2018 Fall Competition
Case Packet

traders@mit.edu
traders.mit.edu

October 16, 2018

Contents

1 Introduction

Traders@MIT welcomes you to our 11th Annual Intercollegiate Trading Competition! We are excited to be holding the competition again this year and are confident that it will provide a rich and challenging learning experience. The competition will be held on Sunday, 11/11, and competitors will also be required to attend networking events held on Saturday, 11/10. All travel to and from the competition will be fully reimbursed.

Our competition consists of electronic trading. Teams will be ranked overall based on their total weighted rankings from each of the three cases. You will also have several opportunities to speak with our attending sponsors.

There will be networking on 11/10 from 3:30 PM - 10:30 PM, and the competition day (11/11) itself will have programming from 8:30 AM - 6:00 PM. After the competition, DRW will take selected teams out to dinner.

Our sponsors for this year's competition are:

Diamond: DRW Trading, Citadel, SIG, Schonfeld Strategic Advisors LLC

Platinum: Wolverine Trading, Barclays, Five Rings Capital

Gold: D.E. Shaw, Hudson River Trading, Jane Street Capital, Group One Trading, IMC Financial Markets, Old Mission Capital, Belvedere Trading, Quantlab

Silver: Flow Traders, Vatic Labs, TD Securities, Volant Trading

The three cases will be: Algo S&T, the Barclays Options case, and Five Rings Capital Day-Of case. The Algo S&T and Barclays Options cases will be algorithmically traded using our in-browser trading platform known as Mangocore. More information about Mangocore is in the last section of this packet. In particular, competitors *must* algorithmically trade and cannot click trade. You will be expected to to submit your code for the competition before the competition day. More details on the two cases will be in the following sections of the case packet.

The day of case will involve quantitative modeling a set of price signals. Relatively little advance preparation is expected, but we'd recommend familiarizing yourself with statistical analysis packages in `Python` and installing relevant packages on the laptop you'll be bringing to competition—more information can be found in the case description.

1.1 Schedule of Events

Note that all events in this schedule are **mandatory** for competitors to attend if they wish to receive travel reimbursements and be eligible for prize money; attendance will be taken. If this is not possible for you, then please email `traders-exec@mit.edu`, with a detailed reason as to why you have a conflict with any of the events.

Saturday, November 10

- 3:30PM – 5:30PM: Billiards with Schonfeld Strategic Advisors LLC
- 6:00PM – 8:00PM: Networking + Games with Citadel
- 8:30PM – 10:30PM: Networking + Games with Susquehanna International Group

We will have transportation arranged from MIT to the first networking event starting at 3pm. Buses will be departing from Maseeh Hall (305 Memorial Drive, Cambridge, MA 02139). We will provide transportation between each of the following events and back to Maseeh Hall after the final event finishes at 10:30 PM.

Sunday, November 11

8:30AM	–	9:00AM	Check-In and Breakfast at MIT Media Lab
9:00AM	–	9:30AM	Opening Remarks
9:30AM	–	10:45AM	Case 1: Barclays Options
10:45AM	–	12:00PM	Case 2: Algo S&T
12:00PM	–	1:30PM	Sponsor and Competitor Lunch
1:30PM	–	4:00PM	Case 3: Five Rings Capital Day-Of Case
4:00PM	–	5:00PM	Networking
5:00PM	–	5:30PM	Closing Remarks
5:30PM	–	6:00PM	Awards Ceremony

1.2 Housing and Transportation

We will be providing out of town competitors with housing on Saturday night. All competitors should have already been contacted about arrangements. We will also be reimbursing all travel for out of town competitors. Details regarding reimbursement will be communicated as necessary.

1.3 Directions

The competition will be held at the MIT Media Lab this year, which is located at the following address:

Floor 6
MIT Media Lab (E14)
75 Amherst St,
Cambridge, MA 02139

1.4 Piazza

We have created a Piazza forum to answer questions about cases and any other questions you may have. **Almost all of our updates and announcements will be placed here, so it is in your best interest to be on this forum. We will be answering all questions through this forum as well and not by email**:

piazza.com/tradersmit/fall2018/tamit2018/home

The access code is **tamit2018**. Traders@MIT executive board members will do our best to reply to all questions within 24 hours.

1.5 Attire

Both the Friday networking session and the Saturday competition are business casual. Our Gold, Platinum, and Diamond sponsors will be in attendance.

2 Barclays Options Case

2.1 Overview

In this case, you will algorithmically trade European options on an index based on three different equities. Your ability to earn profits will depend on your ability to:

1. Exploit inconsistencies in implied volatility across different strike prices
2. Market make when spreads are large, without becoming overly exposed to risk
3. Hedge positions to reduce risk

This case is relatively complex. We strongly recommend investing time in understanding the relevant concepts and building a useful model before the competition.

For the rest of this case description, we will simply refer to European options as options.

2.2 Options Basics

Options come in two flavors: A *call* option confers the right to buy a given product (the underlying) at a fixed price (the strike price) on – but not before – a given date (the expiration date). Likewise, a *put* option gives the right to sell the underlying at the strike price on the expiration date.

For example, imagine that an investor is holding an Apple call option at a strike of $100 and expiry of November 15th. (S)he would exercise the right to buy if the price of Apple was greater than $100 on November 15th.

The price or value of an option depends on the volatility of the underlying. *Volatility* is the standard deviation of the logarithmic return distribution of the underlying. Intuitively, it is a way to quantify how much the price will fluctuate. There are two relevant measures of volatility:

- Realized volatility is the volatility of the underlying over some past time period and is generally estimated as the sample standard deviation of the annualized log returns during the period.

- Implied volatility is the volatility of the underlying that when used in a pricing model returns the current market price of the option. We will explore this concept more later.

Options can be either in-the-money, out-of-the-money, or at-the-money. In-the-money refers to an option that is profitable if it were to be exercised. Out-of-the-money is defined analogously. At-the-money refers to an option that is whose strike price is at the spot price; i.e. the option would break even if it was exercised today.

2.3 Options Pricing Models

Given the current price of the underlying (the *spot* price) S, strike price K, risk-free interest rate r, volatility σ, current time t, and expiration time T, the Black-Scholes formula for the value C of a call option on a non-dividend-paying asset is

$$C = N(d_1)S - N(d_2)Ke^{-r(T-t)}$$

where

$$d_1 = \frac{\ln(\frac{S}{K}) + (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}}$$

$$d_2 = d_1 - \sigma\sqrt{T-t}$$

and $N(x)$ is the standard normal cumulative distribution function. A derivation of this model can be found here: http://www.math.cuhk.edu.hk/~rchan/teaching/math4210/chap08.pdf. Briefly, the Black-Scholes formula and related extensions assume the spot price follows a geometric Brownian motion, which is equivalent to claiming that incremental logarithmic returns are normally-distributed and have a constant standard deviation. In reality, returns are not typically normally distributed. They tend to have a higher probability of extreme events than would be the case under a normal distribution ("fat tails"), particularly for extreme negative events. The Black-Scholes formula does give a fast and relatively accurate approximation, and is widely used as a basic model. The value P for a corresponding put option can be obtained from put-call parity, which for European options is as follows:

$$C + Ke^{-r(T-t)} = P + S$$

The intuition behind this formula is as follows: Ignoring for the moment the interest rate (which will be $r = 0$ during the competition), the asset package on the left side of this equation (i.e. one call option plus the dollar value of the strike) will pay out a guaranteed K, plus an additional bonus value if the option is in-the-money equal to the difference between the price of the underlying and the strike price. Therefore the left side will pay out the larger of K and the price of the underlying at expiry. Meanwhile, the right side (comprised of a put P and a future S) will pay out the value of the underlying, plus additionally the difference between the strike price and the

underlying if the put option is in-the-money. Take a moment to understand why these are equal; the interest rate factor simply accounts for the time value of money.

2.4 Volatility Curves, Smile and Skew

Under the assumption that the options pricing follows Black-Scholes, we can invert the above formula for a given strike price and expiry time to deduce the implied volatility. Perhaps surprisingly, options with different strike prices often have different implied volatilities. Typically options that are more out-of-the-money or in-the-money will have higher implied volatilities, because the actual distribution of the returns of the underlying is more fat-tailed than the normal distribution. This effect is known as the *volatility smile*. Moreover, options with strike prices much lower than the current spot price often exhibit higher implied volatility than options with strike prices equally higher than the current spot price. This effect is known as the *volatility skew*. The best way to visualize this is by plotting the implied volatility against the strike price; this is known as the volatility curve. Figure 1 shows the effect of the October 1987 stock market crash on the volatility curve.

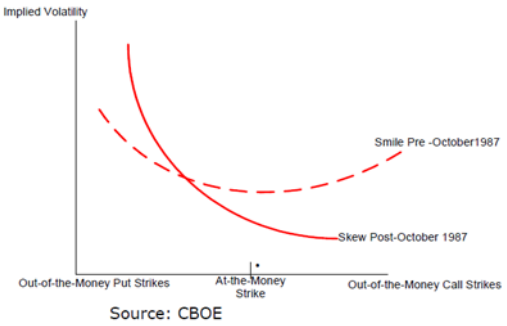
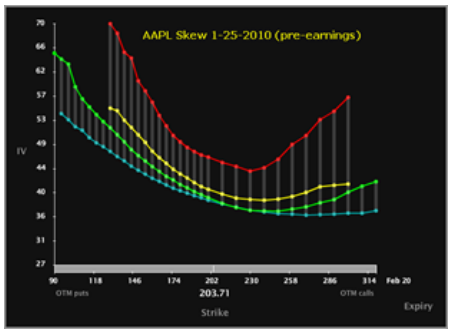


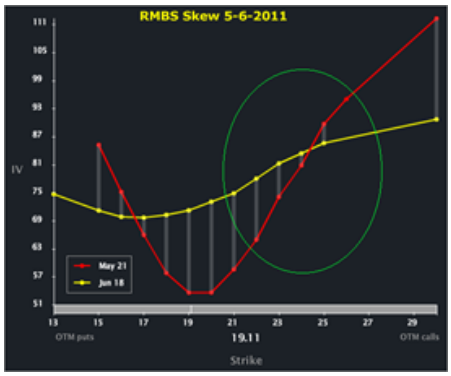
Figure 1: The S&P 500 Implied Volatility Curve Pre- and Post-1987

As seen above, an increase in the perceived likelihood of extreme negative events can increase volatility skew. Similarly, a belief of increased likelihood in extreme events in either direction compared to more typical events unchanged can make the volatility smile more pronounced. Note however, that these effects are *not* equivalent to that of belief that price movements will on average be larger than they have been in the recent past; this instead causes a parallel shift upward in the implied volatilities of at all strike prices. These effects can occur simultaneously.

Below, we show two examples of changing volatility curves. The left is from just prior to an earnings release for a stock. Note the pronounced curvature and asymmetry in the front-month options (red). Also note that the overall level of volatility for the front-month options is higher than for longer-dated options, indicating that most of the variability in price is expected to occur in the near-term:

In contrast, the chart on the right comes from a company that engages in numerous patent lawsuits. A particular one is expected to reach a decision soon, and the decision





will be either very good or very bad for the future of the company. The front-month options again exhibit a substantial volatility smile, but it is more symmetrical this time. Also, subsequent volatility remains high in this example given the risks of other ongoing litigation and R&D. However, these later risks are expected to be less binary than the impending decision: market participants predict the later return distribution will have thinner tails, so the implied volatility surface is flatter.

2.5 Greeks and Hedging

Option prices are a function of multiple variables, including the price of the underlying, the volatility of the underlying and time. We can differentiate with respect to these variables to get various quantities, commonly called the "Greeks." The Greeks describe the effects of changes in various parameters on the price of the option. We can think of the Greeks as representing exposure to market risk along different axes. Hedging is the process of minimizing such risk across each Greek.

The simplest and most important Greeks are:

- Delta: the rate of change of an option's value relative to a change in the underlying. It is always positive for calls and always negative for puts. Traders can think of delta as the impact of the movement of the underlying on the value of their portfolio; therefore, it is also equivalent to how many shares of the underlying to sell in order to hedge their option against the movement of the underlying. Having a portfolio with a small (in absolute value) delta corresponds to not having an opinion as to whether the underlying will move up or down. The value of delta for a call option is always between 0 and 1, and the values of delta for put and call options at the same strike price can be related via put-call parity (since the derivative of the underlying with respect to itself is clearly 1). Intuitively, for call options we should expect that delta should approach 1 for very in-the-money options and 0 for very out-of-the-money options; this is indeed the case.

- Gamma: the rate of change of delta relative to a change in the underlying. Since delta increases from 0 to 1 as the underlying passes the strike price, gamma is always positive for call options; since the delta for put options is simply 1 less

than that for call options via put-call parity, gamma is the same for put and call options. Gamma determines how quickly you must adjust a delta hedge when price changes.

- Vega: the rate of change of an option's value relative to the (implied) volatility of the underlying. Vega is positive for every option. Intuitively, this is because extremely out-of-the-money options result in a fixed loss, whereas extremely in-the-money options can have large payouts proportional to exactly how in-the-money they are; hence, as volatility and hence the likelihood of both extremely good and extremely bad events increases, the options price should only reflect the added value of the extremely good outcomes and will therefore increase. Vega tends to be higher for options with later expiry dates and for options with a strike price near the current spot. A trader holding a portfolio with a small (in absolute value) vega has no opinion about whether the volatility of the underlying will go up or down across all strike prices; however, (s)he may still have an opinion about the shape of the volatility curve (which will be discussed in the next section). The vega for the underlying will be zero.

Expressions for the Greeks from differentiating Black-Scholes are below, assuming an interest rate of 0. Note these can become unstable for options very close to expiry: where $N(x)$ is the standard normal cumulative distribution function and $N'(x)$ is the

Table 2.1: Formulas for Greeks

Greek	Calls	Puts
Delta	$N(d_1)$	$N(d_1) - 1$
Gamma	$\frac{N'(d_1)}{S\sigma\sqrt{T-t}}$	$\frac{N'(d_1)}{S\sigma\sqrt{T-t}}$
Vega	$SN'(d_1)\sqrt{T-t}$	$SN'(d_1)\sqrt{T-t}$

standard normal probability density function. Successful competitors will be able to compute the Greeks for each product using a model receiving live updates from MangoCore, making use of the implied volatilities computed by inverting Black-Scholes.

2.6 Volatility Arbitrage

Volatility arbitrage aims to take advantage of inconsistent future and current volatilities. For a certain underlying P, profits can come from differences between the anticipated future realized volatility of P and current implied volatilities of options for P. Profits can also come from differences between current and future implied volatilities, and from the differences in implied volatilities across different strike prices.

For example, suppose a trader believes that the future realized volatility for a given product will be higher than the current implied volatility of a call option on that prod-

uct across all strike prices. Then the trader believes the option is cheaper than it should be, so (s)he buys it, hedges delta by shorting some amount of the underlying, and adjusts the hedge as the price of the underlying changes. Note that this trade has a positive vega, because the trader believes that the implied volatility will increase.

One of the most common types of volatility arbitrage is called volatility surface relative value trading. It involves attempting to earn profits by predicting changes in the shape of the volatility smile, independent of beliefs about vega.

Suppose you believe the volatility smile is relatively flat, compared to what you believe to be the true shape of the curve. Then you could buy options far out on the curve, perhaps hedge some vega exposure by shorting options with strike prices closer to the current spot price (at-the-money options) if you have no view on the future overall level of volatility, and then hedge your delta exposure by holding a position in the underlying equal and opposite to the remaining delta of your options portfolio. This leaves you exposed only to changes in the curvature of the smile. A completely analogous strategy could be employed if you believe that the skew of the volatility smile will increase.

One way to model the volatility smile of the options is to assume that the world exists in a discrete number of volatility states. Each volatility state is associated with its own underlying return distribution. One would then come up with volatility curves that corresponds to these different states. The volatility curves are then subjected to a weighted average based on how probable you believe each volatility state is. If one finds a way to predict the evolution of probabilities of the volatility states, one could predict the future shape of the volatility curve.

Successful competitors in this case will be able to come up with a model of the evolution of the volatility curve and trade based on the inconsistencies between the reality they observe and their model.

2.7 Market Making

Another possible way to make money in options is market making, because options spreads are often relatively large. However, this runs the risk of adverse selection. For example, if you are offering 100 call options, you should be willing to do so *given that someone is willing to trade against you*. Someone who has a better grasp of the market, or better information, could take advantage of an overzealous market maker offering tiny spreads.

Successful competitors in this case will be able to market make when spreads are high, without exposing themselves to too much additional risk, such as by hedging.

2.8 Case Information

The trade-able instruments are options and TMXFUT futures, where the futures are solely for hedging. Both are cash-settled.

There are 82 options in each round of the case: 41 puts and 41 calls, in range(80, 121). Options expire at the end of each round and their tickers are subsequently re-used. Five example tickers are listed below; other tickers follow the same naming conventions:

Table 2.2: Tradable Options

Strike Price	Put Option Ticker	Call Option Ticker
$90	T90P	T90C
$95	T95P	T95C
$100	T100P	T100C
$105	T105P	T105C
$110	T110P	T110C

The ticker for futures on the index is "TMXFUT".

To successfully trade this case, you should make a pricing model to show the current volatility curve, your current portfolio Greeks, and any refinements or derived metrics that help you trade faster and better. This will allow you to quickly pursue arbitrage opportunities and to hedge your positions. In addition to pursuing such arbitrage opportunities when spreads are low, you should also recognize potential market making opportunities at times when spreads are higher, where the potential gain may outweigh the adverse selection risk. Competitors should read MangoCore API to understand how to get live MangoCore updates programmatically and then build models off of them.

2.9 Case Specifications

There will be four 7.5-minute periods. Each period represents one month of trading. One team member will trade while the other monitors a pricing model and advises the trader. Team members will switch roles after each period. Trading activity by both partners during any given round is grounds for disqualification. Unlike the SEC, we have granular audit logs – please don't make us use them.

All trade-able instruments expire at the end of each round. Each round begins with the T@MIT index at 100. The continuously compounded risk-free interest rate is **0%**. No dividends will be paid out during case periods.

The contract multiplier for **options** on the T@MIT index is 1. Options have a trading fee of $0.005 per contract per transaction. There is a net limit of 5,000 options contracts.

The contract multiplier for **futures** on the T@MIT index is 1. Futures have a trading

fee of \$0.005 per contract per transaction. There is a gross trading limit of 2,500 futures contracts, and a net limit of 2,500 futures contracts. All future contracts are cash-settled upon expiry.

There will also be delta and vega **portfolio limits** of $\pm 1,000$ and $\pm 3,000$ respectively. (Note that vega is scaled down by 1e4 to be on the same order of magnitude.) One share of the underlying will have a delta of 1. Upon exceeding either limit, competitors will be fined proportionally to the square of the difference between their Greek and the limit, at a rate of $\$10^{-3}$ per squared excess per second.

When starting to work on the 3 provided 15-minute sample cases, you will likely find it helpful to play around with click trading to get a sense of the limits. They are indeed fairly strict, but should be predictable enough to be avoidable. When in doubt, you can always trade in smaller quantities.

All outstanding positions at the end of each period will be automatically closed out at their fair prices, and options will be automatically exercised if they are in-the-money. In each round, your rank will be calculated based on your P&L (including any and all fees) relative to other competitors; winners will be determined by the best average rank across the four rounds.

3 Algorithmic Sales and Trading

3.1 Overview

In this case, you will be simulating investment bankers who manage large quantities of stocks from institutional customers. In the real world, trades can be in the tens of millions of dollars and can even encompass entire corporations. Due to the large quantity involved, these orders are not placed on the public market and are instead fulfilled by traders at investment banks.

The sales component of sales and trading denotes deciding what to buy and sell and convincing potential buyers and sellers, while the trading component denotes the execution of the trade itself, as well as making a profit from that trade. In this case, you will be fulfilling both duties. Your main tasks will be to

- Use the orders to determine information about the future of the market, including price impact and adverse selection
- Profit from trading with large customers based on said market prediction as well as the public market
- Manage positions to minimize risk

3.2 Securities

There is one underlying for this case, TRDRS, and you will be trading two tickers, TRDRS.LIT and TRDRS.DARK, both of which contain one share of the underlying. The distinction is made because TRDRS.LIT is traded publicly and TRDRS.DARK is delegated for trading with large customers, which will be in a separate market.

There is a strictly enforced position limit of 5000 shares on the underlying TRDRS but notably not on the tickers TRDRS.LIT and TRDRS.DARK. Thus, it is fine to have large positions on either security as long as the sum of the positions is within 5000 in absolute value.

3.3 Price Impact

The price of the TRDRS.LIT is maintained by a market maker, who will adjust its price based on the trading behavior of the competitors. Specifically, the market maker will make a market around the price

$$P = P_0 - C * \text{position}$$

where P_0 is a price that reflects the market excluding the competitors, C is a positive constant, and the position refers to the current position of the market maker (who obtains this position by trading with the competitors). Fixing P_0, an influx of buy orders from competitors which the market maker fulfills will cause the position to decrease, thus driving up the price. Similarly, an influx of sell orders will decrease the price.

3.4 Trading with Customers

As aforementioned, trading with customers will be done through the ticker TRDRS.DARK. Customers will first announce through the news interface that they will be buying or selling some N shares of TRDRS.DARK. This order will be fulfilled 5 ticks after the announcement.

Between the announcement and the fulfillment of the customer order, the competitors will send orders to the TRDRS.DARK market reflecting the prices and quantities they would like to trade at. There is a minimum order size of 1000 shares, and competitors are unable to view each others' orders. In addition, competitors may only trade on the opposite side as the customer (i.e. they may not fulfill each others' offers on this market).

The customer will take the best offers of the competitors until N shares are bought/sold (the offers with the lowest prices if they are buying and the offers with the highest prices if they are selling). If there is a tie in price, the offers put in first will take priority.

3.5 Adverse Selection

The case will also feature price movement separate from competitor interaction with the market maker (affecting the P_0 parameter in the equation about price impact). This will be reflected in the orders of different customers: in particular, there are customers that are able to predict the future of the market and will offer to buy/sell accordingly.

Specifically, there are three customers, two with insider information on the market and one without (please ignore the legality of this). The customer without insider information will trade essentially randomly: P_0 will not change after the order. When the customers with insider information announces orders, however, this is a sign that P_0 will soon change. For one customer, P_0 will increase by $C *$ order size (where C is the same constant as in the price impact equation and order size is positive for bids and

negative for asks) shortly after the order is fulfilled; for the other, P_0 will increase by $0.5C *$ order size shortly after the order is fulfilled. It is your task to figure out which customer knows what information.

Other than those changes reflected by customer orders, there will be no changes to P_0 throughout each round, save for noise.

3.6 Position Close-Out and Scoring

Any nonzero position in the underlying TRDRS will be closed out at the end of the trading period with the closing price and a penalty of (25% per unit). Your P&L during the case will be displayed in US dollars—our grading script will make the necessary currency conversions in real time.

Within each of the 4 rounds, you will be ranked by your P&L. Your P&L will reset to 0 at the beginning of each of the rounds. To determine final rankings, we will consider your average rank across all three rounds.

3.7 Case Specifications

Starting Endowment Per Round	US$100,000
Rounds of Trading	4
Length of Round	7.5 min/round
Fees	$0.0001/contract
Fines	Trading with yourself: $0.008/contract
Constraints	A position limit of $[-5000, 5000]$ will be enforced on the underlying TRDRS; orders that would cause you to go outside your position limit will be rejected. The maximum transaction size is USD$1,000. You may only send 25 pieces of information a second, otherwise your orders will get rejected. More information is available in the Mangocore API section.

We will use $C = \frac{1}{25000}$ for this case: in other words, 25000 shares of TRDRS.LIT need to be bought or sold to change the price by $1. This number may be subject to change.

4 Five Rings Capital Day Of

4.1 Introduction

You'll have prepared your strategy and code for the other two cases before you even walk in to the competition, so, just so you something more to do than watch your code battle it out on our scoreboards, we thought we'd have a case for you to work on completely onsite. You'll get complete details the day of competition, but we thought it might be fair to give you some idea of what to expect. The Day Of Case is sponsored by **Five Rings Capital**.

4.1.1 Components

The goal of this case is to test two skills: **modeling**, and **executing** based on your model. Unlike our other cases, you won't be trading against other competitors.

For the modeling aspect of the case, you'll get signals corresponding to prices, as well as a set of features that might be relevant in explaining the movement of the signal over time. It might be useful for your model to also tell you something about the uncertainty of your estimate since real models tend to be quite noisy.

We won't test your model directly—instead, you'll want to use your model to write an allocation function, where at a given timestep, using the data currently available, you'll want to decide what to do with the money you have available to maximize your profit. Your allocation function will be scored based on its performance on a test dataset we'll give you—you'll be submitting your allocations as a text file over a set of test cases. There might be constraints around what you can do with your money and how you're scored, but it would be a shame to give away too much before the actual competition.

This portion of the contest will last three hours, and you'll be free to divide this time in whatever manner you find appropriate.

4.1.2 Technical Details

We'll provide template Python loaders to load currently accessible data as well as the data in raw format. We'll provide template files for the allocation task and a template

notebook to get you started visualizing and exploring data, but everything else will be left up to you. The template code we provide will be written in `Python3`, and we strongly recommend you use it as it will make your job easier. However, raw data and input/output specifications will be provided, so you will have the option of using the tools of your choice.

You'll definitely want to be familiar with doing statistics (especially common regressions) in Python. Being familiar with the `numpy` and `pandas` packages might be helpful since we might provide utility methods to load data either as a `numpy ndarray` and as a `pandas DataFrame`. As a minimum, you'll want to have a recent `Python3` and `numpy`, `scipy`, and `pandas` installed, to ensure our loading and testing code works on your system. In addition, you might also want to install `Jupyter`, as it's possible that we'll provide a template `Jupyter` notebook to get you started in your data exploration. We'll try to minimize the boilerplate code you'll have to write and get you right into the quantitative work.

5 MangoCore API

5.1 Overview

MangoCore interfaces with external clients with a JSON API over a websocket connection. Users can use this API to programmatically query our simulator servers which may be helpful for cases that are suited for higher frequency trading. This guide details how to set up a local server for testing and provides information about the TradersBot Python package (available on pip), which allows easy interfacing with Mangocore's JSON messages.

For all cases, competitors should expect to pull price paths in addition to other information relevant to each case from MangoCore and analyze this data with real-time models. Ideally information from these models should inform competitors' trades. The only case for which traders will be allowed to *submit* orders programmatically through MangoCore is the algorithmic trading case (FX), where competitors can both pull price paths and submit orders for various securities in real-time.

5.2 TradersBot Python Package

As MangoCore interfaces with external clients with a JSON API over a websocket connection, theoretically it is possible to interact with the client through any programming language of choice, provided the user knows how to set up a websocket connection. However, we *highly* recommend that competitors use the TradersBot Python package, which provides a Python wrapper to the MangoCore API that allows for easy websocket setup to interface with the Mangocore server. TradersBot's installation, usage and general documentation are extensively detailed at `http://mangocore-client.readthedocs.io`, which provides instructive examples along with a thorough description and purpose of each feature. This URL is subsequently referred to as the *Documentation*.

In this section, we will broadly describe the functions provided by TradersBot as well as the types of messages that the MangoCore server sends to TradersBot. We also provide details on how to run a local instance of MangoCore to allow for local testing.

5.3 Server setup

MangoCore is run on a server; during competition day, we will be hosting the MangoCore server and provide the necessary IP/URL for competitors to connect to. However, competitors might want to run MangoCore on a local server in order to test their strategies. To run the server locally, grab the executable for your operating system from the competitor Dropbox and follow the instructions in the instructional PDF in the Dropbox folder. The server will open up a port on localhost:10914 and you can connect to it by opening up a websocket connection to `ws://localhost:10914/#trader0`, although if you are using TradersBot, the websocket connection will be setup for you. *Note: you cannot use your given competition traderid locally because the provided binaries default to a small set of recognized traderids that will not include your assigned traderid*.

To run your algorithm on our server for competition day and practice sessions, switch the host from `localhost:10914` to an IP/port we will provide and `trader0` to your given trader ID.

5.4 TradersBot Setup

There are several parameters when connecting with a MangoCore server: the MangoCore server IP/URL (which will be provided to you on competition day for the competition server and can be run locally with information located in the section Server Setup), a team username, a password, and an optional token.

TradersBot will automatically establish a connection with the server when you instantiate it with necessary parameters. A thorough description of how to use TradersBot, including instantiation, is in the documentation.

5.5 Messages

MangoCore has several types of messages that it will deliver to TradersBot. We describe the types of messages below; the exact information they contain and the message format is described in more technical detail in the documentation. In general, you can expect that each message is delivered as some sort of Python dictionary with keys corresponding to relevant information for that type of message. For example, a trade message might contain a dictionary with a key “ticker" that denotes which stock ticker the trade is referring to.

For each type of message, you can implement a function in your Python script that takes in as input the `message`, and parses the information in the message, possibly analyzing it and/or updating any internal variables you have with the parsed information. TradersBot has an internal function on receiving each type of message that we can set to a user defined function. An example TradersBot use case is in the documentation, which implements functions for each type of message.

We describe the content and purpose of each message below. Again, for specific details about the message format and the dictionary that comes with each message, refer to the documentation. This is meant to outline and guide competitors on the purpose of each type of message.

5.6 Passive Information

5.6.1 onMarketUpdate

Sometimes, you will receive market updates for securities. This generally happens every 500ms as opposed to following every API call that may modify the state of the order books. However, you will receive every trade notification so you may be able to maintain an orderbook estimate that can be more accurate.

Additionally, you can ping the market with micro-orders to discover the top of the book. You will want to do this sparingly because of the trading limits (see the last section).

5.6.2 onTraderUpdate

As with market updates, you will generally receive trader updates every 500ms. However, you will be able to calculate your precise trader state at any moment in time based on other information you receive.

5.6.3 onTrade

You will receive a notification following every trade (not just the ones concerning your orders) immediately (after network/processing latencies of course). You can use this information to keep an accurate internal order book and also have more precise information about the state of the market.

5.6.4 onNews

For Algo S&T, you will receive news updates for announcements of customer orders that will be placed at a specified time in the future. These pieces of news will contain the price, quantity, direction, time and identity of the customer.

5.7 Submitting and Canceling Orders

For Algo S&T and Options, competitors will be able to submit and cancel orders through the Mangocore API.

For each message type, the user should have some specified function that takes in as parameters a `message` and also an object `order`, which initially starts off as an empty order. If the user wishes to buy a stock, they should use the method `addBuy` with several parameters, including the ticker of the stock, quantity, price and an optional unique token that identifies the order. If the user wishes to sell a stock, use the method `addSell` with the same parameters. If the user wishes to cancel an already existing order, then use the method `addCancel` with the unique token identifier of the order that they wish to cancel.

More detail about these functions is given in the documentation, along with a clarifying example.

5.8 Message Limits

There are limits for how frequently you may send messages: you may send a maximum of 25 pieces of information per second, where a piece of information is defined as a message, order, or cancel. For example, a message with 6 orders and 2 cancels counts as 1 + 6 + 2 = 9 pieces of information. Any piece of information past this limit of 25 gets discarded. Also to note, messages are either entirely processed or entirely discarded. In other words, if you have sent 20 pieces of information in the last second and you attempt to send 10 more pieces in your next message, nothing in your next message will be processed (but we will record that you have tried to send 30 pieces of information in this past second).

If you send over 250 pieces of information in any second, your connection will be immediately killed (this isn't because we don't like you; we just don't want you to overload our network and starve other connections).

We will be monitoring server load and if it seems that our limits are too strict, we will relax them.

5.9 More Information

You will run your algorithms using your own computers - we will not run them for you. We will provide AWS EC2 instance logins for all competitors, in order to ensure reliable network connection. Our server will also be running in the same AWS region.

If your client falls more than 5 seconds behind in receiving information, we will kill the connection.

5.10 Backtesting

A brief note about back-testing: profits that you see in your backtesting may not be the same as profits you may see during the actual competition. This may be due to

competition (other market makers decreasing your edge), increased latencies (your algorithm will not be running on the same machine as the exchange), and other factors. You may want to simulate these pessimistic conditions in your backtesting to make it more rigorous.