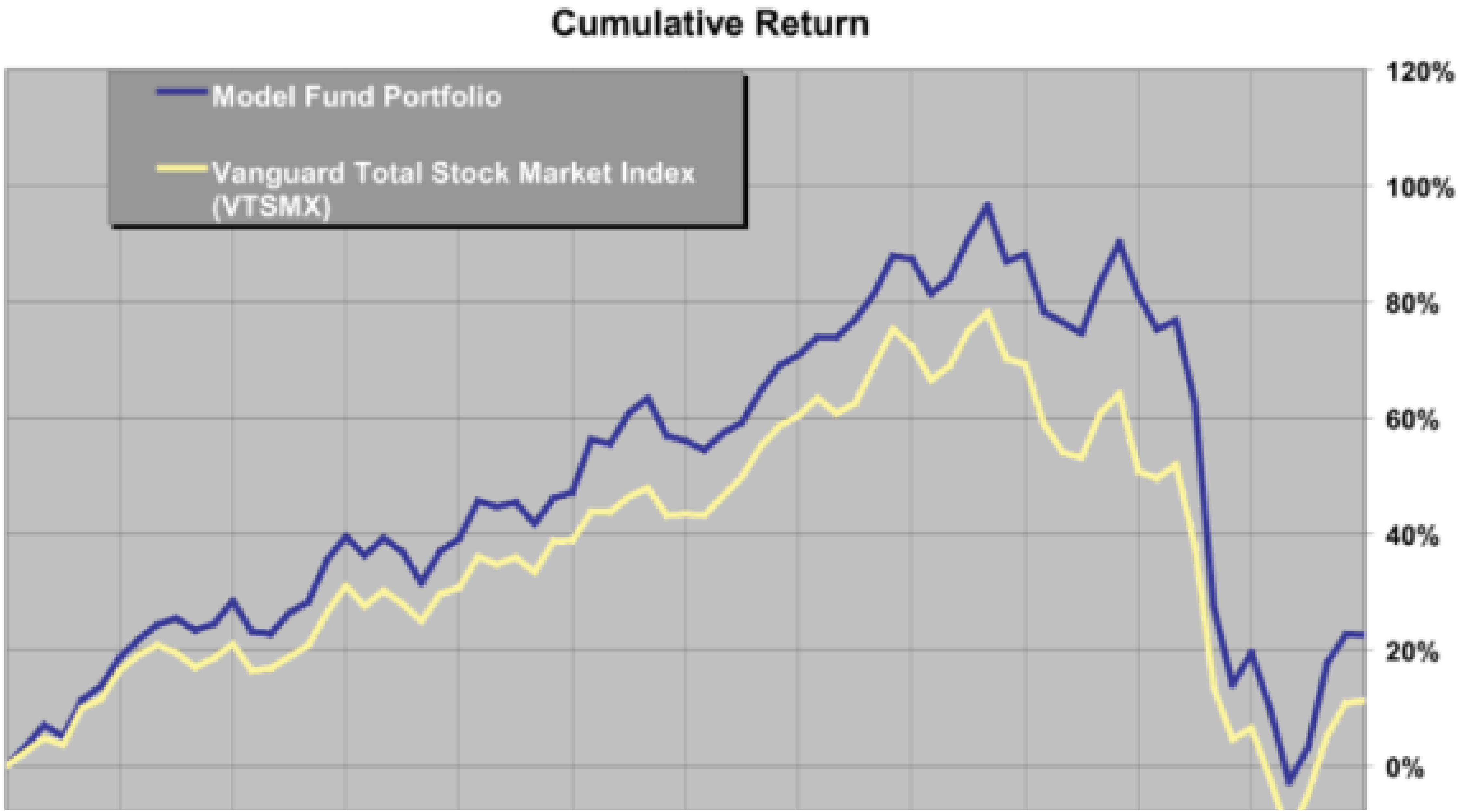
Comparing against a benchmark

INTRODUCTION TO PORTFOLIO ANALYSIS IN PYTHON

Charlotte Werger
Data Scientist

Active investing against a benchmark



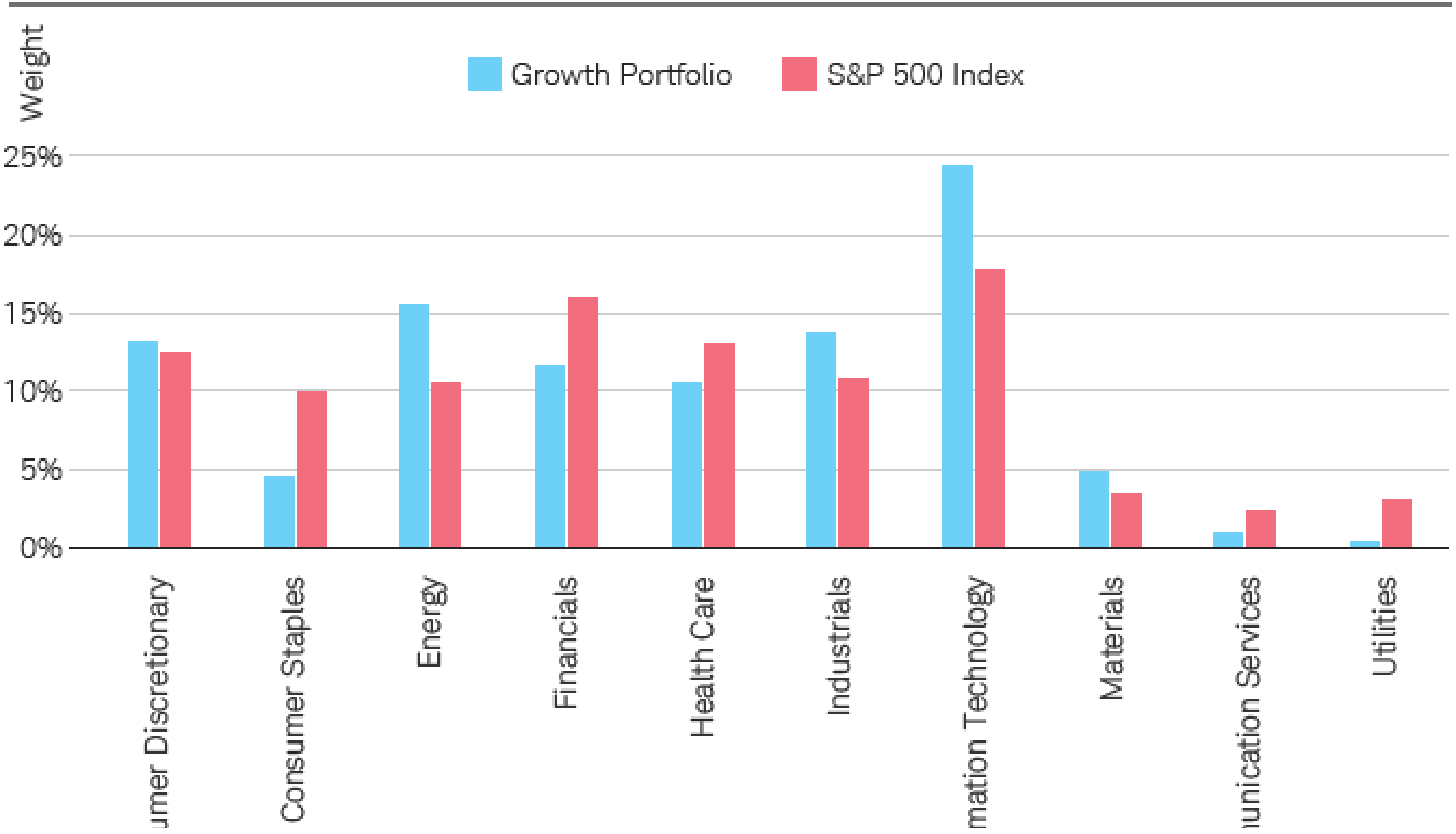
Active return for an actively managed portfolio

- Active return is the performance of an (active) investment, **relative** to the investment's benchmark.
- Calculated as the **difference between the benchmark and the actual return**.
- Active return is achieved by "active" investing, i.e. taking **overweight and underweight positions** from the benchmark.

Tracking error for an index tracker

- Passive investment funds, or **index trackers**, don't use active return as a measure for performance.
- **Tracking error** is the name used for the difference in portfolio and benchmark for a **passive** investment fund.

Active weights



[1] Source: Schwab Center for Financial Research.

Active return in Python

```
# Inspect the data
portfolio_data.head()
```

```
       mean_ret    var     pf_w     bm_w     GICS Sector
Ticker
A         0.146    0.035   0.002    0.005    Health Care
AAL       0.444    0.094   0.214    0.189    Industrials
AAP       0.242    0.029   0.000    0.000    Consumer Discretionary
AAPL      0.225    0.027   0.324    0.459    Information Technology
ABBV      0.182    0.029   0.026    0.010    Health Care
```

[1] Global Industry Classification System (GICS)

Active return in Python

```
# Calculate mean portfolio return
total_return_pf = (pf_w*mean_ret).sum()
```

```
# Calculate mean benchmark return
total_return_bm = (bm_w*mean_ret).sum()
```

```
# Calculate active return
active_return = total_return_pf - total_return_bm
print ("Simple active return: ", active_return)
```

```
Simple active return: 6.5764
```

Active weights in Python

```
# Group dataframe by GICS sectors
grouped_df=portfolio_data.groupby('GICS Sector').sum()
```

```
# Calculate active weights of portfolio
grouped_df['active_weight']=grouped_df['pf_weights']-
                            grouped_df['bm_weights']
```

```
print (grouped_df['active_weight'])
```

```
GICS Sector
Consumer Discretionary          20.257
Financials                      -2.116
...etc
```

Let's practice!

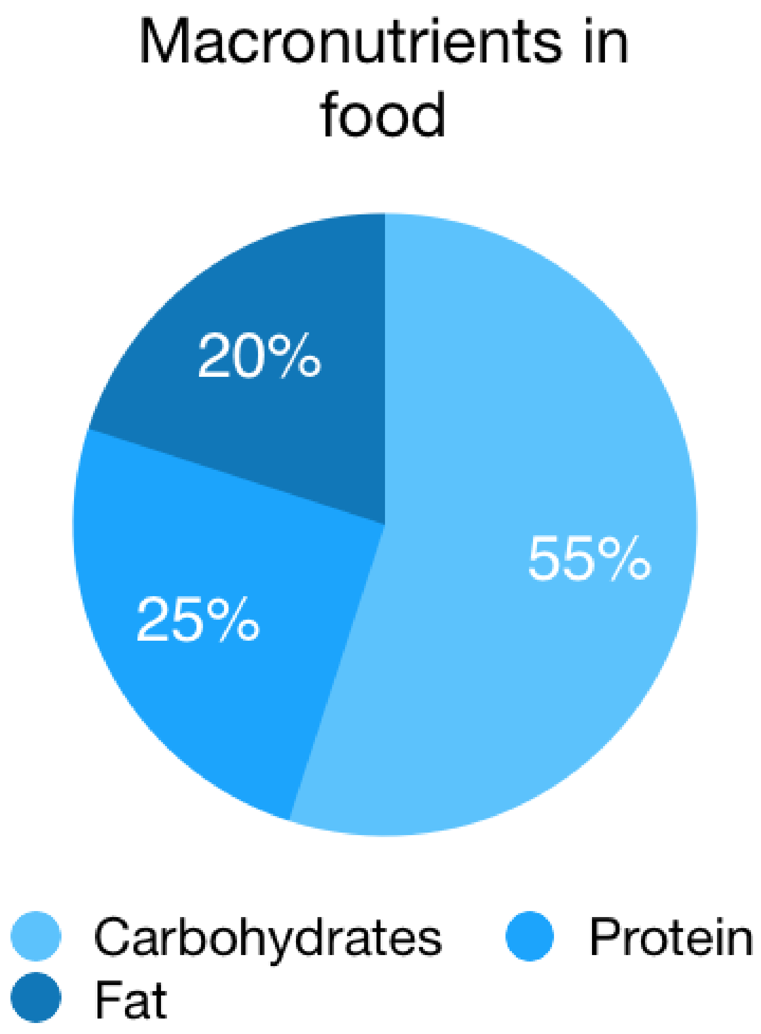
Risk factors

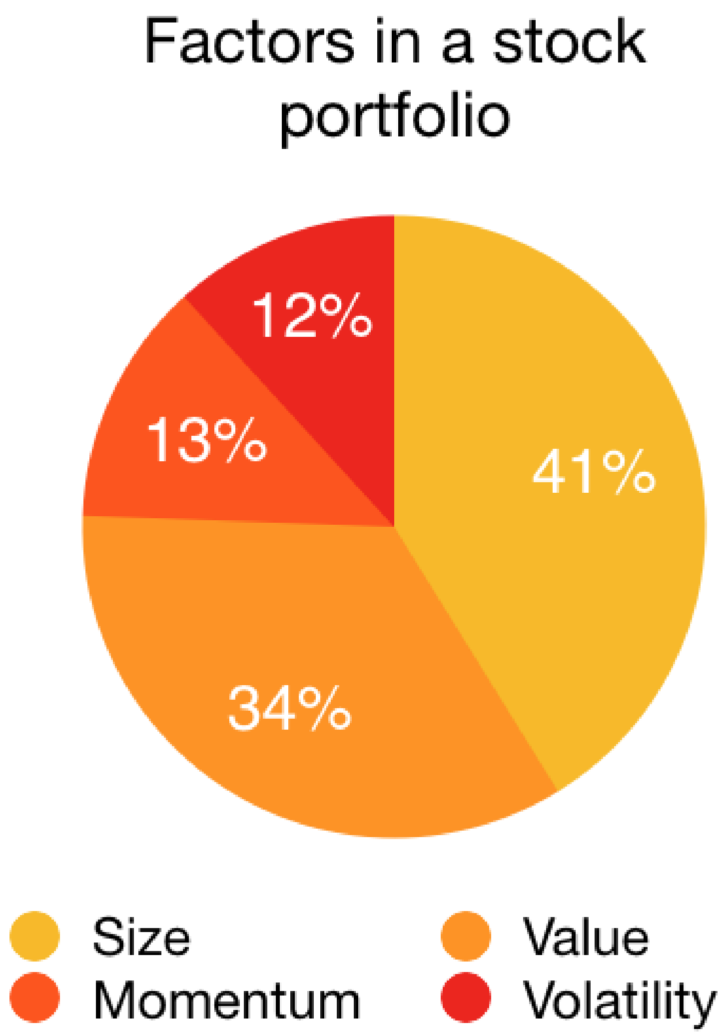
INTRODUCTION TO PORTFOLIO ANALYSIS IN PYTHON

Charlotte Werger
Data Scientist

What is a factor?

Factors in portfolios are like nutrients in food



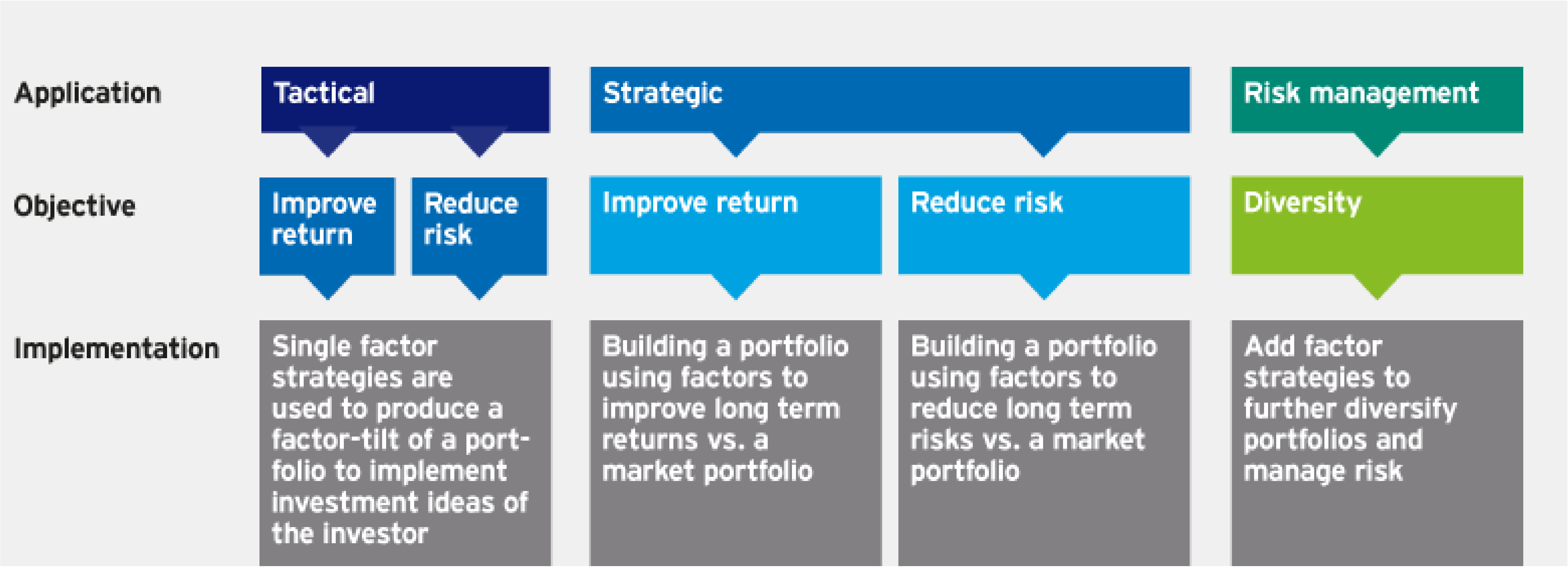


Factors in portfolios

Different types of factors:

- Macro factors: interest rates, currency, country, industry
- Style factors: momentum, volatility, value and quality

Using factor models to determine risk exposure



[1] Source: https://invesco.eu/investment [2] campus/educational [3] papers/factor [4] investing

Factor exposures

```
df.head()
```

```
date          portfolio     volatility     quality
2015-01-05    -1.827811     1.02           -1.76
2015-01-06    -0.889347     0.41           -0.82
2015-01-07     1.162984     1.07            1.39
2015-01-08     1.788828     0.31            1.93
2015-01-09    -0.840381     0.28           -0.77
```

Factor exposures

```
df.corr()
```

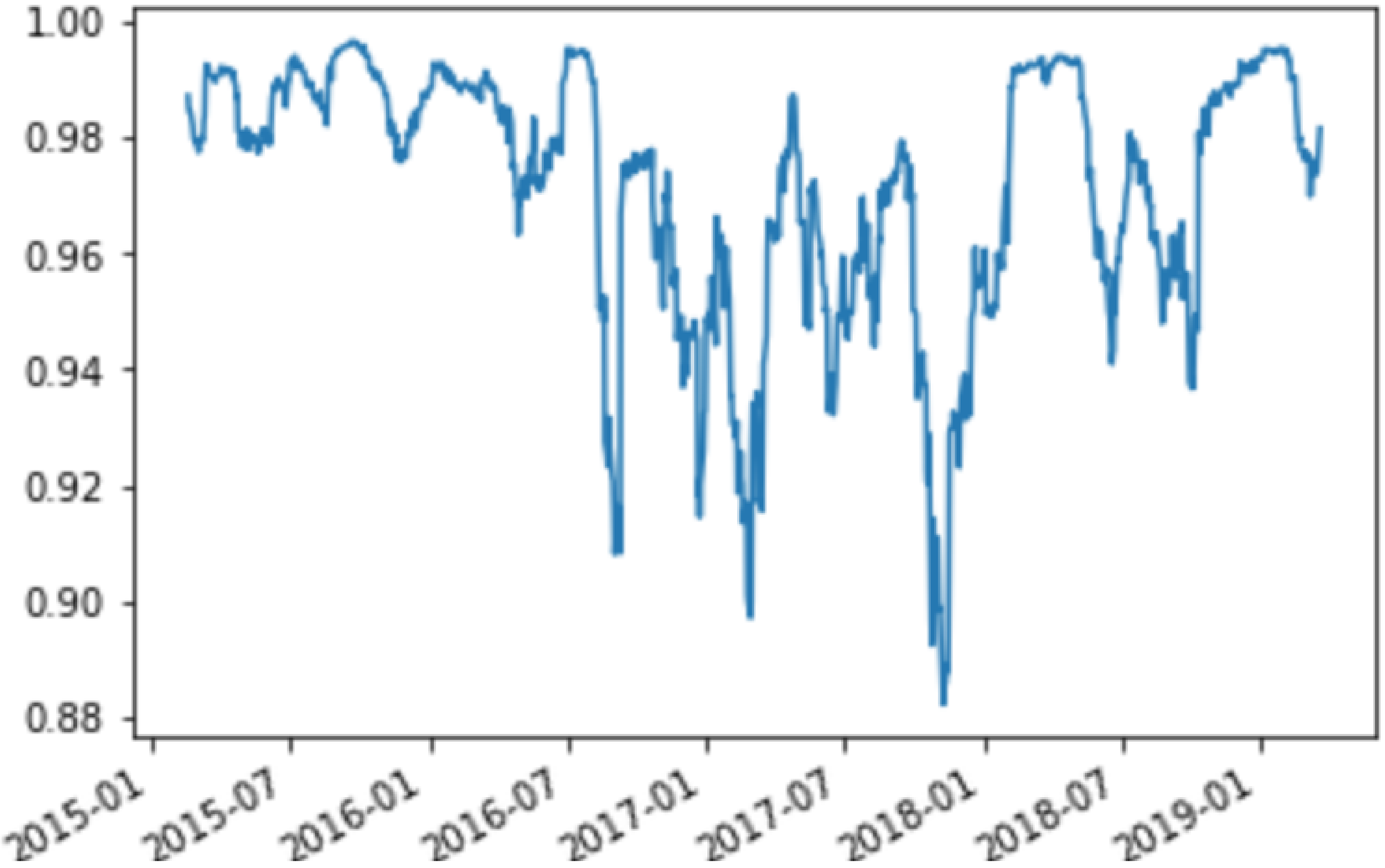
```
             portfolio   volatility   quality
portfolio     1.000000    0.056596   0.983416
volatility    0.056596    1.000000   0.092852
quality       0.983416    0.092852   1.000000
```

Correlations change over time

```
# Rolling correlation
df['corr']=df['portfolio'].rolling(30).corr(df['quality'])
```

```
# Plot results
df['corr'].plot()
```

Rolling correlation with quality



Let's practice!

Factor models

INTRODUCTION TO PORTFOLIO ANALYSIS IN PYTHON

Charlotte Werger
Data Scientist

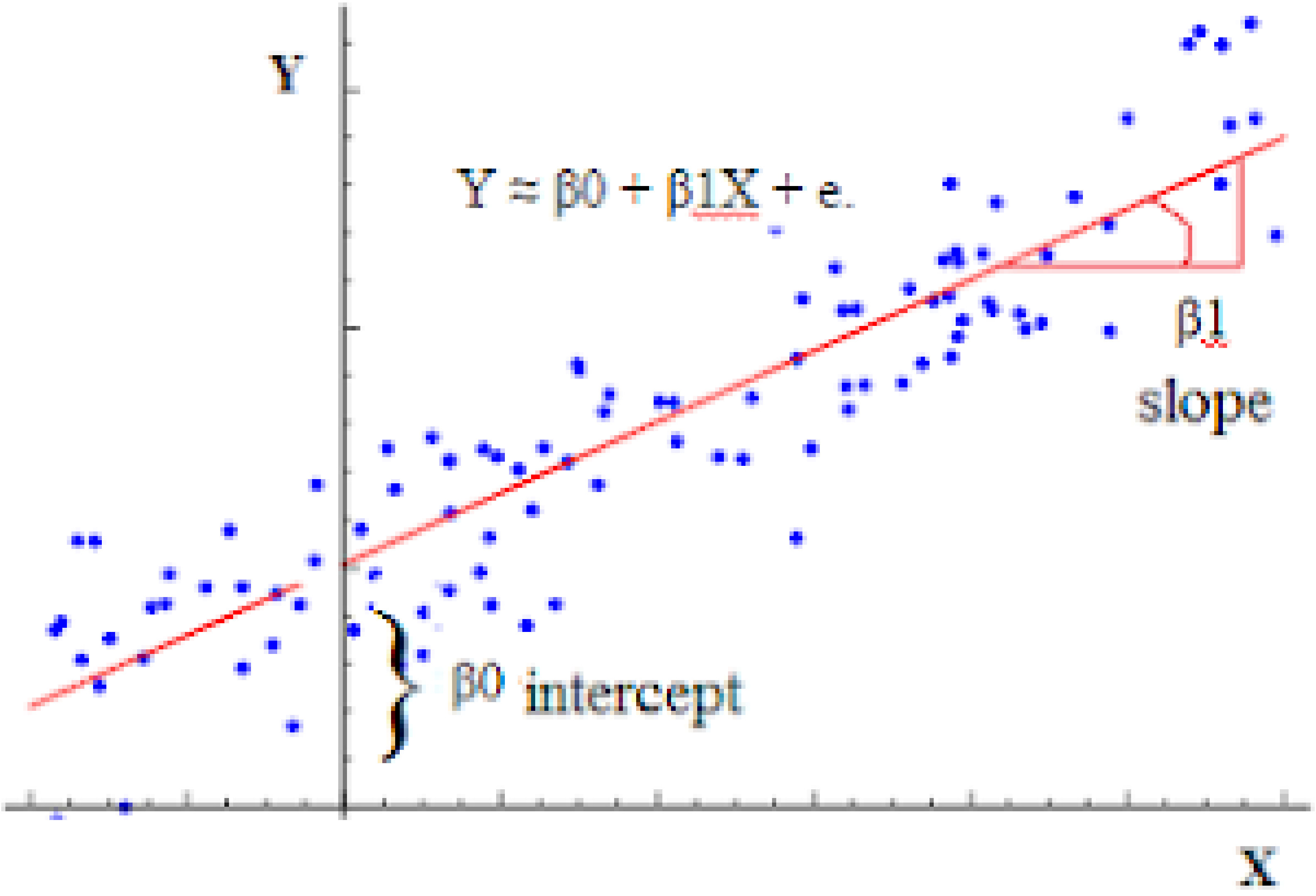
Using factors to explain performance

- Factors are used for risk management.
- Factors are used to help explain performance.
- Factor models help you relate factors to portfolio returns
- Empirical factor models exist that have been tested on historic data.
- Fama French 3 factor model is a well-known factor model.

Fama French Multi Factor model

- $R_{pf} = \alpha + \beta_m MKT + \beta_s SMB + \beta_h HML$
- MKT is the excess return of the market, i.e. $R_m - R_f$
- SMB (Small Minus Big) a size factor
- HML (High Minus Low) a value factor

Regression model refresher



Difference between beta and correlation

Beta	Correlation
"How much does factor movement X **change** your portfolios returns Y."	"How completely does factor movement **explain** your portfolio return"
Not standardised	Between -1 and 1
Strict direction: effect of X on Y only, and not visa versa.	Does not have direction, correlation between X and Y is the same as Y and X.

Regression model in Python

```python
import statsmodels.api as sm
```

```python
# Define the model
model = sm.OLS(factor_data['sp500'],
               factor_data[['momentum','value']]).fit()
```

```python
# Get the model predictions
predictions = model.predict(factor_data[['momentum','value']])
```

```python
b1, b2 = model.params
```

The regression summary output

```
# Print out the summary statistics
model.summary()
```

OLS Regression Results

Dep. Variable:	sp500	**R-squared:**	0.964
Model:	OLS	**Adj. R-squared:**	0.963
Method:	Least Squares	**F-statistic:**	3322.
Date:	Tue, 28 May 2019	**Prob (F-statistic):**	8.59e-181
Time:	19:38:35	**Log-Likelihood:**	109.08
No. Observations:	252	**AIC:**	-214.2
Df Residuals:	250	**BIC:**	-207.1
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
momentum	-0.0381	0.013	-2.896	0.004	-0.064	-0.012
value	0.9859	0.013	74.741	0.000	0.960	1.012

Obtaining betas quickly

```
# Get just beta coefficients from linear regression model
b1, b2 = regression.linear_model.OLS(df['returns'],
                                     df[['F1', 'F2']]).fit().params
```

```
# Print the coefficients
print 'Sensitivities of active returns to factors:
                    \nF1: %f\nF2: %f' %  (b1, b2)
```

```
Sensitivities of active returns to factors:
F1: -0.0381
F2: 0.9858
```

Let's practice!

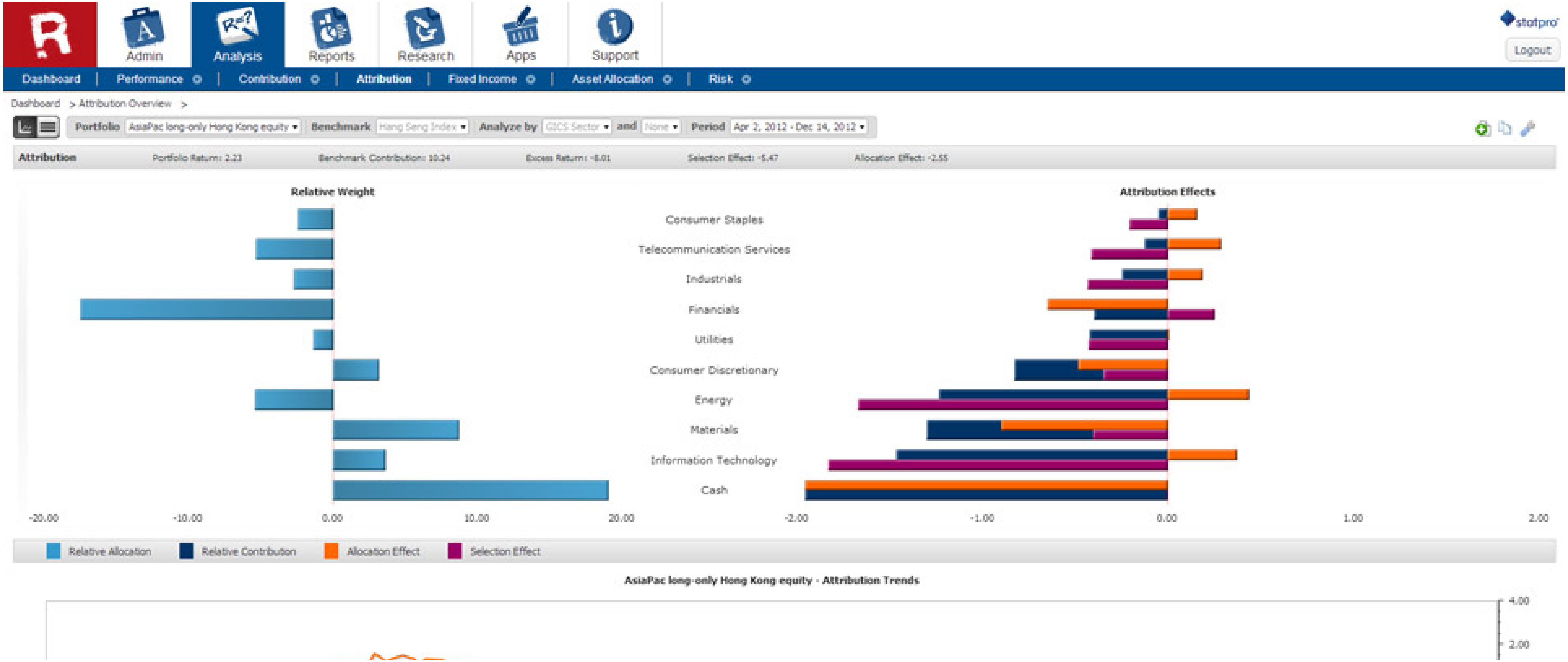
INTRODUCTION TO PORTFOLIO ANALYSIS IN PYTHON

Portfolio analysis tools

INTRODUCTION TO PORTFOLIO ANALYSIS IN PYTHON

Charlotte Werger
Data Scientist

Professional portfolio analysis tools



Back-testing your strategy

- Back-testing: run your strategy on historic data and see how it would have performed
- Strategy works on historic data: not guaranteed to work well on future data -> changes in markets

Quantopian's pyfolio tool

[1] Github: https://github.com/quantopian/pyfolio

Performance and risk analysis in Pyfolio

```
# Install the package
!pip install pyfolio
# Import the package
import pyfolio as pf
```

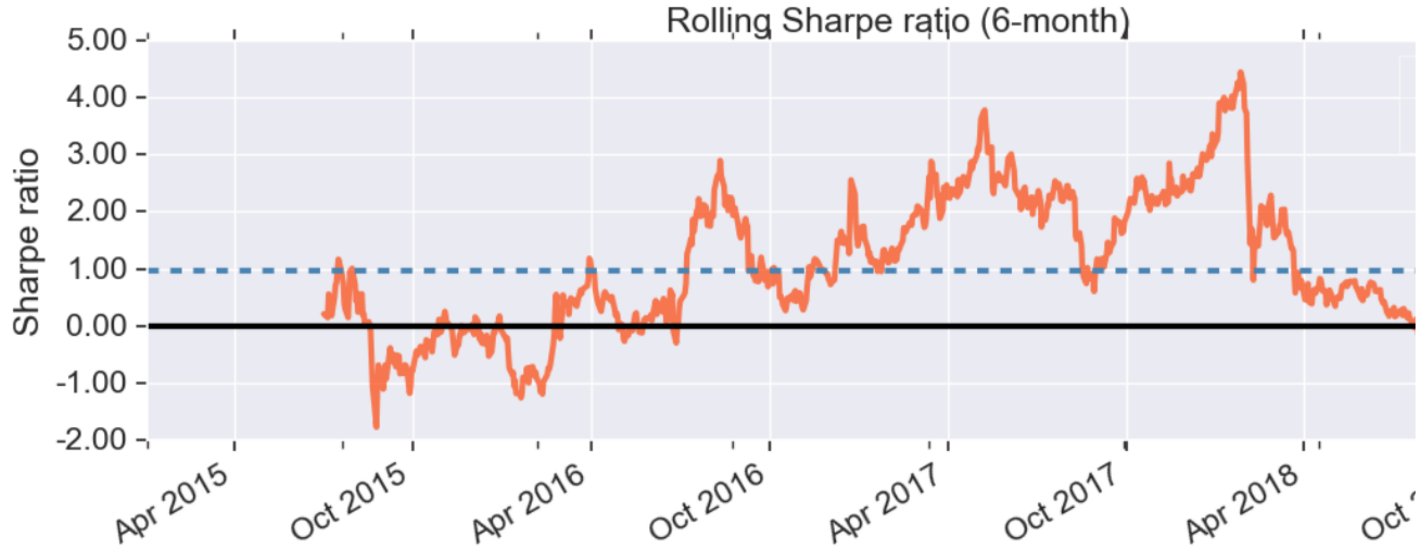
```
# Read the data as a pandas series
returns=pd.Series(pd.read_csv('pf_returns.csv')
returns.index=pd.to_datetime(returns.index)
```

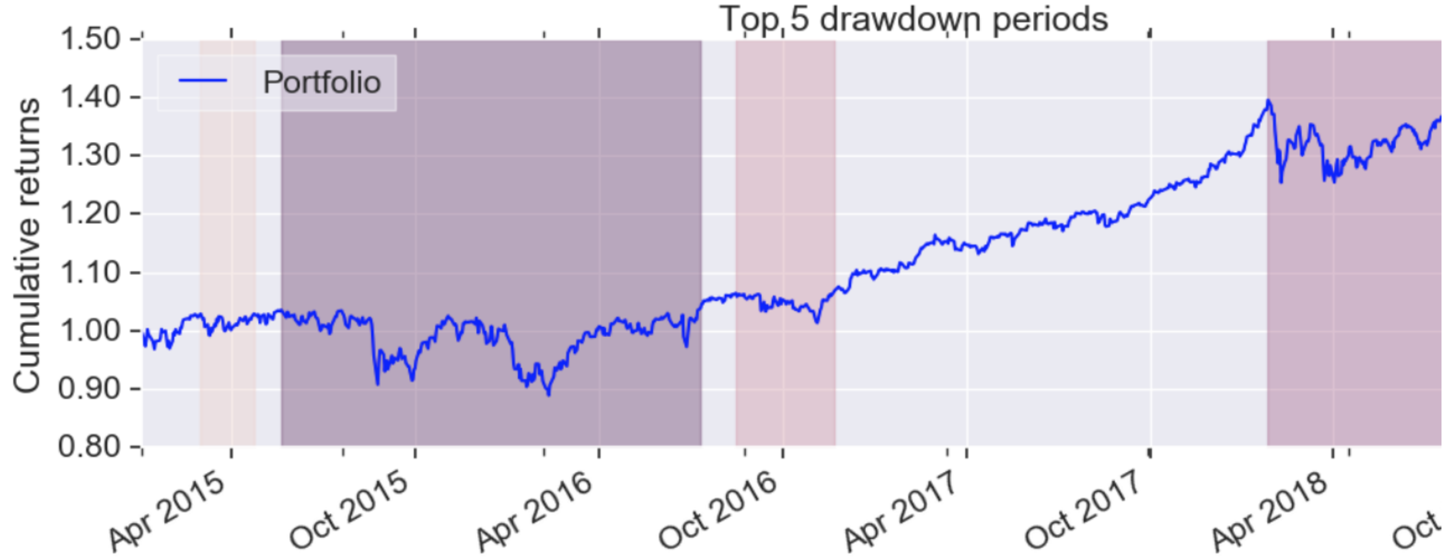
```
# Create a tear sheet on returns
pf.create_returns_tear_sheet(returns)
```

```
# If you have backtest and live data
pf.create_returns_tear_sheet(returns, live_start_date='2018-03-01')
```

Pyfolio's tear sheet

Start date	2015-01-02		
End date	2019-03-19		
In-sample months	37		
Out-of-sample months	12		
	All	**In-sample**	**Out-of-sample**
Annual return	7.9%	9.2%	4.2%
Cumulative returns	37.6%	31.9%	4.4%
Annual volatility	13.7%	12.8%	16.0%
Sharpe ratio	0.62	0.75	0.34
Calmar ratio	0.40	0.65	0.21
Stability	0.85	0.76	0.00
Max drawdown	-19.8%	-14.2%	-19.8%
Omega ratio	1.12	1.15	1.06





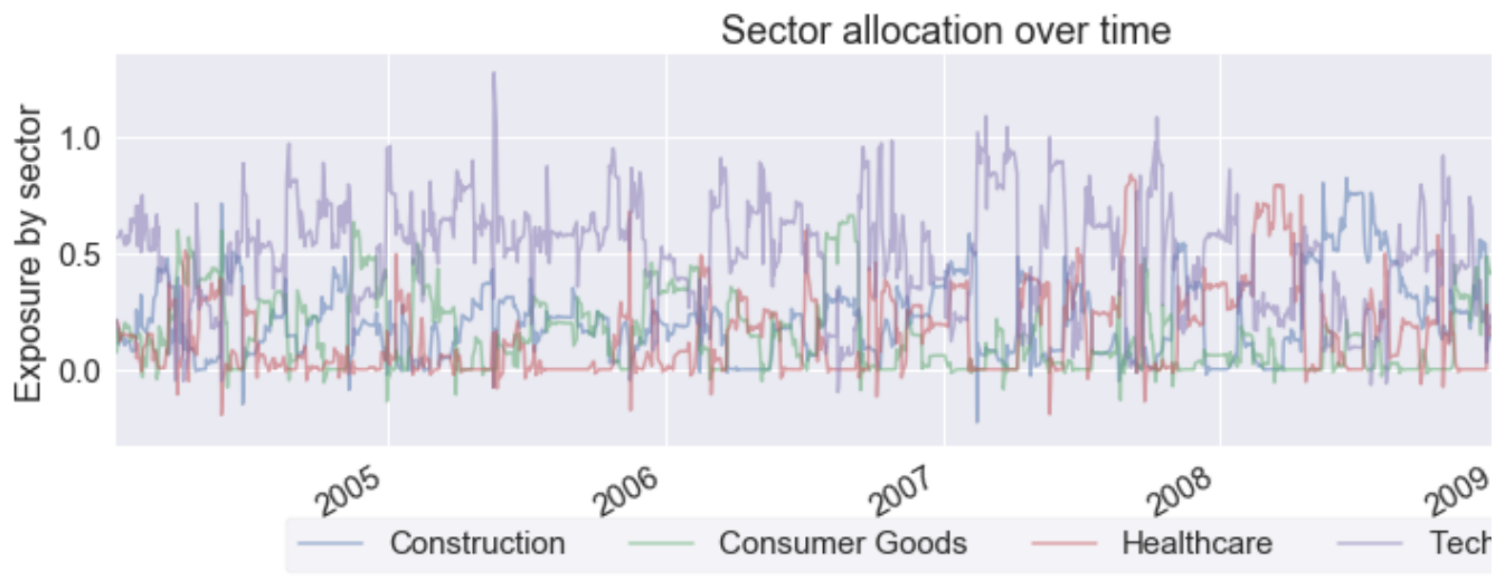
Holdings and exposures in Pyfolio

```
# define our sector mappings
sect_map = {'COST': 'Consumer Goods',
            'INTC': 'Technology',
            'CERN': 'Healthcare',
            'GPS': 'Technology',
            'MMM': 'Construction',
            'DELL': 'Technology',
            'AMD': 'Technology'}
```

```
pf.create_position_tear_sheet(returns, positions,
                              sector_mappings=sect_map)
```

Exposure tear sheet results

Top 10 positions of all time	max
COST	90.01%
DELL	85.73%
CERN	83.53%
MMM	82.09%
INTC	78.59%
AMD	75.76%
GPS	62.24%



Let's practice!

INTRODUCTION TO PORTFOLIO ANALYSIS IN PYTHON