1) Introduce quantitative trading and backtesting from a theoretical point of view

2) Show how to implement in Python a backtesting environment for simple trading strategies
Quantitative trading

► Also called *systematic trading* or *algorithmic trading*

► Creates a set of rules to generate trade signals and risk management of positions with minimal manager intervention

► Attempts to identify statistically significant and repeatable market behaviour that can be exploited to generate profits

► Low-frequency (weekly, daily) through to high-frequency (seconds, milliseconds...)
Quantitative trading system

Four major components of a quantitative trading system:

1) Strategy identification
2) Strategy backtesting
3) Execution system
4) Risk management

Focus on first two, last two won’t be covered here
Strategy identification

- Research strategies in blogs, forums, journals, etc. For example:
  - Journal of Investment Strategies
  - Quantpedia.com
  - Many more (GIYF)

- Many of these strategies are either not profitable anymore or only slightly profitable (they get “crowded” or “arbitraged away”)

- Key to making them highly profitable is to build on them, e.g. adapt them to new market conditions or optimise their parameters
Strategy identification

- Two main categories of strategies:
  - **Trend-following**: Trades on *momentum*, i.e. on the basis of the slow diffusion of information
  - **Mean reversion**: trades on the deviation of a *stationary* time series (price or spread) from its expected value

- Range of trading frequencies
  - **Low frequency trading** (LFT): days-years
  - **High frequency trading** (HFT): intraday
  - **Ultra high frequency trading** (UHFT): seconds-milliseconds

- High frequency trading requires detailed knowledge of market microstructure (how the order book and exchange work)
Once a strategy is identified, need to test its performance using historical data as well as out-of-sample data.

**Data**
- Many types: fundamental, OHLC, sentiment, news
- Many frequencies: intraday, daily
- Many instruments: equities, futures
- Many sources: many are expensive, but there are a few good free sources, e.g. Yahoo Finance, Quandl

**Qualities of good data:**
- Clean and accurate (no erroneous entries)
- Free of survivorship bias (see next slide)
- Adjusted for stock splits and dividends
Backtesting

Biases

- Biases tend to inflate performance. A backtest is likely an upper bound on the actual performance.

- **Optimisation bias**
  - Over fitting the data as a result of too many free parameters
  - Strategy will fail with real data

- **Lookahead bias**
  - Introduction of future information into past data
  - e.g. using the day’s high/low, calculating a parameter using data that would not have been available at the time

- **Survivorship bias**
  - Using only instruments which exist at present
  - Companies that went bankrupt would have made your performance worse
Backtesting

**Transaction costs**
- Backtest performance is inflated if transaction costs are not modelled appropriately

**Commissions/fees**
- A commission is paid to the broker for every transaction
- Bid-ask spread is also important, especially for illiquid instruments

**Slippage**
- Price difference between time of trade signal and time of order fill
- Depends on the volatility of the asset and the latency between the trading system, the broker and the exchange
- Especially important for HFT

**Market impact**
- Placing large orders can “move the market” against you
- May want to break the transaction into smaller chunks
The last two components of a quantitative trading system would entail a whole other talk. Very briefly:

**Execution system**
- Generates trades in real time
- Provides an interface to the broker (e.g. via an API)

**Risk management**
- Decides how to act on trade signals
- Controls leverage
- Assigns capital to trades or strategies as optimally as possible
Analysing performance

Some common measures of performance

► **Compounded growth rate**
  ► Usually annualised, gives the average annual return

\[
CAGR = \left( \frac{\text{Ending Value}}{\text{Beginning Value}} \right)^{\frac{1}{\# \text{ of years}}} - 1
\]

► **Volatility**
  ► Usually annualised, given by the standard deviation of annual returns
  ► Measure of risk

► **Sharpe ratio**
  ► Measure of reward/risk ratio
  ► Usually annualised and measured with respect to a benchmark \( b \) (e.g. risk-free rate or S&P500)

\[
S = \frac{\mathbb{E}(R_a - R_b)}{\sqrt{\text{Var}(R_a - R_b)}}
\]
Analysing performance

Some common measures of performance

- **Drawdown**
  - A period of time in which equity is below the highest peak so far
  - Can calculate *maximum drawdown* and *maximum drawdown duration*

- **Alpha, Beta**
  - Fit a straight line (*security characteristic line*) to strategy returns against the returns of a benchmark (e.g. S&P or “the market”)
  - Beta is the gradient – the variance/correlation with respect to the market i.e. gives a measure of *systematic risk* (want beta ~ 0)
  - Alpha is the intercept – the *excess return* over the market, i.e. a measure of performance (want large positive alpha)
Let’s put this into practice with Python

My backtesting code:

► Disclaimer: Very simple and incomplete
► Feel free to use it or contribute!

Makes use of pandas, numpy, and matplotlib

Employs vectorised calculations as opposed to an ‘event-loop’ (so less realistic as a simulation, but handy for doing quick research)

* Inspired by:
  * [www.quantstart.com](https://www.quantstart.com)
  * [www.github.com/quantopian/pyfolio](https://www.github.com/quantopian/pyfolio)
Components of the backtester

- **Data handler**
  - Downloads OHLC data from Quandl

- **Strategy**
  - Generates signals for each day
  - +1 long, -1 short, 0 cash (no position)

- **Portfolio**
  - Generates/rebalances positions
    - e.g. assign equal dollar weights to all assets
    - Computes returns (potentially for risk management)

- **Analyser**
  - Analyses the performance of the backtest
    - e.g. equity curve, Sharpe ratio, etc.

- Still missing: transaction costs, risk manager...
Moving average crossover

- Let’s look at a “hello world” example strategy
  - *Moving average crossover*
  - This is a momentum strategy

- Strategy rules:
  - Create two simple moving averages (SMA) of a price series with different lookback periods, e.g. 9 days and 200 days
  - If the short MA exceeds the long MA then “go long”
  - If the long MA exceeds the short MA then “go short”
backtests/macross/macross_cfg.py

► Choose trading parameters: tickers, dates, frequency, window lengths

```python
symbols = ['AAPL']
quotes = ['GOOG/NASDAQ_'] + s for s in symbols
date_start, date_end = "2010-01-01", "2015-12-31"
frequency = "daily"
datas = ['Close']
short_window = 9
long_window = 200
```

► Initialise strategy, portfolio, analyser and backtest classes

```python
strategy = MovingAverageCrossoverStrategy(short_window, long_window)
portfolio = EqualWeightsPortfolio()
analyser = [PerformanceAnalyser()]
backtest = Backtest(strategy = strategy,
                  portfolio = portfolio,
                  analyser = analyser,
)
```

► Run the backtest!

```python
backtest.run()
```
The `DataHandler` class fetches data from Quandl and returns a pandas DataFrame of prices, e.g.

```
+---------+-------+-------+-------+
<table>
<thead>
<tr>
<th>Date</th>
<th>AREX</th>
<th>WLL</th>
<th>SPY</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012-01-03</td>
<td>31.23</td>
<td>49.20</td>
<td>127.49</td>
</tr>
<tr>
<td>2012-01-04</td>
<td>31.27</td>
<td>50.25</td>
<td>127.63</td>
</tr>
<tr>
<td>2012-01-05</td>
<td>31.92</td>
<td>51.83</td>
<td>128.10</td>
</tr>
<tr>
<td>2012-01-06</td>
<td>32.04</td>
<td>51.84</td>
<td>127.82</td>
</tr>
<tr>
<td>2012-01-09</td>
<td>32.39</td>
<td>52.10</td>
<td>127.99</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2012-12-24</td>
<td>24.56</td>
<td>42.49</td>
<td>142.35</td>
</tr>
<tr>
<td>2012-12-26</td>
<td>24.73</td>
<td>43.04</td>
<td>141.75</td>
</tr>
<tr>
<td>2012-12-27</td>
<td>24.32</td>
<td>42.57</td>
<td>141.56</td>
</tr>
<tr>
<td>2012-12-28</td>
<td>24.07</td>
<td>41.67</td>
<td>140.03</td>
</tr>
<tr>
<td>2012-12-31</td>
<td>25.01</td>
<td>43.37</td>
<td>142.41</td>
</tr>
</tbody>
</table>
```

The `Backtest` class then creates empty signals and weights DataFrames that need to be filled by the `Strategy` and `Portfolio` classes, respectively.
Create a `MovingAverageCrossoverStrategy` that inherits from `Strategy`

```python
class MovingAverageCrossoverStrategy(Strategy):
    def __init__(self, short_window=None, long_window=None):
        super(MovingAverageCrossoverStrategy, self).__init__()
        self.short_window = short_window
        self.long_window = long_window
```

Implement a `generate_signals` method that fills in the `signals` DataFrame

```python
def generate_signals(self):
    super(MovingAverageCrossoverStrategy, self).generate_signal
    mavg_short = pd.rolling_mean(self.prices, self.short_window)
    mavg_long = pd.rolling_mean(self.prices, self.long_window)
    self.signals[mavg_short > mavg_long] = 1
    self.signals[mavg_long > mavg_short] = -1
```
Portfolio class

portfolios/equalweights.py

► Create a EqualWeightsPortfolio that inherits from Portfolio

class EqualWeightsPortfolio(Portfolio):
    def __init__(self):
        super(EqualWeightsPortfolio, self).__init__()

► Implement a generate_positions method that fills in the weights DataFrame

    def generate_positions(self):
        super(EqualWeightsPortfolio, self).generate_positions()
        nassets = len(self.weights.columns)
        self.weights.loc[:, :] = 1. / nassets

► If weights sum to 1, total return of portfolio is the weighted average of the assets’ returns
Analyser class

- `analysers/performance.py`

- Generic `Analyser` that computes performance measures like Sharpe ratio, drawdown etc. and makes performance plots like equity curve etc.

- Can also create and add additional `Analyser` sub-classes to the backtest
Analyser class

Performance plots

Equity curve

Rolling Sharpe ratio (6 months)

Drawdown

Top drawdown periods by magnitude

Top drawdown periods by duration

Distribution of daily returns
Outlook

- Would like to expand on this to build a more sophisticated quantitative trading system with many improvements:
  - Event-driven backtesting
  - Realistic handling of transaction costs
  - Risk management framework
  - GUI?
  - Real time execution

- As well as doing actual quant research

- Would anyone like to work on this together?
  - We could set up a *quant trading* or *quant research* arm within the club
Bibliography

- Michael H. Moore
  www.quantstart.com

- Ernest P. Chan
  Quantitative Trading: How to Build Your Own Algorithmic Trading Business

- Ernest P. Chan
  Algorithmic Trading: Winning Strategies and Their Rationale