OEC Programming Challenge

StockBots Challenge

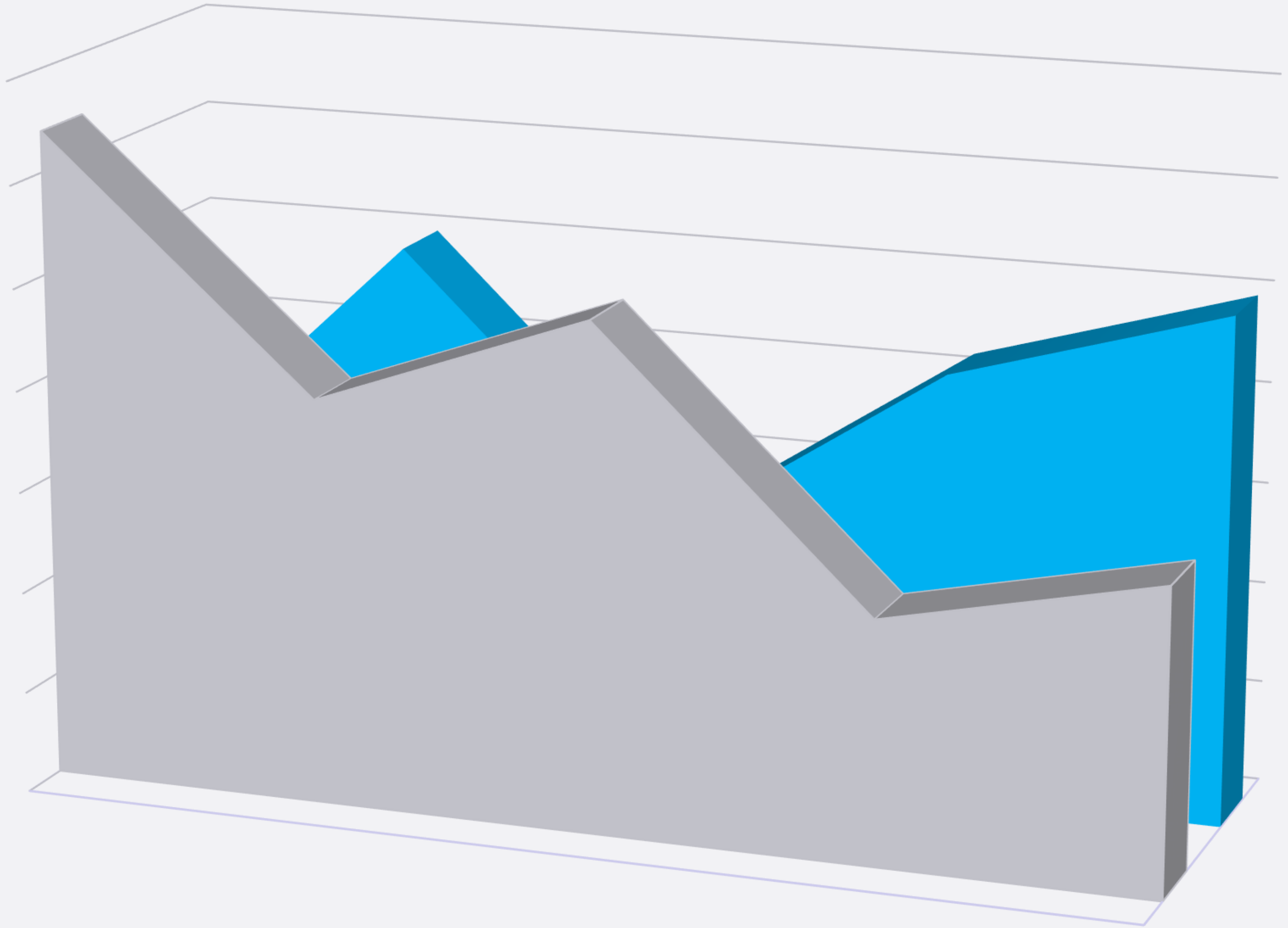
School: Western University

MUSTANG CAPITAL

EST. 2017

Adam Kovalcik
Colin Nowers
Colin Russell
Dane Sisinni

environment introduction



equity trading

Able to purchase, and only purchase, equities in a market.

unknown behaviour

Unknown underlying price and return behaviour algorithm in market simulator.

cash

One of a few market players with $100,000 in deployable capital.

problem introduction

Create an automated trading strategy that can maximize returns while minimizing risk.

our development procedure

01

get a jump start

Theory: behaviour of market is based on supply and demand so if we can be the first to market, our value will be propped up by others demand.

Result: good move in hindsight but only two others clearly invested significant capital.

our
development procedure

02

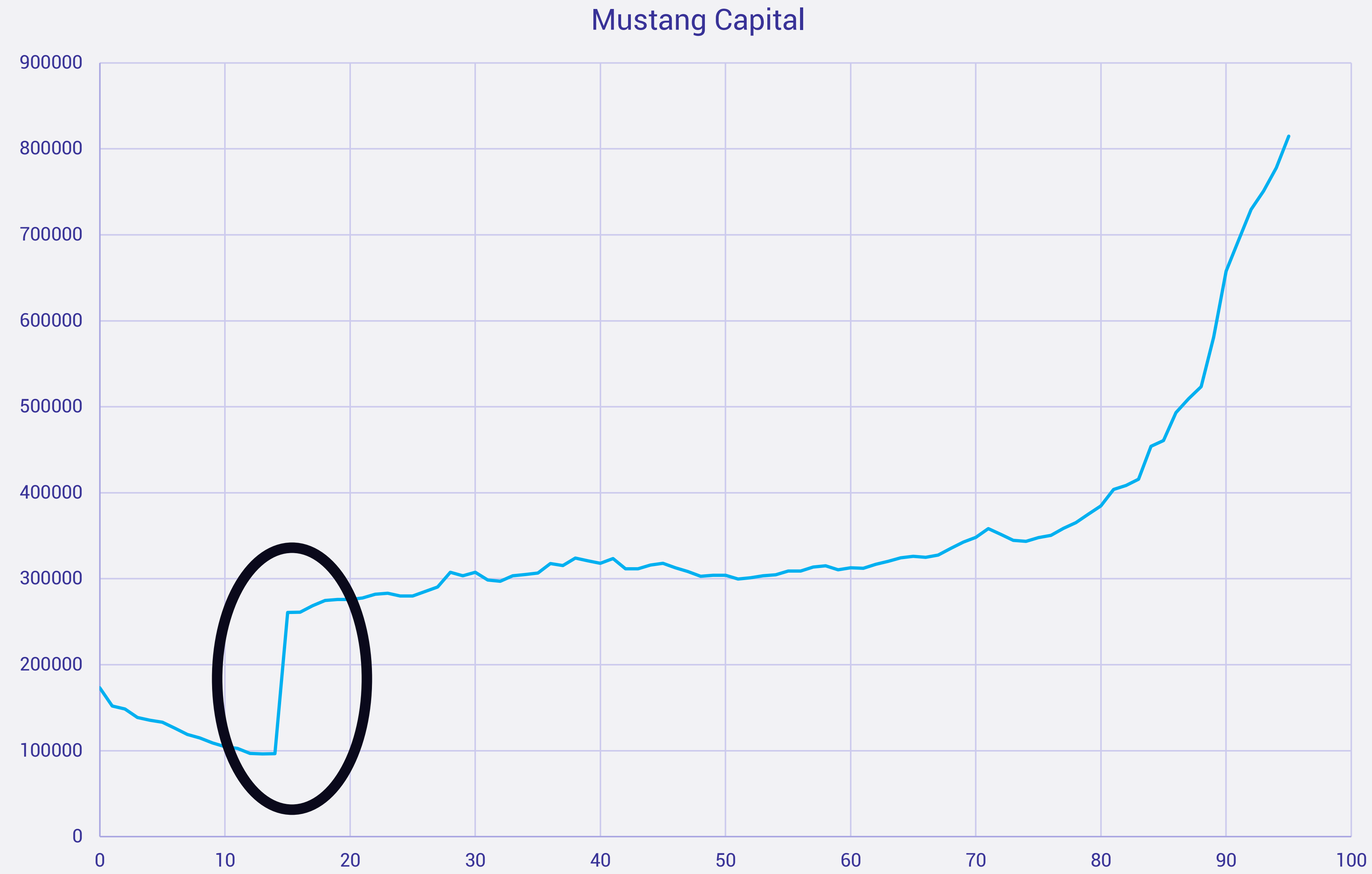
test underlying market machinery

Theory: flooding the market with investment should prop up equity prices, flooding the market excess shares through divestment should reduce equity prices.

Result: based off only one test point, major investment significantly propped up equity values, major divestment had no apparent effect.

our
development procedure

◀ 02



our
development procedure

03

diversifying to reduce risk

Theory: by having wide spread holdings of the market, much like an index, and arbitrarily rebalancing at a specific frequency ensures that we minimize volatility.

Result: volatility, especially relative to other trading strategies was very low and the rebalancing did not create or destroy significant value.

our development procedure

04

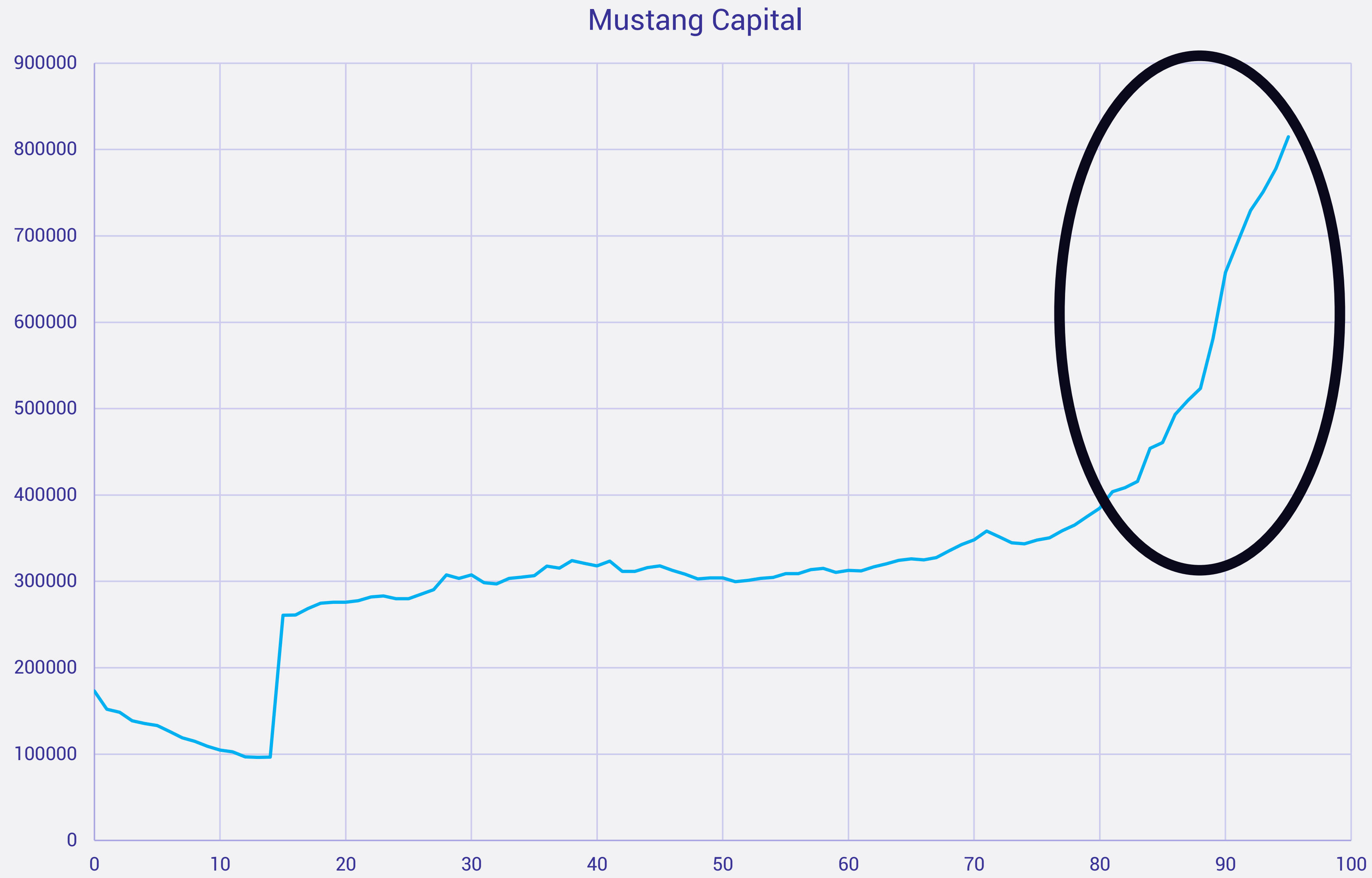
harnessing stable returns

- Theory: employing a stop-loss and stop-gain strategy will create value between portfolio rebalances. A 3:1 ratio will be used

- Result: returns appeared to be consistent in a seemingly random market

our
development procedure

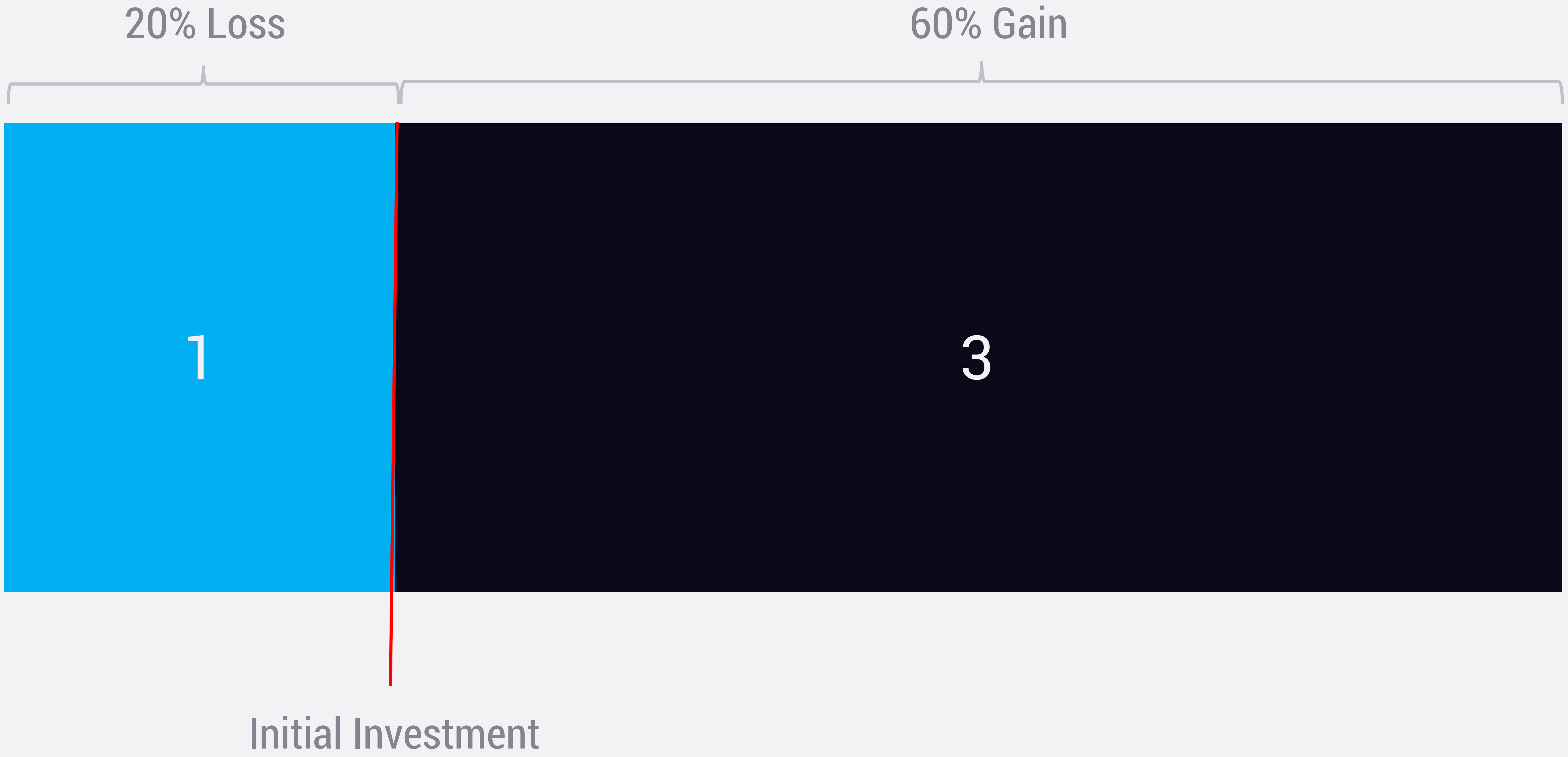
◀ 04



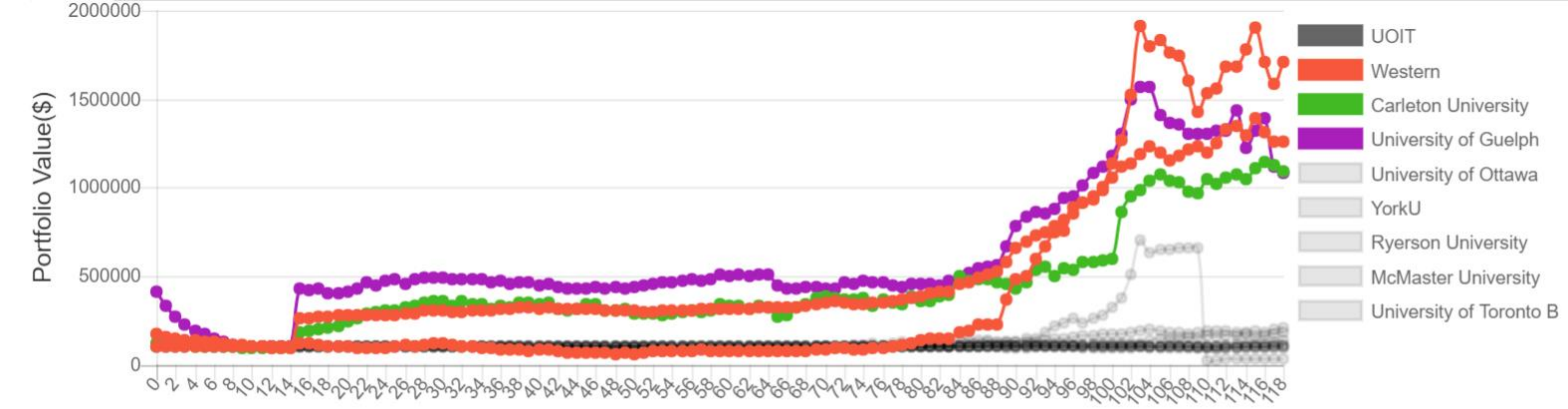
trading philosophy

- Identifying a philosophy prior to assembling the trading algorithm to keep the investment thesis consistent
- Three to one ratio of take profit to stop loss representing the risk profile of the portfolio
- The results of a consistent philosophy and 3:1 ratio is three fold:
  - 1) Reduced Volatility
  - 2) Reduced Risk
  - 3) Consistent Results

algorithm
logic



overall results



how the system works

01 market API

A market API was provided which linked to generated market data.

02 java backend

A java backend connected to the market API sending HTTP requests that gets data, performs our algorithm, and sends updates back to the market simulator

03 database

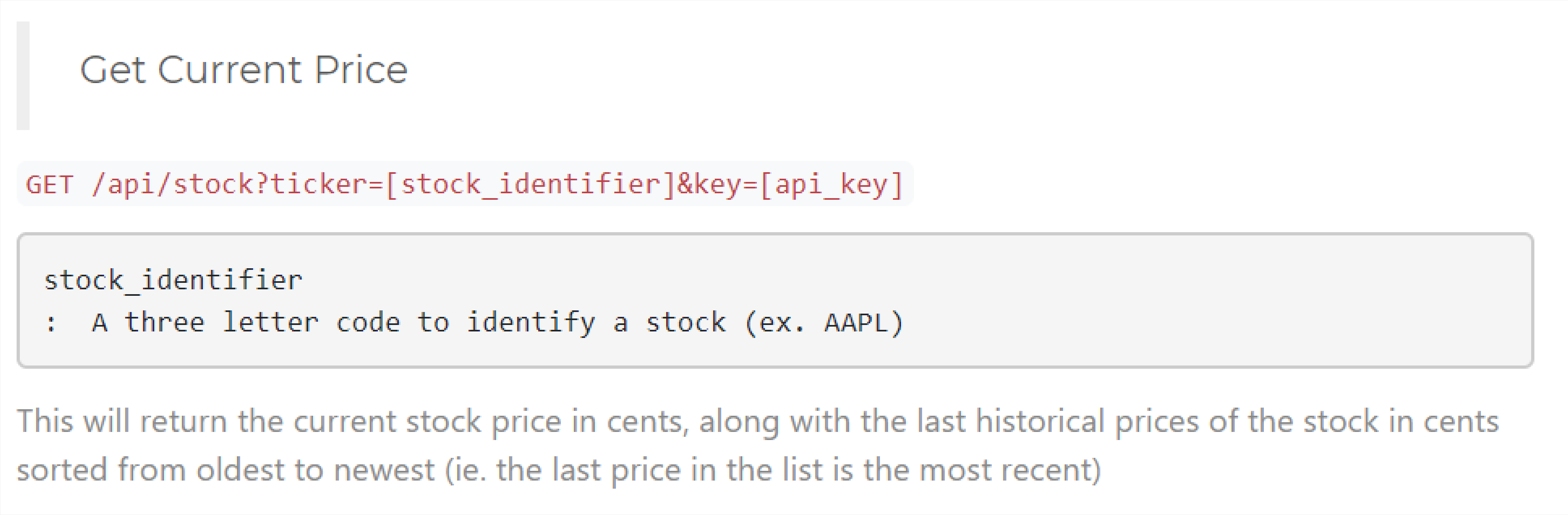
Storing important stock data for tracking what stocks fell past the 20% stop loss

04 javascript front-end

Client side scripting providing users of our trading algorithm with an easy experience to take the complication out of investing

market API

- A thorough application programming interface for HTTP requests used for communication with our back end



Get Current Price

```
GET /api/stock?ticker=[stock_identifier]&key=[api_key]
```

```
stock_identifier
:  A three letter code to identify a stock (ex. AAPL)
```

This will return the current stock price in cents, along with the last historical prices of the stock in cents sorted from oldest to newest (ie. the last price in the list is the most recent)

01

java backend

- Java Spring Framework
- Trading Bot running on separate thread
- REST API for UI data integration
- Authentication – OAUTH2, JWT tokens
- Libraries: Guava, Apache Commons, JSON Simple, Apache HTTP Client

02

03

database

- Store stock data
- Track of stop/loss trades

PostgreSQL

04

- HTML visualization of performance
- Dynamic jquery and HTTP request to acquire market data
- Secure authorization and future investment management functionality
- Future integration with news and other market data

javascript front-end

pulling
it together with a server

PostgreSQL
Java™
spring
by Pivotal™
GUAVA™
Apache
SOFTWARE FOUNDATION
IJ

demo time.

q&a