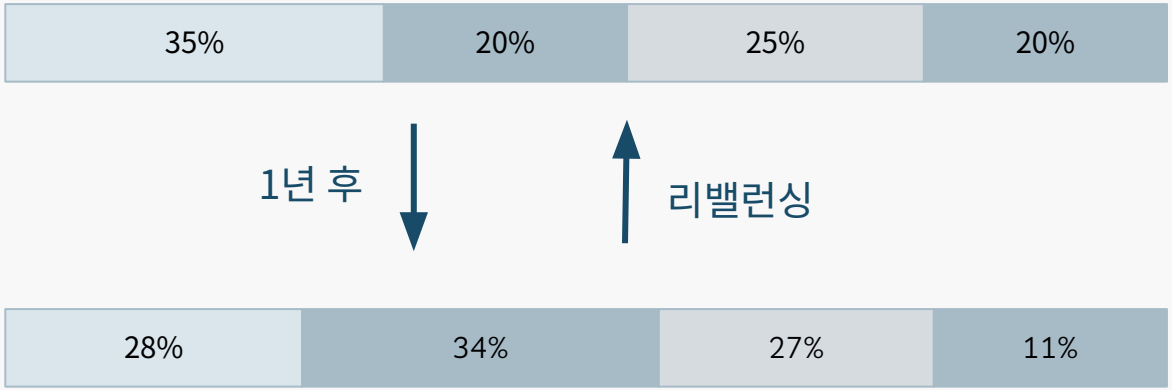
→ 매달 균일하게 수령받는 배당금을 통해 연금의 효과 창출

리밸런싱 (Rebalancing)

리밸런싱이란 정해진 일정 주기마다 현재 포트폴리오의 종목별 비중을 목표 비중으로 조정하는 것

리밸런싱 주기 기업마다 배당금을 지급하는 시기가 다르기 때문에 1년을 리밸런싱 주기로 설정 (21’ ~ 23’ 1.1 마다)

리밸런싱 방법



지난 3년 간의 데이터를 통해 향후 3년간 투자할 투자비율 설정

주가의 차익으로 인한 자산별 비중의 변화

리밸런싱 규칙

01. 연말 장 마감일에 전량 매도하고 앞서 구한 투자 비율에 맞추어서 연초 첫 개장일에 매수를 진행

02. 이전 연도에 수익이 발생하였다면 양도소득세 (개장일의 환율 기준)를 제외한 수익을 합쳐서 모두 재투자비용으로 사용

03. 이전 연도에 손실이 발생하였다면 추가 투자 없이 남은 금액을 재투자비용으로 사용

04. 배당 수익금의 경우 생활비로 사용하였다고 가정하였기에 재투자비용에 포함시키지 않음

슬라이딩 윈도우

투자를 진행하면서 사용하는 과거 데이터들을 최신화하면서 포트폴리오에 계속해서 최신 경향성을 녹여내는 방법

적용방법

: 과거 3년 간의 데이터를 바탕으로 투자비율을 정해서 1년 동안 투자를 진행한다.

2018	2019	2020	**2021 투자**
2019	2020	2021	**2022 투자**
2020	2021	2022	**2023 투자**

포트폴리오 종목 & 종목 선정 과정

선정된 종목 By Payout Ratio (배당 성향) 0.2~0.65

#귀족배당주: 25년 이상 꾸준히 배당을 증가시키고 지급해온 기업

Month/Ti	WBA	LEG	BEN	SWK	XOM	CVX	BKH	NFG	CAT	TGT	ITW	EMR	APD	PG	JNJ
1									O					O	
2	O				O	O	O			O		O			O
3		O	O	O				O			O		O		
4									O					O	
5	O				O	O	O			O		O			O
6		O		O				O			O		O		
7			O						O					O	
8	O				O	O	O			O		O			O
9		O	O	O				O			O		O		
10									O					O	
11	O			O	O	O	O			O		O			O
12		O	O					O			O		O		

WBA LEG BEN SWK XOM
CVX BKH NFG CAT TGT
ITW EMR APD PG JNJ
ATO GPC GD AOS HRL
LOW PPG PH ABM ADM
ABT CWT ADP TNC DOV

02

모델링

Sharpe & Sortino

고든 성장 모형의 활용

고든 성장 모형이란?

기업의 이익과 배당이 매년 g%만큼 일정하게 성장한다고 가정할 경우, 미래 예상되는 모든 배당금의 현재가치의 합으로 현재 주식의 가치를 선정

고든 성장 모델의 기대 수익률 idea 활용

기대수익률 E(p) = 성장률 + 배당수익률 = g% + d%

*성장률 = 배당 성장률 (기업이 일정한 비율로 성장 = gi)
*배당 수익률 = 직전년도 배당수익률 * (1+배당수익률 (gi))

고든 성장모형 응용 이유

- 선정한 30개의 종목의 경우 평균 25년 이상 꾸준히 배당금을 늘려옴
- 배당금이 매년 g%씩 일정하게 증가한다는 가정을 하는 고든 성장 모형에 부합

샤프지수

1. **샤프지수: 위험대비 투자수익에 대한 지표**

$$\frac{R_p - R_f}{\sigma_p}$$ **:** 분자는 초과수익 (포트폴리오 수익에서 무위험 수익을 뺀 값)
분모는 포트폴리오 초과수익의 표준편차를 의미

- 포트폴리오 기대수익률에 고든성장모형의 가정을 차용하여 구현하였다.

R_p **= g_i(%)(배당성장률) + d_i(%)(배당수익률)**

2. **샤프지수의 의미**
- 포트폴리오의 초과수익을 포트폴리오의 변동성으로 나눠 투자자가 위험에 대한 대가로 얼마나 많은 초과수익을 얻었는지를 평가
- 샤프지수가 높을수록 리스크가 조정된 포트폴리오를 의미한다.

Modeling

01. 목적함수 : F(x_i)

02. 결정변수 : x_i

03. 제약식

$$\text{Minimize} \quad 500\,x_1 + 200\,x_2 + 250\,x_3 + 125\,x_4$$

$$\text{subject to} \quad 50\,x_1 + 25\,x_2 + 20\,x_3 + 15\,x_4 \geq 1{,}500$$

$$0 \leq x_1 \leq 20$$

$$0 \leq x_2 \leq 15$$

$$0 \leq x_3 \leq 10$$

$$0 \leq x_4 \leq 15$$

Parameter

배당 성장률 (g_i)

종목 당 3년 간의 배당 성장률의 기하 평균값을 Parameter로 사용

WBA	LEG	BEN	SWK	XOM	CVX	...	LOW	PPG	PH	ABM	ADM	ABT
0.050184	0.053496	0.062712	0.035855	0.058892	0.06119	...	0.118813	0.05896	0.153642	0.039946	0.03847	0.096561

배당 수익률 (d_i)

종목 당 3년 간의 평균 배당 수익률을 Parameter로 사용

WBA	LEG	BEN	SWK	XOM	CVX	...	LOW	PPG	PH	ABM	ADM	ABT	CWT
0.031317	0.036792	0.038328	0.018607	0.051975	0.043308	...	0.018273	0.0165	0.01811	0.021023	0.032077	0.01664	0.017079

투자 비중 (w_i)

x_i / 300000 = w_i

CVX	BKH	NFG	CAT	TGT	...	LOW	PPG	PH	ABM	ADM
0.02237	0.018042	0.012577	0.080501	0.038511	...	0.054576	0.033126	0.092449	0.010708	0.014551

w_i 투자비중, g_i 배당성장률, d_i 배당수익률, v_{ij} i종목, j월 배당금, a_i 주가

Parameter

주가(a_i)
:i 종목의 해당 날짜의 주가

WBA	LEG	BEN	SWK	XOM	CVX	...	LOW	PPG	PH	ABM	ADM	ABT
21.219999	23.620001	24.27	91.760002	104.959999	144.460007	...	203.699997	135.869995	432.640015	41.07	73.959999	99.550003

배당 수익금 (v_ij)
i 기업에 j월에 한 주당 지급하는 배당금

연도	월	WBA	LEG	BEN	SWK	XOM	CVX	BKH	NFG	...	LOW	PPG	PH	ABM	ADM	ABT	CWT	ADP	TNC	DOV
2023	1	0.00	0.00	0.0	0.0	0.00	0.00	0.000	0.000	...	1.05	0.00	0.00	0.22	0.00	0.51	0.00	0.00	0.000	0.000
2023	2	0.48	0.00	0.0	0.0	0.91	1.51	0.625	0.000	...	0.00	0.62	1.33	0.00	0.45	0.00	0.26	0.00	0.000	0.505
2023	3	0.00	0.44	0.3	0.8	0.00	0.00	0.000	0.475	...	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.25	0.265	0.000
2023	4	0.00	0.00	0.0	0.0	0.00	0.00	0.000	0.000	...	1.05	0.00	0.00	0.22	0.00	0.51	0.00	0.00	0.000	0.000
2023	5	0.48	0.00	0.0	0.0	0.91	1.51	0.625	0.000	...	0.00	0.62	1.48	0.00	0.45	0.00	0.26	0.00	0.265	0.505
2023	6	0.00	0.46	0.3	0.8	0.00	0.00	0.000	0.495	...	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.25	0.000	0.000
2023	7	0.00	0.00	0.0	0.0	0.00	0.00	0.000	0.000	...	1.10	0.00	0.00	0.22	0.00	0.51	0.00	0.00	0.000	0.000
2023	8	0.48	0.00	0.0	0.0	0.91	1.51	0.625	0.000	...	0.00	0.65	1.48	0.00	0.45	0.00	0.26	0.00	0.265	0.510
2023	9	0.00	0.46	0.3	0.0	0.00	0.00	0.000	0.495	...	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.25	0.000	0.000
2023	10	0.00	0.00	0.0	0.0	0.00	0.00	0.000	0.000	...	1.10	0.00	0.00	0.22	0.00	0.51	0.00	0.00	0.000	0.000
2023	11	0.48	0.00	0.0	0.0	0.95	1.51	0.625	0.000	...	0.00	0.65	1.48	0.00	0.45	0.00	0.26	0.00	0.000	0.000
2023	12	0.00	0.00	0.0	0.0	0.00	0.00	0.000	0.000	...	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.000	0.000

w_i 투자비중, g_i 배당성장률, d_i 배당수익률, v_{ij} i종목, j월 배당금, a_i 주가

Modeling

$$Max: Sharpe = \left(\frac{R_p - R_f}{\sigma_p}\right)$$

$$E(p) = \sum_{i=1}^{30} g_i * w_i + \sum_{i=1}^{30} w_i d_i (1 + g_i),\ \sigma_p = \sqrt{\sum^{30}\sum^{30} w_i w_j Cov(i,j)}$$

$$Max\ \frac{\sum_{i=1}^{30} w_i * g_i + \sum_{i=1}^{30} w_i d_i (1 + g_i) - R_f}{\sqrt{\sum_{i=1}^{30}\sum_{j=1}^{30} w_i * w_j * Cov(i,j)}}$$

01. 목적함수

→ 샤프지수를 최대화 (샤프지수의 기대수익률 계산에 고든성장 모형을 사용)

→ 고든 성장 모형의 가정 차용 + g%(배당성장률)을 통한 기대수익 추정

→ 향후 3년간 투자를 진행할 것이라고 가정하고 목적함수 값을 Maximize

w_i 투자비중, g_i 배당성장률, d_i 배당수익률, v_{ij} i종목, j월 배당금, a_i 주가, x_i 구매한 주식 수

Modeling

$$Max : Sharpe = \left(\frac{R_p - R_f}{\sigma_p} \right)$$

$$E(p) = \sum_{i=1}^{30} g_i * w_i + \sum_{i=1}^{30} w_i d_i (1 + g_i), \ \sigma_p = \sqrt{\sum^{30} \sum^{30} w_i w_j Cov(i,j)}$$

$$Max \ \frac{\sum_{i=1}^{30} w_i * g_i + \sum_{i=1}^{30} w_i d_i (1 + g_i) - R_f}{\sqrt{\sum_{i=1}^{30} \sum_{j=1}^{30} w_i * w_j * Cov(i,j)}}$$

w_i 투자비중, g_i 배당성장률, d_i 배당수익률, v_{ij} i종목, j월 배당금, a_i 주가, x_i 구매한 주식 수

제약식

→ $\sum_i x_i v_{ij} > 0,\ for\ \forall j$

배당금을 안 받는 달이 없도록 한다

→ $\sum_i x_i \times a_i \leq 300,000$

총 투자 비용은 현재 가진 자본금을 넘지 못한다

→ $\$300 < \sum_i x_i v_{ij}\ for\ \forall j$

매달 지급되는 배당금이 최소 $300은 되게 한다

i월에 x비중으로 투자 시 들어오는 배당금

w_i 투자비중, g_i 배당성장률, d_i 배당수익률, v_{ij} i종목, j월 배당금, a_i 주가, x_i 구매한 주식 수

Modeling

실제 상황에 맞는 조건들 추가

포트폴리오를 실제로 투자할 때 발생될 세금에 대해서 추가적인 고려를 진행함

배당금을 지급받을 때 당시의 환율을 고려하였고 실제 원화로 받는 배당금을 계산

또한 미국주식에 투자시에 발생하는 양도소득세(22%)에 대한 제약을 구현

최적화 진행

2018~2020년의 데이터를 이용해서 2021~2023년도 3년 동안 리밸런싱을 포함하여 백테스팅 진행

실제 리밸런싱을 함께 진행하면서 3년의 투자를 진행하였을 때의 결과 분석

세금

1. **증권거래세**
   : 자산이 보유중인 주식을 매도했을 경우에 발생되는 세금
   해외주식의 경우 0.0022%로 매우 낮고 5년에 거래를 10번만 진행하기에 고려하지 않음
2. **배당소득세**
   : 배당금의 경우 미국에서 원천징수 되어서 지급되는 과정에서 15%의 세금을 제하고 배당금 지급
   미국주식 배당소득세의 경우 15%
3. **양도소득세**
   : 자산을 사고팔 때 발생되는 차익에 대한 세금
   해외주식의 경우 1년에 250만원 이상의 수익이 발생할 경우,
   250만원을 제외한 금액에 대하여 22%의 양도소득세 적용
   리밸런싱을 진행할 때 양도소득세를 내야하는 경우 세금을 제외한 자본 이득을 추가하여 재투자

파이썬 구현 - 데이터 불러오기

```python
import numpy as np
import pandas as pd
from scipy.optimize import minimize, LinearConstraint, basinhopping #최적화 패키지
```

[97]

+ 코드 + Markdown

```python
dividend = pd.read_excel('data.xlsx', sheet_name='dividend') # 2000년 ~ 2023년, 30개 종목의 배당금 데이터
cons12 = pd.read_excel('data.xlsx', sheet_name='cons1') #월별 배당금 제약조건
tickers = cons12.T.index[2:].to_list() #tickers list
stock = pd.read_excel('data.xlsx', sheet_name='stock_price') # 2000-01-03 ~ 2023-11-28, 30개 종목의 주가 데이터
exchange = pd.read_excel('data.xlsx', sheet_name='exchange_rate') # 2000-01 ~ 2023-11, 한 달 평균 환율 데이터
```

[99]

```python
# 데이터 전처리: 날짜 데이터 타입으로 변환
stock['Date'] = pd.to_datetime(stock['Date'])
stock = stock.sort_values('Date')

dividend['Ex-Dividend Date'] = pd.to_datetime(dividend['Ex-Dividend Date'])
dividend = dividend.sort_values('Ex-Dividend Date')
dividend['Month'] = dividend['Ex-Dividend Date'].apply(lambda x: x.month)  # 월 feature 추가

exchange['Date'] = pd.to_datetime(exchange['Date'])
```

[100]

샤프비율 목적함수 구현

```python
def obj(x): #목적함수 구현

    x = np.array(x) # 결정변수 행렬 # 종목별 구매 주식 수
    total_investment = np.dot(x, price) # 전체 투자 금액
    weights = (x*price)/total_investment # 투자 비중
    variance = np.dot(weights.T, np.dot(cov,weights)) # variance = sigma(wiwjcovij)
    std = np.sqrt(variance) #표준편차

    Ep = np.dot(weights,growth_rate) + np.dot(weights,div_rate*(1+growth_rate)) # 기대수익률 = 기대 성장률 + 기대 배당수익률

    result = -(Ep-Rf)/std  #sharpe ratio

    return result
```

$$Max \frac{\sum_{i=1}^{30} w_i * g_i + \sum_{i=1}^{30} w_i d_i (1+g_i) - R_f}{\sqrt{\sum_{i=1}^{30}\sum_{j=1}^{30} w_i * w_j * Cov(i,j)}}$$

w_i 투자비중, g_i 배당성장률, d_i 배당수익률, $\mathbf{v_{ij}}$ i종목, j월 배당금, $\mathbf{a_i}$ 주가

파이썬 구현 – parameter 전처리

```
#목적 함수 제약 함수 활용

#목적 함수 Parameter 1 : 배당성장률
growth_rate = pd.read_excel('data.xlsx', sheet_name='growth_rate', index_col='Unnamed: 0')
growth_rate = growth_rate.values[3]
#목적 함수 Parameter 2 : 배당수익률
div_rate = pd.read_excel('data.xlsx', sheet_name='div_rate',index_col='Unnamed: 0')
div_rate = div_rate.values[0]
#목적 함수 Parameter 3 : 무위험수익률 #미국 1년 국채 수익률
rf = pd.read_excel('data.xlsx', sheet_name='rf')
Rf = float(rf[rf['date'] == str3]['close'].values)/100

#제약식 Parameter 1 : 내년 예상 배당금 matrix (30*12) (직전 연도 배당 테이블 * 평균배당성장률)
cons1 = pd.read_excel('data.xlsx', sheet_name='cons1')
cons1 = cons1[cons1['연도']==year-1].iloc[:,2:] #해당 연도의 배당금 지급
cons1 = cons1.mul(1+growth_rate)
```

배당 수익금 (v_ij)
i 기업에 j월에 한 주당 지급하는 배당금

w_i 투자비중, g_i 배당성장률, d_i 배당수익률, v_{ij} i종목, j월 배당금, a_i 주가

파이썬 구현 – 최적화 진행

```
#1 : 주식분배금액 = 초기 투자금액
cons = [{'type': 'eq', 'fun': lambda x: np.dot(x,price)-investment}]
#2 : 예상 배당금을 매달 300 달러 이상!
for i in range(0,12):                                   #1월에 들어올 배당금 >= $300
    cons.append({'type': 'ineq', 'fun': lambda x, i=i: np.dot(x, cons1.iloc[i,:]) - 300})

bnds = ((0,None), ) * 30
# 임의의 초기값
x0 = np.array([100]*30)

minimizer_kwargs = {"method":"SLSQP", "constraints":cons, "bounds":bnds}

res = basinhopping(obj, x0, minimizer_kwargs=minimizer_kwargs, niter=20)
```

$\sum x_i \times a_i \leq 300,000$ = 들고있는 돈

$\$300 < \sum_i x_i v_{ij} \ for \forall j$

w_i 투자비중, g_i 배당성장률, d_i 배당수익률, v_{ij} i종목, j월 배당금, a_i 주가

파이썬 구현 - 리밸런싱 함수

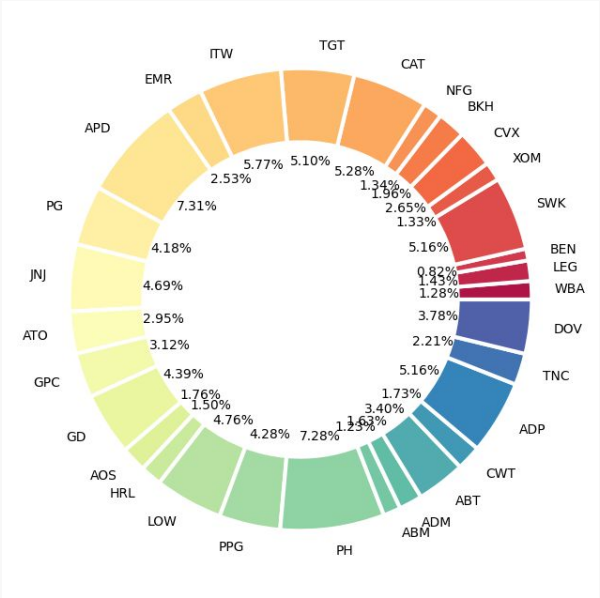
```python
##### 리밸런싱 함수 #####

def Rebalancing(year, weight, investment):
    price_start = np.array(stock[stock['Date'].apply(lambda x: x.year) == year].iloc[0, 1:]) # 1월 1일 주가
    price_end = np.array(stock[stock['Date'].apply(lambda x: x.year) == year].iloc[-1, 1:]) # 해당 연도 마지막 거래 주가
    allocation = (weight * investment) / price_start # 종목별 주식 수 list 값으로 저장

    div = [] #주식에 1년동안 투자시에 받을 배당금
    for ticker in tickers:
        year_div_data = dividend[dividend['Ex-Dividend Date'].apply(lambda x: x.year) == year]
        div.append(np.sum(year_div_data[year_div_data['Ticker'] == ticker]['Dividend']))

    monthly_real_div = [] # 월별 실수령 배당금 리스트 -> 생활비로 사용 가정
    for i in range(1, 13):

        year_div_data = dividend[dividend['Ex-Dividend Date'].apply(lambda x: x.year) == year].groupby(['Month', 'Ticker']).sum()['Dividend'].to_frame().

        year_div_data = year_div_data[year_div_data['Month'] == i].drop('Month', axis=1).set_index('Ticker').T

        a = []
        for ticker in tickers:
            try: a.append(np.float(year_div_data[ticker]))  # 0에 float 취하면 오류가 생겨서 그런건지 try로 진행해야함
            except: a.append(0)
        monthly = np.dot(allocation, a) # 월별 배당금 내적(i월 배당급 지급급액*투자비중)
        monthly_real_div.append(monthly)

    actual_stock = np.dot(allocation, price_end) # 연말 포트폴리오 가치
    actual_dividend = np.dot(allocation, div) # 배당금 총합

    return actual_stock, actual_dividend, monthly_real_div
```

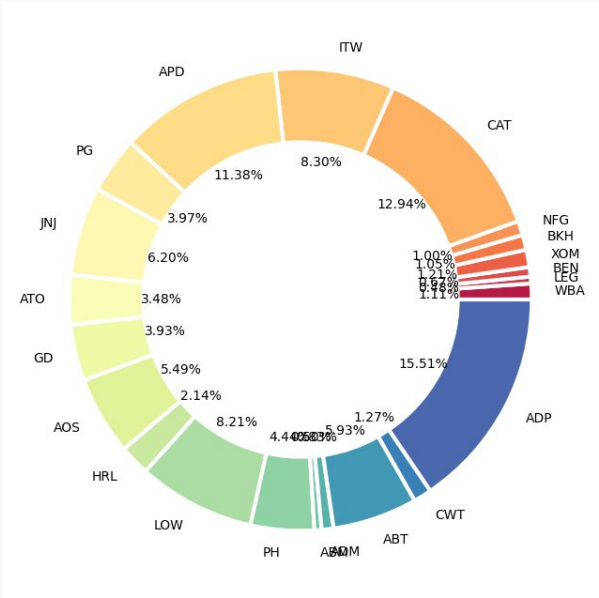
Python

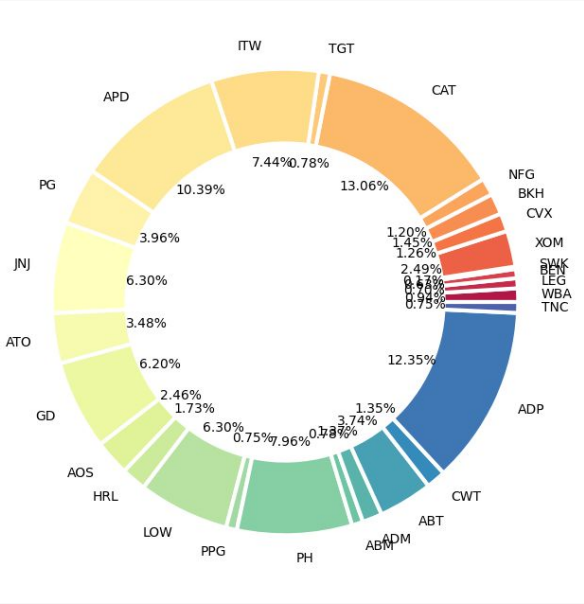
파이썬 구현 - 환율&세금 적용

```
#환율 계산
rate = exchange[exchange['Date'].apply(lambda x: x.year) == year]

### 세금 계산
if profit * rate.iloc[0, 1] > 2500000:
    tax = ((profit * rate.iloc[0, 1]) - 2500000) * 0.22 # 0.22: 해외주식 수익의 250만원을 제외한 금액에 대해 양도소득세 22% 적용
    x1 = x1 - (tax/rate.iloc[0, 1])
else:
    tax = 0
```

파이썬 구현 – 투자 비중 파이 (sharpe)

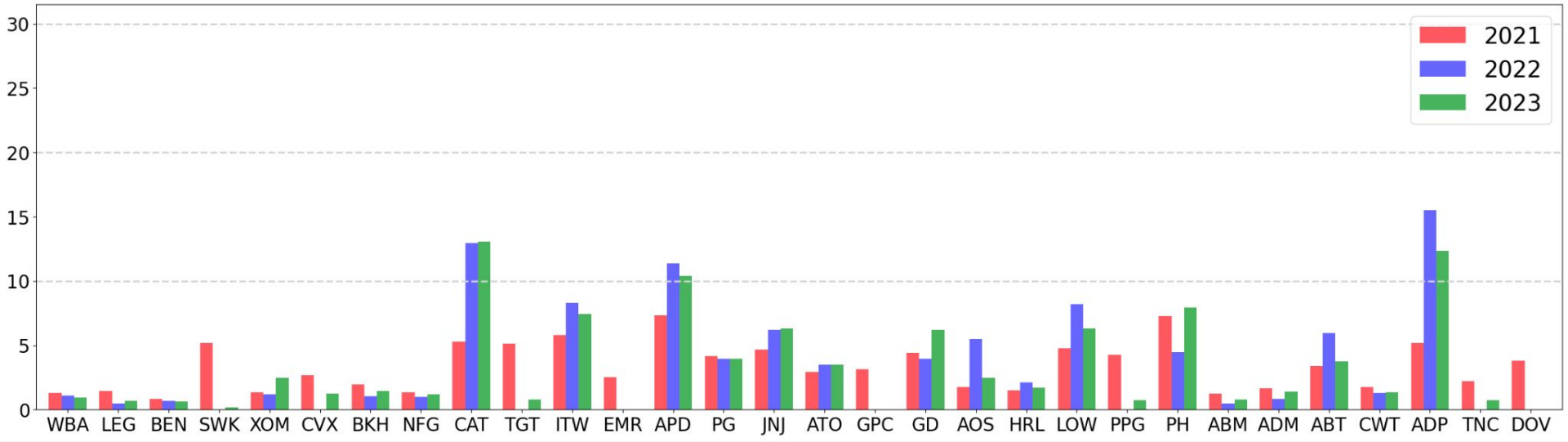




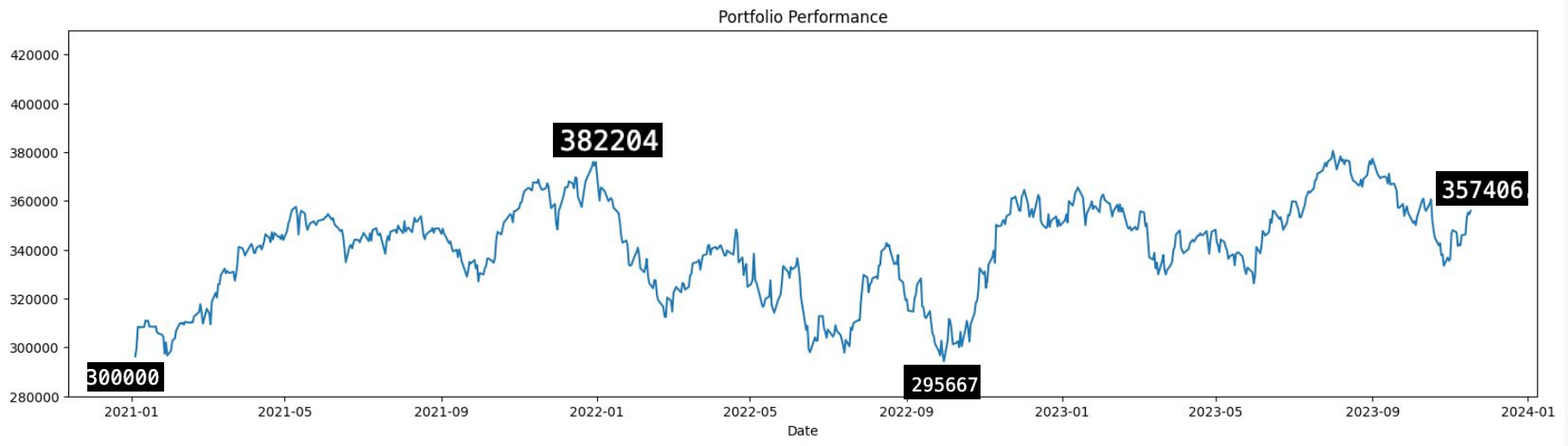


2021 2022 2023

파이썬 구현 – 투자 비중 차트 (sharpe)



파이썬 구현 - 결과 그래프 - (sharpe)



	1	2	3	4	5	6	7	8	9	10	11	12	sum
Date													
2021	415.40	827.25	563.48	446.00	832.95	536.14	474.34	901.95	527.79	434.58	911.50	512.14	7383.52
2022	700.94	443.96	696.48	706.20	459.72	697.95	752.94	460.53	700.11	767.19	470.22	729.48	7585.72
2023	709.44	614.64	702.84	720.37	668.85	815.57	650.69	673.19	706.65	775.31	505.17	NaN	7542.72

Sortino Ratio

1. Sortino Ratio

: upside potential 대신 downside deviation(하방위험)을 이용하여 포트폴리오의 위험대비 기대수익을 나타내는 지표

$$\frac{R_p - R_f}{\sigma_d}$$: 분자에서 목표수익률을 고려하고 분모에서 해당 비율의 하방편차만 반영

2. 샤프비율과의 차이점

: 둘 다 위험조정수익지표이지만 sharpe 비율과 달리 하락 변동성만 고려하여
투자자에게 보다 정확한 위험 측정치를 제공한다.

3. Sortino Ratio의 의미

: Sortino Ratio가 높을수록 포트폴리오가 기대수익 대비 하락위험을 줄이는 것으로 판단
보수적 투자자들은 Sortino 지수선호

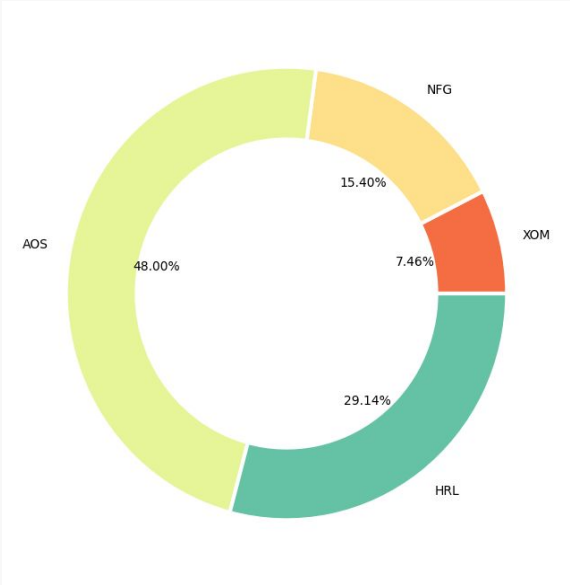
Sortino Ratio 목적함수

```python
def obj(x): #목적함수 구현

    x = np.array(x) # 결정변수 행렬 # 종목별 구매 주식 수
    total_investment = np.dot(x, price) # 전체 투자 금액
    weights = (x*price)/total_investment # 투자 비중
    variance = np.dot(weights.T, np.dot(cov,weights)) # variance = sigma(wiwjcovij)
    std = np.sqrt(variance) #표준편차

    Ep = np.dot(weights,growth_rate) + np.dot(weights,div_rate*(1+growth_rate)) # 기대수익률 = 기대 성장률 + 기대 배당수익률

    result = -(Ep-Rf)/std  #sharpe ratio

    return result
```

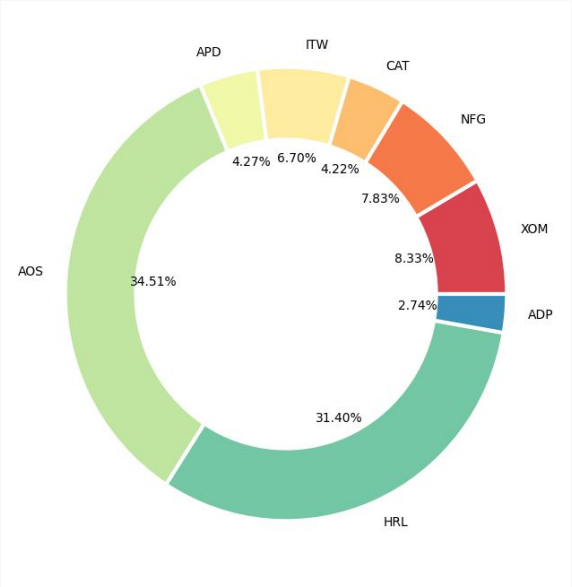
$$Max \frac{\sum_{i=1}^{30} w_i * g_i + \sum_{i=1}^{30} w_i d_i (1+g_i) - R_f}{\sqrt{\sum_{i=1}^{30}\sum_{j=1}^{30} w_i * w_j * Cov(i,j)}}$$

```python
s1 = stock_1.pct_change().fillna(0)
cov1 = s1[s1<0].cov()
s2 = stock_2.pct_change().fillna(0)
cov2 = s2[s2<0].cov()
s3 = stock_3.pct_change().fillna(0)
cov3 = s3[s3<0].cov()
cov = 0.1*(cov1)+0.25*(cov2)+0.65*(cov3)
```

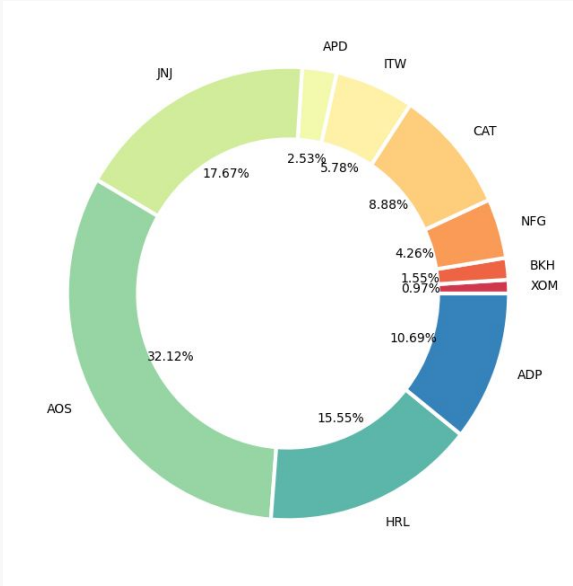
파이썬 구현 – 투자 비중 파이 (sortino)



2021

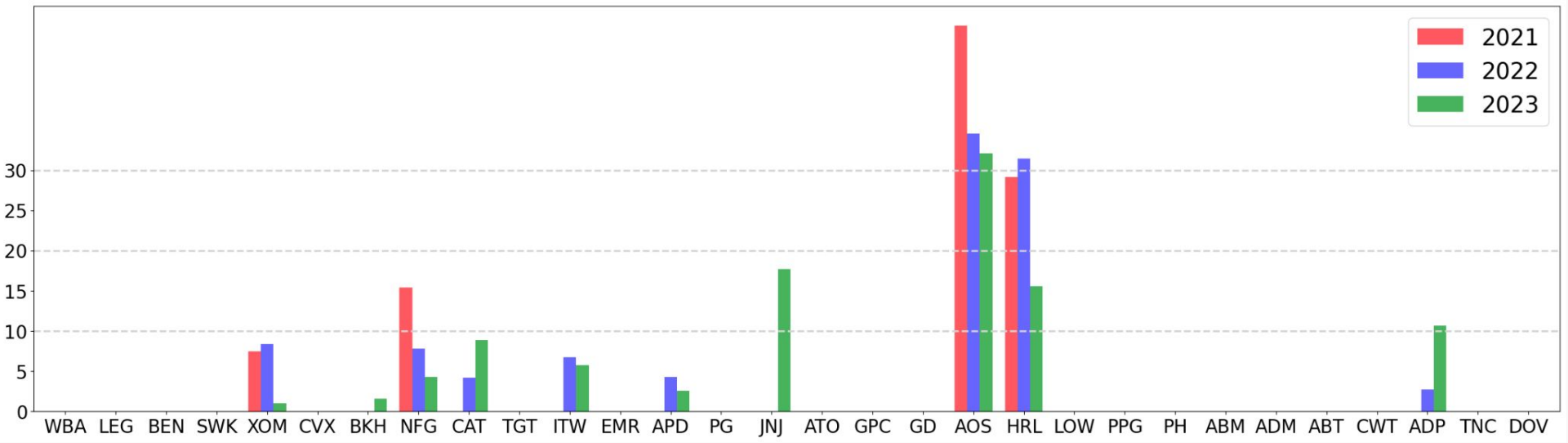


2022

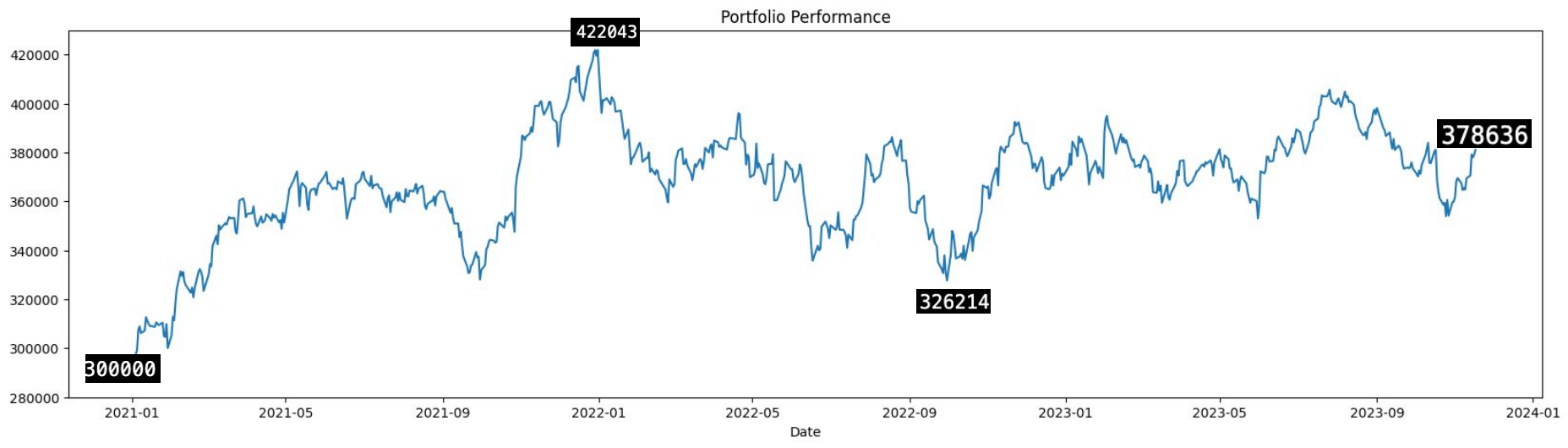


2023

파이썬 구현 - 투자 비중 차트 (sortino)



파이썬 구현 – 결과 그래프 – (sortino)



	1	2	3	4	5	6	7	8	9	10	11	12	sum
Date													
2021	1233.48	500.00	500	1233.48	500.00	511.38	1233.59	500.00	511.49	1295.55	500.00	511.5	9030.47
2022	1268.23	500.00	500	1268.23	500.00	503.84	1275.03	500.00	517.07	1313.81	505.75	522.0	9173.96
2023	1187.63	517.76	500	1187.63	547.87	503.28	1203.68	547.87	515.25	1250.28	500.00	NaN	8461.25

투자 결과분석 (2021.1.1~2023.11.17)

Sharpe Ratio 최종 수익 : 18.66%+7.5%(배당)
→ 26.16%

Sortino Ratio 최종 수익 : 27.02%+8.8%(배당)
→ 35.82%

S&P 500(SPY) : 20.57% + 4.2%(배당)
→ 24.77%

SCHD : 10.91% + 10.95%(배당)
→ 21.86%

한계 및 개선사항

1)백테스팅 진행 X (모델의 성능 평가 데이터 부족)

2)고든성장모형을 활용한 기대수익률 구현에 대한 타당성 부족 (현실은 가정과 맞지않음)

3)sortino ratio 구현과정에서 cov matrix 계산 방식에 대한 근거 부족

4)MDD, 변동성 등을 고려하지 않은 단순 수익 기반 결과 분석

5)투자 시작 시점을 반드시 연초에 진행해야 돌아가는 제한적 코드

Thanks!

Any questions?

chu ga mun E sa hang to seungyun0727@naver.com
010-9476-1793

CREDITS: This presentation template was created by **Slidesgo,** and includes icons by **Flaticon,** and infographics & images by **Freepik**