A guide to participating in CRUCIAL prediction markets

October 2nd 2025, Version 1.1

Contents

1. Introduction

CRUCIAL is an initiative to use prediction markets, with expert participants, to elicit and aggregate the diverse expertise and knowledge of the participants and produce unified probability forecasts of climate-related risks. These forecasts can evolve as new information becomes available to participants.

Prediction markets are similar to financial markets, or recreational gambling, but they are not intended to transfer the ownership of assets of risks, or to provide entertainment, but are specifically designed to synthesize and summarize information from many sources.

Participants in CRUCIAL markets do not have to pay to take part or stake money. Instead, they are provided with credits with which to trade contracts that pay out more credits if an outcome associated with the contract occurs. The credits that participants accumulate can, however, be converted into cash rewards using funds provided by market sponsors. Through this mechanism it is hoped that prediction markets can become an alternative way of funding climate forecasting research in a performance-driven way.

This document is intended to provide guidance on how to trade in CRUCIAL's markets. If you are participating, then it is assumed you have relevant domain expertise in forecasting climate-related risks. The aim of this guide is to help you translate your own knowledge and expertise into effective trading strategies. By doing this you will also contribute to improving the collective predictions generated by the markets.

CRUCIAL uses a prediction market platform called AGORA that was specifically designed for climate-related applications.

2. How CRUCIAL markets are structured

2.1 Outcomes

Each CRUCIAL prediction market on the AGORA platform consists of a set of possible outcomes that are *mutually exclusive and comprehensively exhaustive* — that just means the outcomes are defined so that only one of them can occur but one of them *must* occur. For example, in a market to predict how many Atlantic hurricanes will occur in a particular year there might be 21 outcomes: no hurricanes, one hurricane, and so on up to "20 or more hurricanes". Only one of these outcomes can occur and the inclusion of the open-ended final interval means that all eventualities are included.

Each outcome is assigned a price by the platform's "market maker" based on how many of that outcome the market maker has already sold (more about this later). The prices across all the outcomes in a market will always add up to one. Participants can buy one or more units of an outcome. When the actual outcome becomes known the market is settled and all units of the correct outcome become worth one credit while all the other outcomes become worthless.

If the prevailing price of an outcome matches its expected value, then this price can be interpreted as the probability that this outcome will occur — according to the “market consensus”.

2.2 Contracts

On the AGORA platform you don’t trade individual outcomes directly. Instead, you can create *contracts* that consist of one or more outcomes. This feature is because some markets can have large numbers of outcomes (thousands in some cases) and having to trade each one individually would be inconvenient. For example, the tropical Pacific sea surface temperature anomaly might be partitioned into outcomes with each covering 0.1°C (with open-ended intervals to include any extremes). The contract feature allows you to create a single contract containing all outcomes in a range of your choosing: you could create a contract covering all outcomes larger than +2°C which you can trade as a single unit. The contracts you create are for your convenience, other participants cannot see them or trade them; they are just a handy way of trading multiple outcomes with the market maker.

Once you have created a contract the market maker will give it a price which is simply the sum of the prices of the outcomes it covers. This price is the “instantaneous” or “marginal” price, essentially the per unit price for buying an infinitesimally small number of the contract. If you place an order for a finite number of the contract the price per contract will increase, possibly quite a lot if it’s a large order. This will be explained further when we discuss how the market maker works.

2.3 Buying and selling contracts

Once you have bought a contract you can hold it until expiration — when the actual outcome is known, and the market is settled — or you can sell it back to the market maker at any time. To sell a contract you must have previously bought it. AGORA does not allow “shorting” (the selling of contracts that you don’t own). In financial markets shorting is a way to take advantage of overpricing (you borrow the asset and sell it in the hope you can buy it back at a lower price before you must return it to whoever loaned it to you). Because AGORA markets have all possible outcomes, and prices across these outcomes always sum to one, shorting is unnecessary: If you believe any outcomes are overpriced there must be other outcomes that are underpriced, so you can take advantage of the overpricing by buying the underpriced outcomes.

2.4 Joint-outcome markets

AGORA can support joint-outcome markets in which each outcome corresponds to values for two separate variables: e.g., temperature and rainfall, or carbon dioxide concentration and global temperature anomaly. The outcome space for these kinds of markets is a two-dimensional grid. The prices in a joint-outcome market provide an implied joint-probability distribution. Although they will typically have many more outcomes than one-dimensional markets there is nothing fundamentally different about joint-outcome markets. The outcomes are still defined to be mutually exclusive and comprehensively exhaustive and have prices that sum to one.

3. The automated market maker

3.1 Why use an automated market maker?

CRUCIAL markets use an automated market maker (AMM) that is always willing to quote prices at which it will sell or buy back contracts. This contrasts with other prediction markets (and financial markets) that use a *continuous double auction* (CDA) that matches buyers and sellers directly. In a market with thousands of possible outcomes it might be very unlikely that two participants will want to trade the same collection of outcomes, making direct matching impossible. The AMM overcomes this and allows markets with large numbers of outcomes to function properly. Another feature of the AGORA AMM is that it is *subsidized*, it is willing to lose money in return for good information. This feature allows markets with small numbers of highly informed participants (and no "dumb money") to work. The AMM essentially takes the role that uninformed participants play in other prediction markets.

3.2 How the market maker works

To trade you don't need to know the details of how the AMM is setting its prices, you just need to decide whether the price it is offering is above or below what you consider the "fair value" of the contract, which is the probability that one of the outcomes it includes will occur. However, the kind of people who participate in CRUCIAL's markets tend to be curious and knowledge of how the AMM works can help when it comes to constructing more sophisticated trading strategies, so this section describes the AMM that CRUCIAL's AGORA platform uses.

AGORA uses an AMM based on the logarithmic market scoring rule (LMSR). Let there be m outcomes (mutually exclusive and comprehensively exhaustive). Let q_i be the exposure of the AMM to the ith outcome ($i = 1,2,\cdots,m$), i.e., the number of that outcome it has sold to participants minus any that it has bought back. The AMM uses a cost function given by

$$C(\boldsymbol{q}) = b\log\left(\sum_{i=1}^{m} e^{q_i/b}\right)$$

where $\boldsymbol{q} = (q_1, q_2, \cdots, q_m)$ and b is the *liquidity parameter*. The instantaneous, or marginal, price of outcome j is given by the derivative of this cost function w.r.t. to q_j.

$$p_j = \frac{\partial C(\boldsymbol{q})}{\partial q_j} = \frac{e^{q_j/b}}{\sum_{i=1}^{m} e^{q_i/b}}$$

from which we see that prices of all outcomes sum to one, $\sum_{i=1}^{m} p_i = 1$. We can also see that if one outcome becomes very popular relative to the others then its price will approach 1.0.

If we specify a contract as $\boldsymbol{w} = (w_1, w_2, \cdots, w_m)$ the price that the AMM will ask for this contract is given by

$$C(\boldsymbol{q} + \boldsymbol{w}) - C(\boldsymbol{q}) = b \log\left(\sum_{i=1}^{m} e^{(q_i + w_i)/b}\right) - b \log\left(\sum_{i=1}^{m} e^{q_i/b}\right)$$

The parameter b controls how much the prices quoted by the AMM move for a given trade size. The larger the value of this parameter the less impact on prices a given trade will have.

3.3 Relationship with the logarithmic scoring rule

An elegant feature of the LMSR AMM (and where it gets its name) is that it rewards participants according to the logarithmic scoring rule for scoring probability forecasts. To see this consider the situation where there is a single participant who believes the true probabilities for each outcome are f_i $(i = 1, 2, \cdots, m)$ and they buy outcomes until the prices quoted by the AMM match these probabilities. The exposure of the AMM that corresponds to these prices is given by

$$q_j = b \log f_j + b \log\left(\sum_{i=1}^{m} e^{q_i/b}\right) = b \log f_j + bA$$

where $A = \log\left(\sum_{i=1}^{m} e^{q_i/b}\right)$.

Buying this exposure will cost the participant

$$C_f = b \log\left(e^A \sum_{i=1}^{m} f_i\right) - b \log m = bA - b \log m$$

If outcome k is the one that ultimately occurs, then the participant will receive q_k, so their net reward will be

$$q_k - C_f = b \log f_k + b \log m$$

which is linear in $\log f_k$, which is the logarithmic scoring rule.

The largest net reward the participant can receive is bounded by the case $f_k = 1$, and is given by $b \log m$. As mentioned, the AMM is subsidized and prepared to lose money, but its potential loss is controlled by the parameter b.

4. Trading strategies

4.1 Trading is driven by differences in opinion

The prices, p_i, for outcomes can be interpreted as the probabilities that they will occur according to the market consensus. If your own estimate of these probabilities, f_i, differs from the market consensus then there is an opportunity for you to trade with the expectation of making a profit.

Given the marginal prices, p_i, the AMM's exposure can be inferred using

$$q_i = b \log p_i + K$$

where K is some constant. The exposures can only be inferred up to some constant because uniform changes in exposure across all outcomes do not affect prices. If you buy one unit of every outcome it will always cost you one credit, and you are guaranteed to get that credit back at settlement.

Prices will move to match your own view of the probabilities if you make the following trade

$$w_i = b \log f_i - b \log p_i + K_w$$

Assuming you have no outcomes to sell you can make $w_i \geq 0$ by setting

$$K_w = -\min_i (b \log f_i - b \log p_i)$$

The cost of buying this position is given by

$$C_{p \to f} = b \log \left(\sum_{i=1}^{m} e^{(b \log p_i + K + b \log f_i - b \log p_i + K_w)/_b} \right) - b \log \left(\sum_{i=1}^{m} e^{(b \log p_i + K)/_b} \right) = K_w$$

So, your expected profit will be

$$\text{Expected profit} = \sum_{i=1}^{m} f_i w_i - K_w = b \left(\sum_{i=1}^{m} f_i \log f_i - \sum_{i=1}^{m} f_i \log p_i \right)$$

which is the liquidity parameter multiplied by the relative entropy between your probability distribution and the market distribution, which is kind of cool.

WHAT IF YOU ARE CASH CONSTRAINED?

i.e. MAXIMIZE EXPECTED PROFIT FOR A GIVEN OUTLAY

4.2 Arbitrage

Arbitrage is a term used in trading that refers to taking advantage of price discrepancies. The word is sometimes reserved for risk-free trades but is also frequently used for trades that aren't completely without risk: e.g., buying a commodity in one place more cheaply than you can sell it in a different place — there is a risk that the logistics of transporting the commodity between these places might scupper the profitability of this trade. We will use the term arbitrage in this more general way to mean trades that exploit apparent inconsistencies in prices. For example, suppose in a market for the number of hurricanes an outcome is priced lower than the adjacent outcomes on either side: e.g., the price of 7 hurricanes is materially lower than both 6 or 8 hurricanes. Whatever your view on the likely number of hurricanes this bimodality seems unphysical and implies 7 is underpriced relative to its neighbours.

Another potential opportunity for arbitrage is when a strip of markets is predicting a quantity at consecutive horizons — e.g. the monthly tropical Pacific sea surface temperature anomaly (SSTA). While tropical Pacific SSTA does change from month to month its probability distribution at longer teams, at the limits of predictability, shouldn't change too much from one month to the next, and large differences between the price distributions for consecutive months.

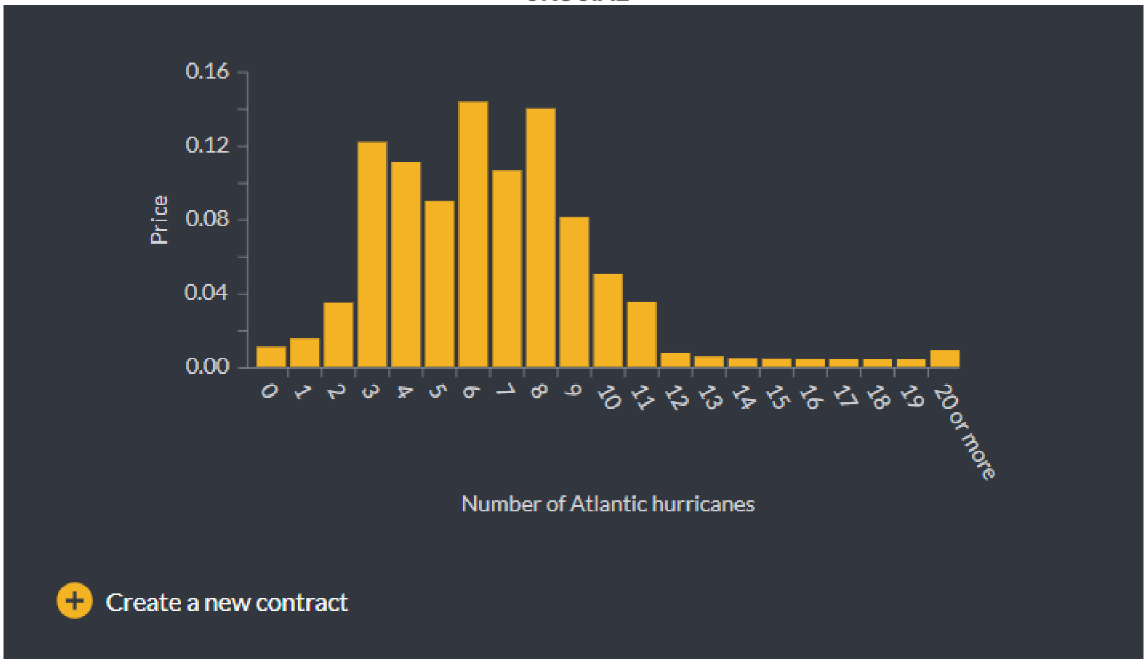
It is possible that two CRUCIAL markets might predict *exactly* the same thing. This might seem odd but with joint-outcome markets it might be that two markets share one axis: e.g. a market to predict CO_2 concentration and global temperature anomaly and a market to predict CO_2 concentration and global sea-level rise. In this case it may be possible to engage in genuine risk-free arbitrage. In a joint-outcome market you can bet purely on one of the variables by creating a contract that covers a limited range of outcomes in that variable but all possible outcomes in the other variable. If the marginal distribution of prices for the same variable differs between markets, then there is an arbitrage opportunity that can be exploited by buying every outcome for that variable but buying each outcome in the market where it is cheapest. This will allow you to buy a complete set of outcomes — which is guaranteed to pay out 1.00 credit — for less than 1.00 credit.

5. Trading using the AGORA user interface

5.1 Outcomes and prices

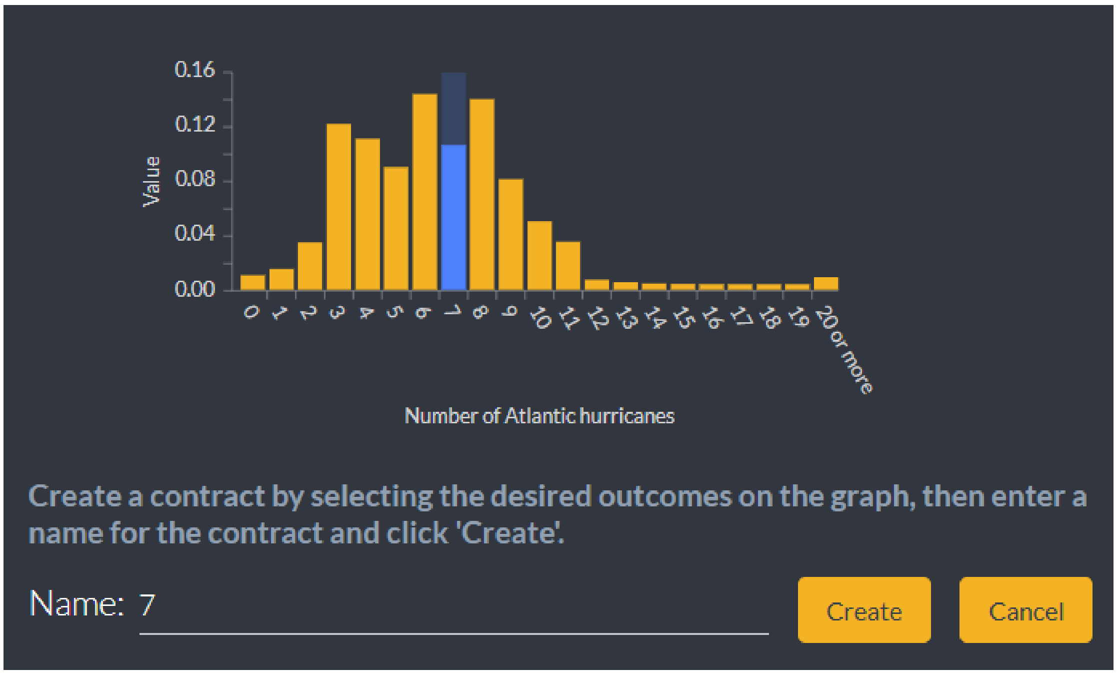
When a market is selected in the AGORA user interface (UI) the current prices of each outcome are displayed. These are the instantaneous, or marginal prices, the prices for an infinitesimally small amount.

CRUCIAL


Price
0.16
0.12
0.08
0.04
0.00
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 or more
Number of Atlantic hurricanes
Create a new contract

5.2 Creating and trading contracts

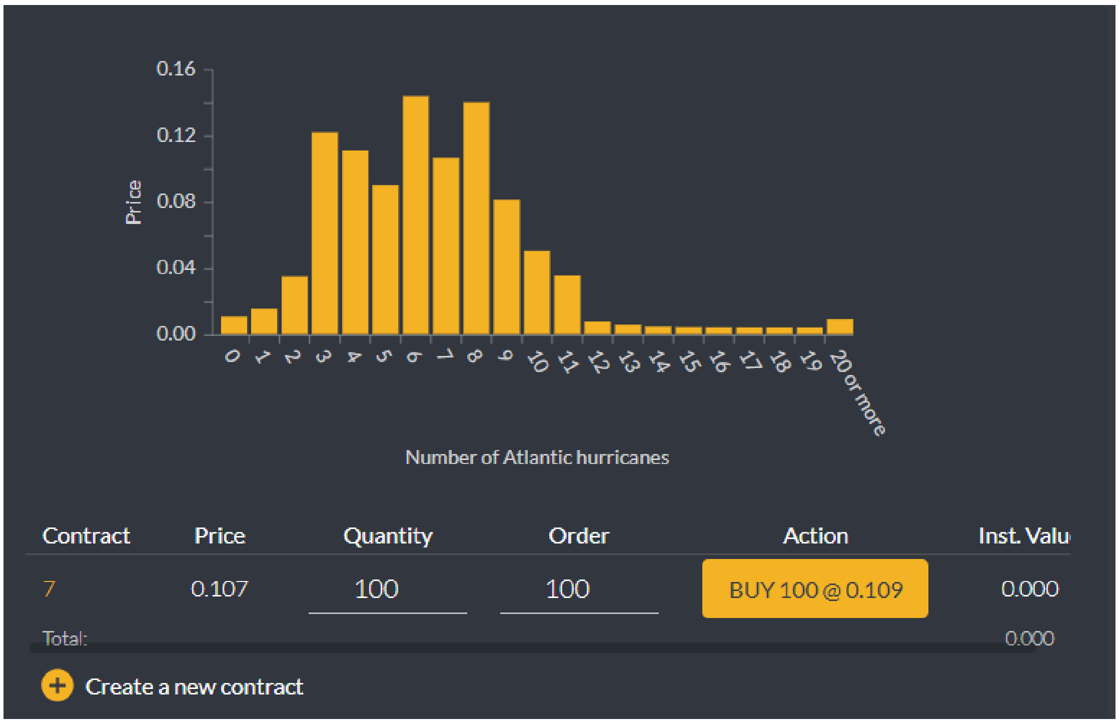
Outcomes cannot be traded directly, instead contracts are traded which can cover one or more outcomes. To create a contract select “create a new contract” and then select the outcome(s) you want to be included in this contract. AGORA will give the contract a default name, which you can change. For example, suppose we want to create a contract for 7 hurricanes.


Value
0.16
0.12
0.08
0.04
0.00
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 or more
Number of Atlantic hurricanes
Create
Cancel

Create a contract by selecting the desired outcomes on the graph, then enter a name for the contract and click 'Create'.

Name: 7

Once the contract has been created it appears in our list of contracts and we can now trade it by entering an order. Suppose we want to buy 100 units of the contract, we enter +100 in the order column (or alternatively the quantity column can be modified).



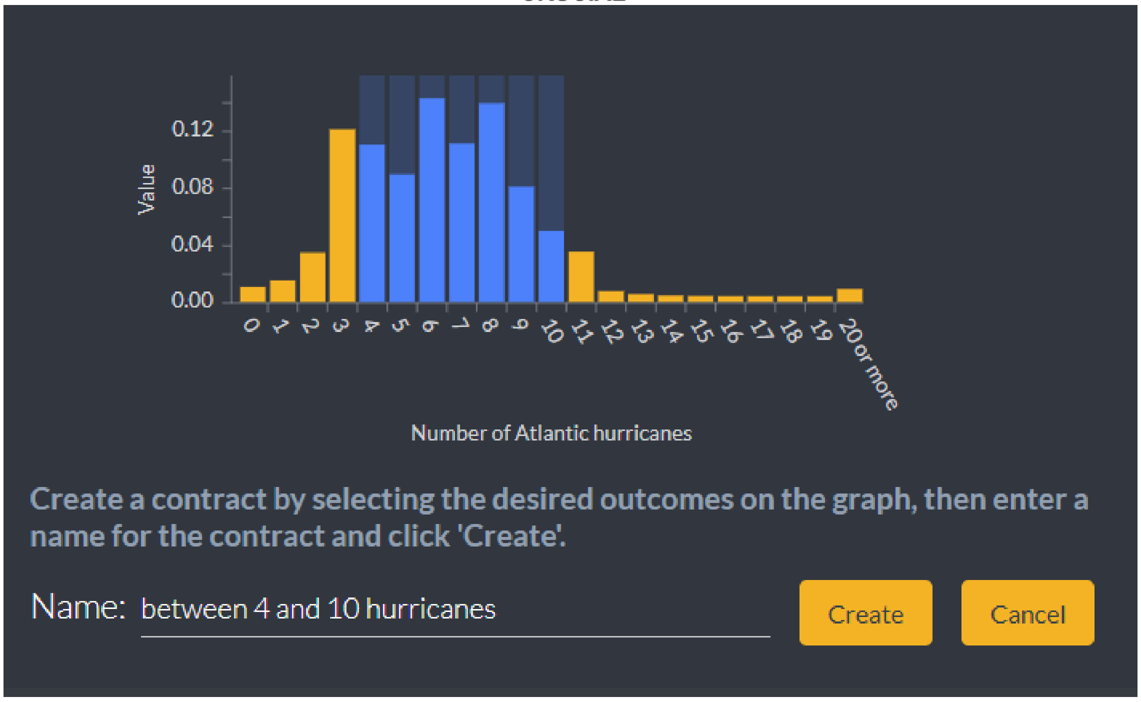
Price
0.16
0.12
0.08
0.04
0.00
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 or more
Number of Atlantic hurricanes
Create a new contract

Contract	Price	Quantity	Order	Action	Inst. Valu
7	0.107	100	100	BUY 100 @ 0.109	0.000
Total:					0.000

The AMM quotes the price for these 100 contracts. Notice that, while the instantaneous price for this contract is 0.107, the AMM quotes a price of 0.109 per contract for 100 of them. The larger the order the greater the difference will be between the instantaneous price and the quoted price per contract. If the quoted price is acceptable, we can click buy and the contracts will be added to our inventory.

5.3 Multiple outcome contracts and baskets

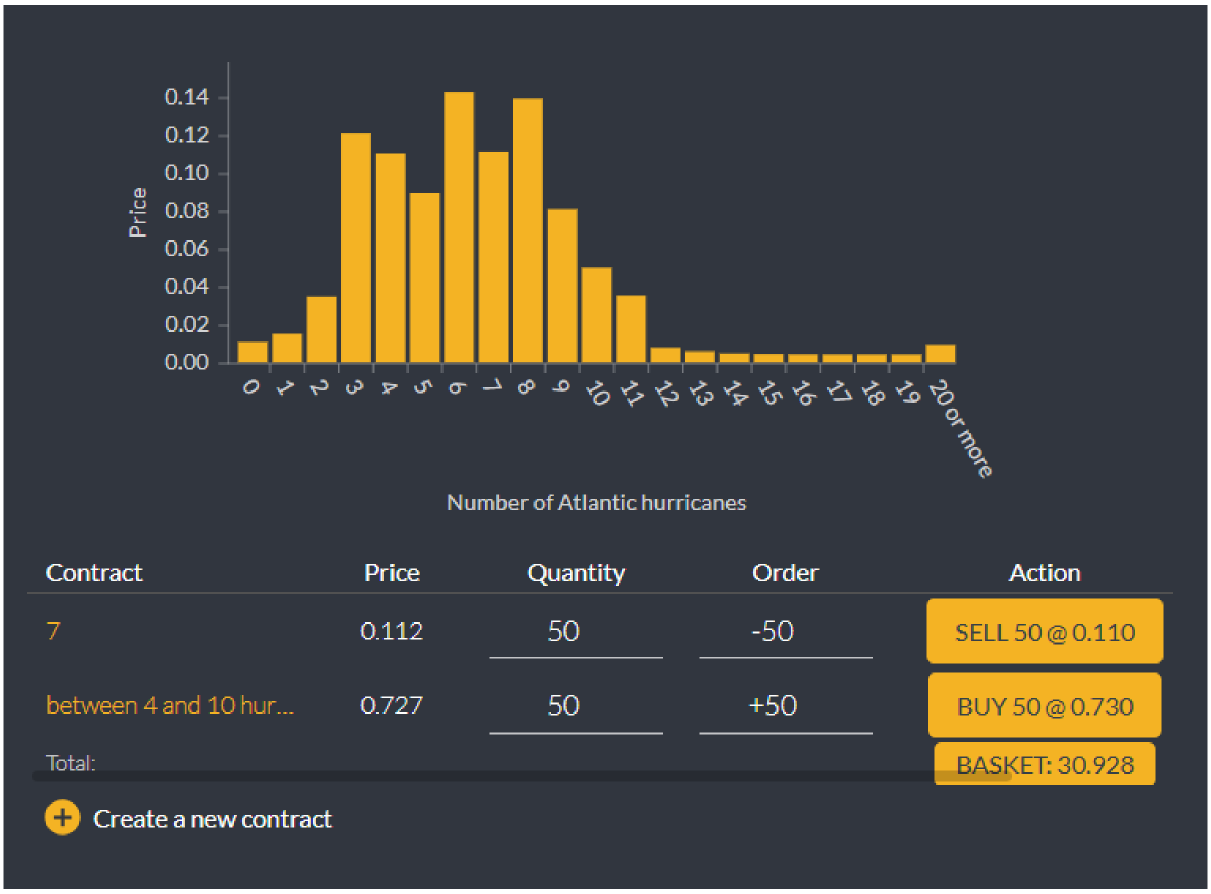
Contracts can contain multiple outcomes. For example, we can define a contract covering a range of outcomes. Notice that we have changed the name of this contract to "between 4 and 10 hurricanes" rather than use the default name provided by AGORA. The outcomes included in a contract don't have to be contiguous.



Create a contract by selecting the desired outcomes on the graph, then enter a name for the contract and click 'Create'.

Name: between 4 and 10 hurricanes

To sell contracts that we have already bought we just need to enter a negative order in the order column. The AMM will quote us the price at which it is willing to buy them back. We can also enter orders for different contracts, including a mix of buy and sells as below, which the AMM will price together as a "basket". This allows us to execute all the trades together.



Contract	Price	Quantity	Order	Action
7	0.112	50	-50	SELL 50 @ 0.110
between 4 and 10 hur...	0.727	50	+50	BUY 50 @ 0.730
Total:				BASKET: 30.928

CRUCIAL

6. Trading using the AGORA API

6.1 Introducing the API

The AGORA API (Application Programming Interface) allows market data to be extracted from AGORA and trades to be placed programmatically, without having to use the user interface.

6.2 agora-client

The library `agora-client.exe` allows access to the AGORA API. It can be downloaded from the CRUCIAL website.

Once you've installed it, agora-client can be used to communicate with the AGORA API.

6.3 Biscuit

The AGORA API uses biscuit authorization tokens. When you are logged on to AGORA go to the 'Profile' tab where you can download a biscuit file. You can then get a public key using the agora-client pubkey command from the console:

```
>agora-client --biscuit-file [biscuit_filename] pubkey new
```

Paste the public key onto the AGORA 'Profile' page.

Try the following command to test your access to the API, it should return your user ID and email address as a json object:

```
>agora-client --biscuit-file [biscuit_filename] get me
```

On success the client writes the response as a json array to stdout. On failure, the client returns an error code and writes nothing to stdout. The client writes log and error messages to stderr.

Machine readable (json) documentation is available using:

```
>agora-client --biscuit-file [biscuit_filename] get /
```

6.4 Using agora-client from R

Calls to the AGORA API can be made from within R using the `shell`[1] command (other programming languages have similar functions). For example:

```
>cmd = sprintf("agora-client --biscuit-file %s get market", biscuit_file)

>x = shell(cmd, intern=TRUE, ignore.stderr=TRUE)
>markets = fromJSON(x)
```

[1] Only the Windows version of R uses `shell()`. For other versions you can use the `system()` function.

Where `biscuit_file` is the name of the biscuit file downloaded from AGORA. These commands download a list of markets as a json object. The json object can be parsed into an R data structure using the jsonlite library.

```
>require(jsonlite)

>markets

  market_id                                              market organization_id pool_id liquidity_factor

1       106 DEMO: Atmospheric CO2 concentration (ppm) in 2050               4      20          1000.00

2       107     DEMO: Number of Atlantic hurricanes this year               4      20          2022.89

3       111                          OPER: RONI-001-2025-SON              91     141           309.00

4       112                          OPER: RONI-002-2026-DJF              91     141           309.00

5       113                          OPER: RONI-003-2026-MAM              91     141           309.00

6       114                          OPER: RONI-004-2026-JJA              91     141           309.00

7       115                          OPER: RONI-005-2026-SON              91     141           309.00

8       116                          OPER: RONI-006-2027-DJF              91     141           309.00

9       117            OPER: CYCLONES-ATLANTIC-HURRICANES-2025              91     141           826.00
```

A list of contracts for a particular market (given by `market_id`) will be returned by the following commands:

```
>market_id = 107

>cmd = sprintf("agora-client --biscuit-file %s get contract --market-id
%i",biscuit_file,market_id)

>x = shell(cmd, intern=TRUE, ignore.stderr=TRUE)

>contracts = fromJSON(x)

>contracts

contract_id market_id            contract             outcomes

1         503       107 2, 3, 4, 5, 6, 7, 8 2, 3, 4, 5, 6, 7, 8

2        1115       107                   4                    4

3        1116       107                   5                    5
```

These are the contracts that you have defined and are specific to you.

Contracts can be created using the API by defining the contracts in a JSON object and piping it to the `post contract` command.

```
>contractJSON = toJSON( list(contract=unbox("Name of Contract"),
market_id=unbox(107), outcomes=c(323,324)))

>cmd = sprintf('echo %s | agora-client --biscuit-file %s post contract',
contractJSON, biscuit_file)

x = shell(cmd, intern=TRUE, ignore.stderr=TRUE)
```

Note the use of the `unbox()` function. In the JSON object the outcomes must be boxed, even if there is only a single outcome, so instead of using `auto_unbox=TRUE` the `contract` and `market_id` must be explicitly unboxed.

CRUCIAL

To make a trade via the API we create a JSON object.

```
>tradeInstruction = toJSON( list(contract_id=503, contracts=+1000, market_id=107), auto_unbox=TRUE)
```

We then pipe this to the `post execution` command.

```
>cmd = sprintf('echo %s | agora-client --biscuit-file %s post execution', tradeInstruction, biscuit_file)

>x = shell(cmd,intern=TRUE, ignore.stderr=TRUE)

>tradeResult = fromJSON(x)
```

The `tradeResult` object will contain details of the trade that was actually executed. If you did not have enough credits for the full trade it will be partially executed and the actual number of contracts traded will be given.

Basket trades can also be sent with a JSON object array.

```
>order1 = list(contract_id=1115,contracts=+1,market_id=107)

>order2 = list(contract_id=1116,contracts=+3,market_id=107)

>tradeInstruction = toJSON(list(order1,order2),auto_unbox=TRUE)
```

Which can be passed to the API as before.

You can get your current portfolio using `get portfolio` and your current balance with `get balance`.

```
>cmd = sprintf('agora-client --biscuit-file %s get portfolio', biscuit_file)

>x = shell(cmd,intern=TRUE, ignore.stderr=TRUE)

>portfolio = fromJSON(x)
```

A suite of helper and wrapper functions for using the API from R is available for download from the CRUCIAL website.

The following code creates a contract corresponding to each outcome in market 107.

```
prices = get_prices(biscuit_file, 107)

nOutcomes = nrow(prices)

contracts = vector(mode='list', length=nOutcomes)

for (iOutcome in c(1:nOutcomes)) {

      contract = list(contract=unbox(sprintf("contract %s (API created)", prices$values[iOutcome])),market_id=unbox(107), outcomes=prices$outcome_id[iOutcome])

      contracts[[iOutcome]] = contract

}

x = create_contract(biscuit_file, contracts)
```

CRUCIAL