

Quant SC

Tuesday, March 2nd

Brain Teaser

Snowflakes are falling on the ground one after another. A particular snowflake is stellar with probability p if its previous particle was also stellar, and with probability q if the previous one was not. If you catch a random snowflake, what's the probability that it is stellar?

A: $q / (1 - p + q)$

Let $x = P(\text{the snowflake caught is stellar})$

Then, $P(\text{the last snowflake was stellar}) = x$

$$P(\text{the snowflake caught is stellar}) = x = x \cdot p + (1-x) \cdot q$$

$$\text{Thus, } x = q / (1 - p + q) = P(\text{the snowflake caught is stellar})$$

Recap from Last Week

- Build something interesting and different
 - Keep it simple
- Leadership structure
- Social events
- Have fun!

Survey Debrief

- Workshops otw
- Github resources
 - Create PRs to contribute
- Speakers/recruiting
 - CMU quant event
 - outreach!

Quant Share

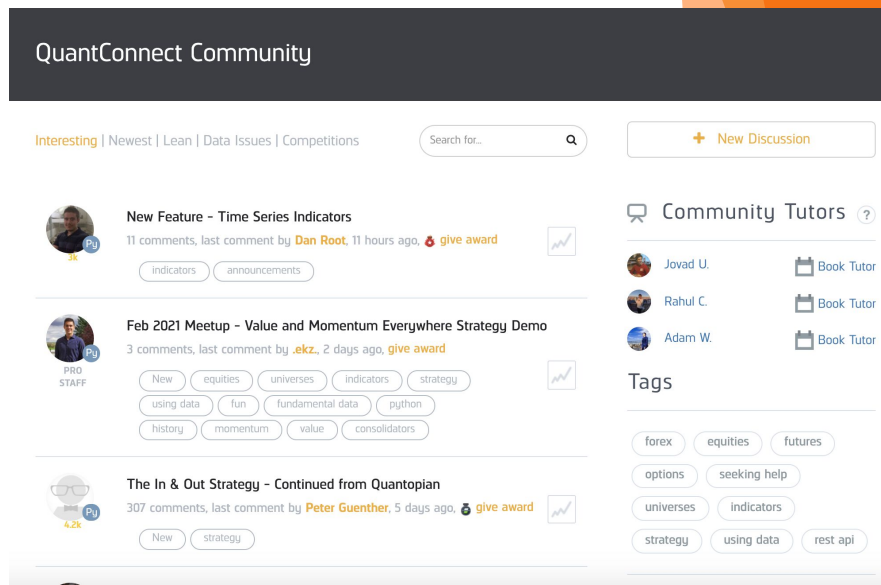
Richard Zhang

Intro to Implementation

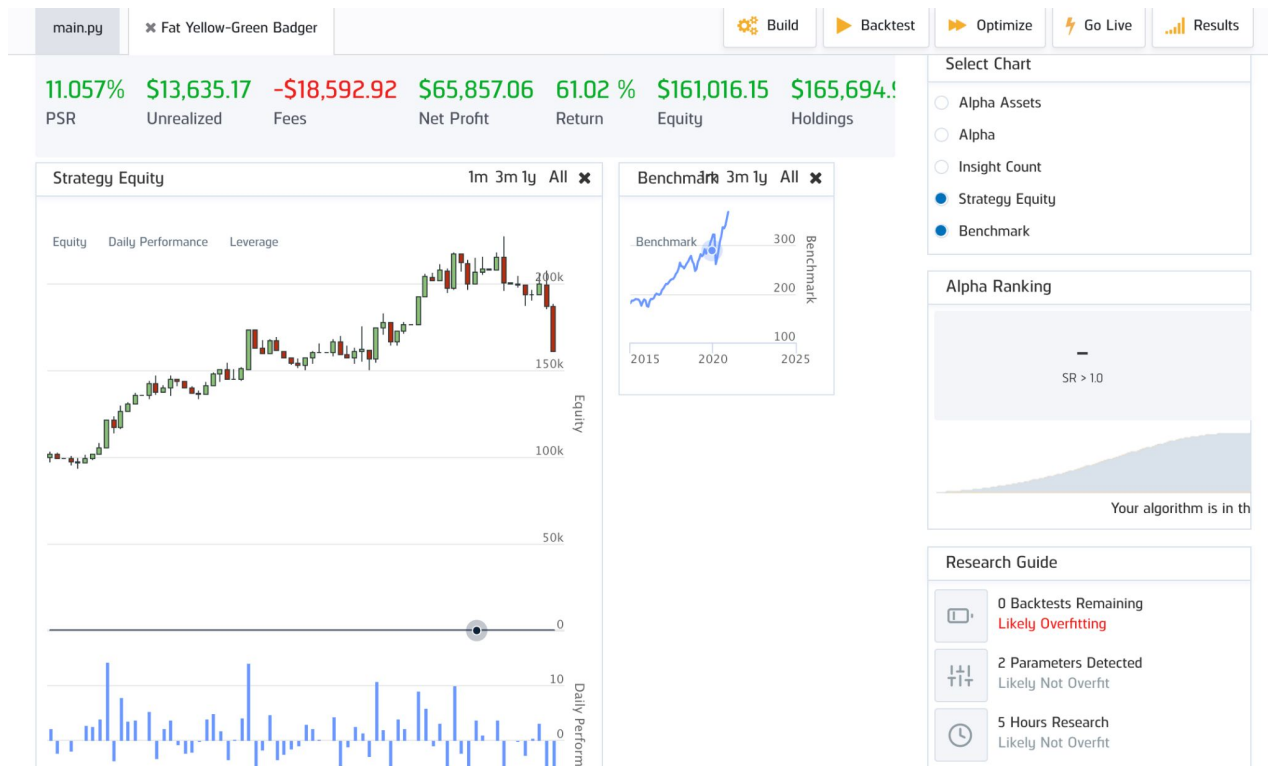
The background of the slide features abstract, flowing shapes in shades of orange and red. On the left, there are overlapping wavy bands of light orange and a darker orange. On the right, a large, sweeping shape in a vibrant red color rises towards the top right corner, partially overlapping the orange shapes.

What is Quant Connect?

- Backtesting, paper trading, live trading
- Data feed
- Forums and algorithm frameworks
- Alpha Streams



Backtesting Display



Documentation

HOME

KEY CONCEPTS

ORGANIZATIONS

ALGORITHM REFERENCE

ALGORITHM FRAMEWORK

ALPHA STREAMS


LIVE TRADING

RESEARCH

DATA LIBRARY


QUANTOPIAN MIGRATION

Documentation / Home / Home




Documentation

Learn to use QuantConnect and Explore Our Features




Using QuantConnect

Learn the basics of working in the terminal




Algorithm Framework

Reusable modular code to fast track design




Tutorials

Series of written introductions to python and finance




Using LEAN

Go deep into the engine powering QuantConnect




Algorithm Reference

Reference to building an Algorithm




Research

Interactive Jupyter development API



Live Trading

Harness live-specific features and trade on your brokerage



Organizations on QuantConnect

Build your organization from QuantConnect's foundation

A Basic Algorithm Framework

1. Universe Selection
 - Pick assets to trade
2. Alpha Creation
 - Create trading signals
3. Portfolio Construction
 - Set the targets for each asset to hold
4. Execution
 - Execution method to reach targets
5. Risk Management
 - Set logic to liquidate elements of portfolio during poor performance

Initialization

Topics to consider

- Start/End dates for backtesting
- Starting capital
- Positions, position concentration
- Scheduled functions
- Equities to track

Data

Options

AddOption

OptionChain

- AddOptionContract

- GetOptionContractList

SetFilter

Fundamental

Financial Statements

Industry Classifications

Third Party API Requests

Prices

AddEquity

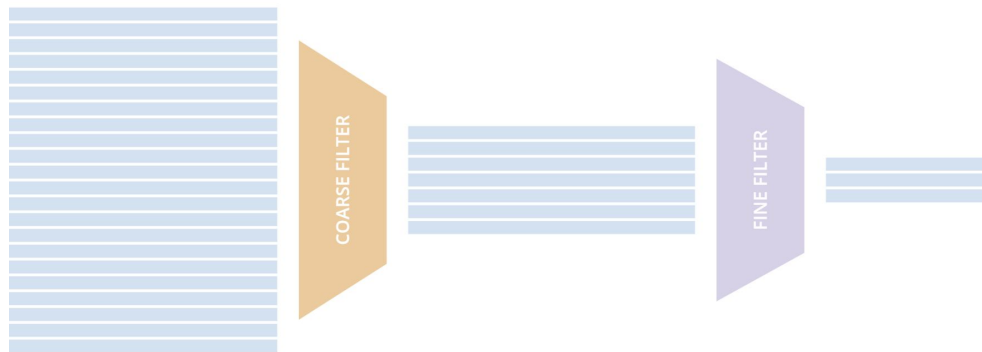
Universe Selection

Universe Selection

AddUniverse

Coarse and fine selection

AddEquities



Trading

Explicit orders

SetHoldings(ticker, weight)

Liquidate(ticker)

Order Type	Summary
Market Order	<code>self.MarketOrder("IBM", 100)</code> Send a market order for 100 IBM at market price
Limit Order	<code>self.LimitOrder("IBM", 100, 21.67)</code> Submit a limit order for 100 IBM @ \$21.67
Stop Market Order	<code>self.StopMarketOrder("IBM", 100, 21.67)</code> Submit a stop limit order for 100 IBM with stop price of \$21.67
Stop Limit Order	<code>self.StopLimitOrder("IBM", 100, 21.67, 22.00)</code> Stop limit order for 100 IBM, stop price \$21.67, limit of \$22.00
MarketOnOpenOrder	<code>self.MarketOnOpenOrder("IBM", 100)</code> Market on open order for 100 IBM
MarketOnCloseOrder	<code>self.MarketOnCloseOrder("IBM", 100)</code> Market on close order for 100 IBM

OnData

- Rebalancing & Risk Management
 - New stocks, new weights
 - Error-checking
 - Delisted stocks
 - (Trailing) stop losses

```
self.AddEquity("IBM", Resolution.Hour) ## Subscribe to hourly TradeBars

def OnData(self, data):
    ## You can access the TradeBar dictionary in the slice object and then subset by
    symbol
    ## to get the TradeBar for IBM
    tradeBars = data.Bars
    ibmTradeBar = tradeBars['IBM']
    ibmOpen = ibmTradeBar.Open      ## Open price
    ibmClose = ibmTradeBar.Close    ## Close price

    ## Or you can access the IBM TradeBar by directly subsetting the slice object
    ## (since you are subscribed to IBM equity data, this will return a TradeBar rather
    ## than a QuoteBar)
    ibmOpen = data['IBM'].Open      ## Open price
    ibmClose = data['IBM'].Close    ## Close price
```


Flags and Scheduling Pt. 2

Use scheduling functions to set flags, then use flags to add greater control of flow to your program

Ex: say you want to run a function weekly to liquidate your positions.

To do so, you can run a flagging function at the end of every day and build in logic to check if it's the end of the week

Then, in a liquidation function running every day, check if you should liquidate based on whether the flag is set

This can provide you with low-effort synchronized control over multiple functions

Team Reveals

The background features abstract, flowing shapes in shades of orange and red. On the left, there are several overlapping, semi-transparent orange shapes that curve upwards and to the right. On the right side, there are similar shapes in shades of red and pink, also curving upwards and to the left. The overall effect is a dynamic, modern design with a warm color palette.

Team 1

Lead: Rohan Sanjay

Kangmin

Alex G

Ryan

Team 2

Lead: Kai

Sagar

Alex B

Jennah

Team 3

Lead: Adam

Avi

Anuj

Team 4

Lead: Rohan Sanjay

Grace

Carson

Athena

Team 5

Lead: Steve

Tina

Rithik

Richard

Deliverable

Meet with your group and decide on what project you want to work on!