LYRA CAPITAL

User guide

What each screen does, in which order to use them, and what the numbers mean.

How to read it. Each chapter is one area of the application. Read it from the start, or go straight to the chapter for the screen in front of you.

To change Lyra, rather than use it, there is another document: the *Guida dello sviluppatore* (developer guide, in Italian), in the same folder.

Document generated from LaTeX source
21st November 2026

Before you start

Lyra is for building investment strategies, testing them on past data and — when they hold up — running them on a real account. The three happen in that order, and this guide follows the same order.

In the text, **This** is a menu item or a tab, This a button to press, *this* a field to fill in, and Settings → Data means: open **Settings**, then the **Data** tab.

The boxes

Note

A clarification. Skip it and the operation still works.

In practice

The operation, step by step.

Warning

Something can be lost here: data, money or time. Worth reading first.

A word about the numbers

No result shown by the application is a forecast. A backtest says how a strategy *would have* behaved on past data, with estimated costs; it does not say how it will behave. The chapter on **Streams** explains which numbers to look at to tell a strategy that worked from one that got lucky.

Contents

1 Getting started . . . 1
1.1 What Lyra does . . . 1
1.2 Opening Lyra . . . 1
1.3 The menu . . . 2
1.4 Do you need an account? . . . 3
1.5 The order in which you work . . . 3

2 Overview . . . 4
2.1 The portfolio . . . 4
2.1.1 Which project is shown . . . 4
2.1.2 The value . . . 5
2.1.3 The chart . . . 5
2.2 Key metrics . . . 6
2.2.1 Choosing the metrics . . . 7
2.3 Allocation, years and next steps . . . 7
2.4 Streams . . . 8
2.4.1 Sorting, searching, filtering . . . 9
2.5 On a smaller screen . . . 9

3 Studio . . . 10
3.1 The five families . . . 10
3.2 What is in the system . . . 11
3.3 The three ways to build . . . 11
3.3.1 Blueprint: with blocks . . . 11
3.3.2 Code: by writing . . . 12
3.3.3 AI agent: in words . . . 13
3.4 Metrics . . . 14
3.5 Genetics . . . 15
3.5.1 Preparing an evolution . . . 16
3.5.2 Following the run . . . 17
3.5.3 Reading the result . . . 17
3.5.4 The evolved strategies . . . 17
3.6 What happens when you save . . . 20
3.7 Bringing in an artifact from outside . . . 20

4 Streams . . . 23
4.1 Running a backtest . . . 23
4.1.1 Running on the live bars . . . 24
4.1.2 Hiding a strategy, or deleting it . . . 26
4.1.3 Obsolete strategies . . . 27
4.1.4 An empty stream: a strategy in a project before its backtest . . . 27
4.2 The page of a result . . . 29
4.2.1 How the positions were sized . . . 29
4.2.2 Robustness: the panel to read first . . . 31
4.2.3 The metrics . . . 32

4.2.4 The charts . . . 32
4.2.5 Choosing the metrics . . . 33
4.2.6 Comparing a live with its backtest . . . 34
4.2.7 Arranging charts and tables . . . 35
4.2.8 Exporting to Excel or CSV . . . 36
4.2.9 The three buttons at the top . . . 37
4.3 View: seeing what it did . . . 37
4.3.1 The page . . . 38
4.3.2 What the strategy drew . . . 39
4.3.3 The trades on the chart . . . 39
4.3.4 The colours of the candles . . . 39

5 Projects . . . 41
5.1 The builder, in four steps . . . 41
5.1.1 Step 1 — Strategies . . . 41
5.1.2 Step 2 — Weighting . . . 42
5.1.3 Step 3 — Overlay . . . 43
5.1.4 Step 4 — Costs . . . 44
5.2 The results . . . 44
5.3 The weights are decided with the past only . . . 45
5.4 When strategies overlap . . . 46
5.5 Saving . . . 46
5.6 Projects with empty streams . . . 46

6 Live . . . 48
6.1 What a runner is . . . 48
6.2 The three brokers . . . 50
6.3 Where the prices come from . . . 50
6.3.1 Each strategy runs as in its backtest . . . 51
6.4 Simulation or real orders . . . 51
6.4.1 The confirmation is for one account . . . 53
6.5 The journal . . . 54
6.6 The kill-switch . . . 54
6.7 When orders do not go out . . . 56
6.8 The simulator . . . 58
6.9 Credentials . . . 58
6.9.1 Symbol names on MetaTrader 5 . . . 58
6.10 Starting and stopping . . . 59

7 Community . . . 61
7.1 Why an account is needed here and not elsewhere . . . 61
7.2 Browsing the storefront . . . 62
7.3 The artifact page . . . 63
7.4 Buying . . . 64
7.4.1 The data an artifact needs . . . 64
7.4.2 What bought code can do on your computer . . . 65
7.4.3 If a purchase does not show up . . . 66
7.5 Selling . . . 66
7.6 If an artifact does not perform as declared . . . 67

8 Settings and data . . . 69
8.1 Data: bringing in the prices . . . 69
8.1.1 Downloading . . . 69
8.1.2 Uploading your own file . . . 70
8.1.3 The categories . . . 70
8.1.4 Keeping a category up to date . . . 70
8.1.5 The data of the live runners . . . 72
8.2 Config: the limits . . . 75
8.3 Market rules: the regimes . . . 77
8.3.1 How to read the page . . . 77
8.3.2 The chart . . . 78

9 Account and subscription . . . 79
9.1 What the account is for . . . 79
9.2 Signing in for the first time . . . 79
9.3 The profile . . . 80
9.3.1 Automatic publishing . . . 81
9.4 Speed: how pages open, hardware acceleration . . . 82
9.5 The plans . . . 83
9.5.1 The two-month trial . . . 84
9.5.2 Desk . . . 84
9.5.3 Monthly or yearly . . . 85
9.5.4 Changing plan . . . 85
9.5.5 Cancelling, card, invoices . . . 85
9.6 When something does not add up . . . 85

1

Getting started

The numbers in the screenshots are made up

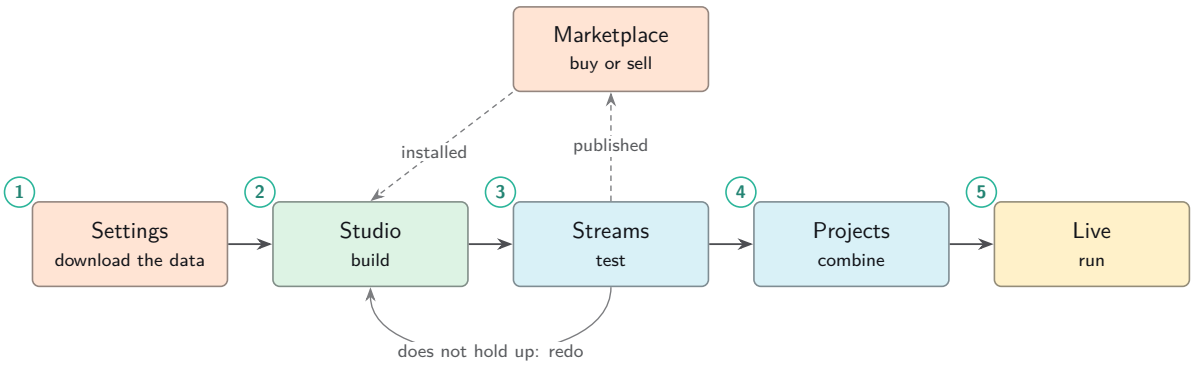
Every image in this guide was taken on an installation with *sample data*: series generated by a computer, not real prices and not the results of real strategies. They are there to show where things are. No Sharpe ratio, return or drawdown on these pages means anything.

1.1 What Lyra does

Three things, in this order:

1. **Build** an investment strategy — by writing it, by drawing it with blocks, or by describing it in words.
2. **Test** it on past data, with costs, and see how it would have behaved.
3. **Run** it on a real account, when it holds up.

Strategies can also be combined with one another, and bought and sold between users on the marketplace of the **Community**. This guide follows the same order: one chapter per area of the application.



Only the Marketplace, in the Community, needs an account. Everything else runs on your own computer.

Figure 1.1. The areas of the application, in the order you use them. The arrow that goes back is the most frequent part of the work.

1.2 Opening Lyra

Lyra opens in its own window. Behind the window there is a small server on your computer, and only the window can talk to it: at every start Lyra makes a new key, gives it to the window and refuses everything else — a web page you are visiting, another program, someone on your

network. If port 5000 is taken (on a Mac it usually is, by AirPlay Receiver), Lyra picks another one by itself.

> **Starting Lyra from a terminal or Docker**
>
> `avvia.sh`, `avvia.bat` and Docker print the address to open, with the key in it: `http://localhost:5000/__lyra/entra?t=...`. Open that one, not `http://localhost:5000` : without the key the page says 'Open Lyra from its window'. The key changes at every start.

1.3 The menu

The bar on the left is always there. It widens when you hover over it with the mouse and shows the names of the items.

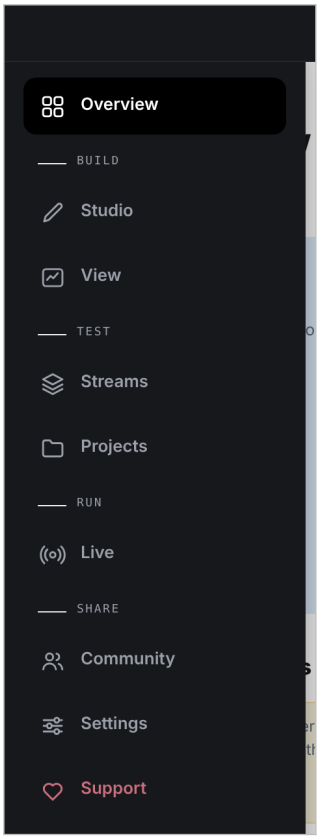


Figure 1.2. The menu, widened. Below **Overview** the items are in four groups — **Build**, **Test**, **Run**, **Share**. **Settings** sits at the bottom.

Item	Group	What it is for
Overview	—	The home screen: how everything is going, on one page.
Studio	Build	Where strategies are built.
View	Build	The charts, with what the strategy did drawn on top.
Streams	Test	The results: one card per strategy.
Projects	Test	Combinations of several strategies.
Live	Run	The link to a real account.
Community	Share	The strategy marketplace, messages, feedback and your own account.
Settings	—	Data, parameters, market regimes.

Top right

The magnifying glass searches the whole application; the bell shows finished jobs and messages; the sun switches between light and dark; the last item is the way into your account.

Back and forward work as in a browser: `Alt+←` and `Alt+→` on Windows and Linux, `Cmd+[` and `Cmd+]` on macOS, or the side buttons of the mouse. Back from an artifact's page, for example, returns to the marketplace as you left it (Section 7.2).

1.4 Do you need an account?

To build, test and run strategies, **no**: the application runs on your own computer and asks for nothing.

An account is needed only for the areas that go through the network: the **Community** — marketplace, messages, feedback, profile — and the subscription. If you open one of them without being signed in, the application says so and offers to sign you in.

1.5 The order in which you work

From nothing to the first result

1. Settings → Data: download the data. Without data, every page is empty — it is not an error, there is simply nothing to show.
2. Streams → Run backtest: choose a strategy and press **Run backtest**.
3. Wait. The job runs in the background and the bell tells you when it is done; meanwhile you can use the rest of the application.
4. **Streams**: the result has appeared as a card.
5. **Projects**: combine two or more results.
6. **Live**: connect a broker, when you are ready.

2

Overview

It is the page that opens first, and it answers three questions: how the portfolio is doing, what it is made of, and which strategies are earning their place in it. Everything sits directly on the page, in four areas separated by thin lines the way a trading platform is: the portfolio at the top left, its numbers on the right, the strategies below.

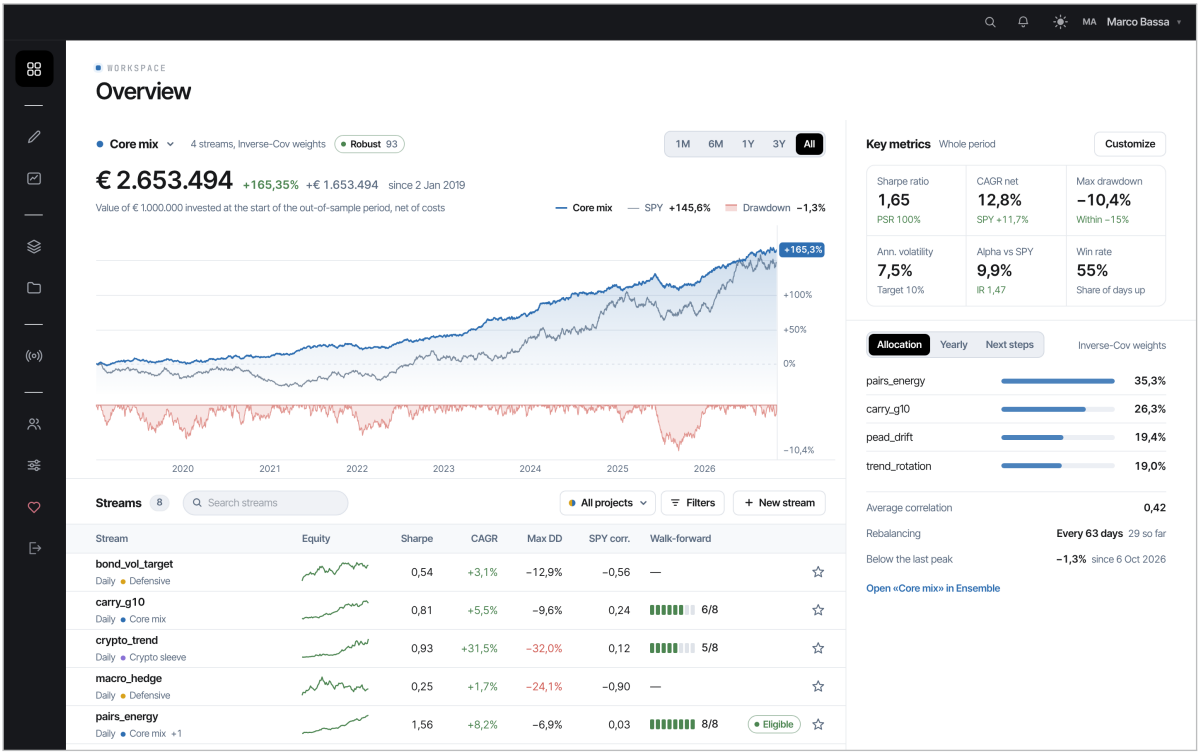


Figure 2.1. The home screen. At the top left the portfolio and its value over time; on the right its key metrics and, below them, its weights; at the bottom the list of every stream.

2.1 The portfolio

The portfolio is a saved **Project**: several strategies combined, with the weights decided in Ensemble (Chapter 5). Its name is at the top left. Next to it, how many streams it holds and how they are weighted.

2.1.1 Which project is shown

Press the name to choose. **Automatic**, the starting choice, shows the project with the highest Sharpe ratio, and says which one that is right now. Any other choice is remembered, on this computer and the next time you open the page. At the bottom of the same menu, **Open in Ensemble** opens the project that is on screen.

The small label after the name is the robustness check of the whole portfolio: **Robust**, **Doubtful** or **Likely overfit**, with a score from 0 to 100. It weighs whether the Sharpe ratio is statistically significant and whether the results hold across periods. Rest the pointer on it for the reasons; press it to open the project with all the details.

2.1.2 The value

The large number is what €1.000.000 invested at the start of the out-of-sample period would be worth today, net of estimated costs. Next to it, how much it changed over the chosen period, as a percentage and in euros. The buttons **1M**, **6M**, **1Y**, **3Y** and **All** choose the period; **All**, the starting one, goes back to the first day of the test.

2.1.3 The chart

Three things are drawn on it.

The portfolio
: The blue line and its shading. The scale on the right is the change since the start of the period; the coloured label on the scale is today's.

SPY
: The grey line: the S&P 500 over the same days, starting from the same point. It answers the first question anybody will ask about a strategy — would I have done better just buying the market?

Drawdown
: The red band underneath: how far the portfolio is, day by day, below its previous high. The deepest point is written on the scale.

SPY and the drawdown can be switched off and on by pressing their names in the legend, above the chart. The page remembers it, together with the period.



Figure 2.2. Reading one day. The large number, the change, SPY and the drawdown all refer to the day under the pointer; the date is written on the bottom scale.

Reading a single day

Move the pointer over the chart: the large number, the change, SPY and the drawdown in the legend all switch to the day under the pointer, and the date appears on the bottom scale. Move away and they return to today. With the keyboard: select the chart with the `Tab` key, then the arrows move one day (with `Shift`, ten), `Home` and `End` jump to the ends, `Esc` returns to today.

Until a project has been saved, an invitation to create one takes the place of the chart. It is not an error: there is simply nothing to draw yet. The same place explains the other cases where there is no value to show — a project with fewer than two streams, or made only of streams that have never been backtested.

2.2 Key metrics

Six numbers, meant to be read together. They are computed on the whole out-of-sample period — the data the strategies did not see while they were being built — whatever period the chart is showing. Under each number, a short note puts it in context: green when it is a good sign, red when it deserves a look.

Metric	What it tells you
Sharpe ratio	How much it earns relative to how much it swings. The note, **PSR**, is the probability that the true Sharpe is above zero rather than a lucky draw: below 95% it is not proven.
CAGR net	The compound annual return, already net of estimated costs. The note is SPY's over the same years.
Max drawdown	The largest loss from a previous peak: 'how much would I have had to endure before recovering'. The note compares it with the 15% limit for capital.
Ann. volatility	How much it swings in a year. The note is the target the system aims for.
Alpha vs SPY	How much it earns in a year *beyond* the market, once the part explained by the market itself is removed. The note, **IR**, says how steadily it does so. A strategy with a good return and an alpha close to zero is buying the market under other names.
Win rate	The share of days in profit. On its own it says little: you can win nine times out of ten and still lose.

Which numbers to look at first

Max drawdown and **Alpha vs SPY**. The first because it is the figure that decides whether a strategy can be lived with: an excellent Sharpe with a 45% drawdown is a strategy nobody holds to the end. The second because it tells a strategy that does something apart from one that follows the market with longer strides.

A backtest is not a forecast

It says how a strategy *would have* behaved on past data, with estimated costs. It does not say how it will behave. Chapter 4 explains which signs tell a strategy that worked from one that got lucky.

2.2.1 Choosing the metrics

Customize opens the full list: twelve metrics, each with its current value and a switch to show or hide it (Figure 2.3). To change the order, drag a card to its new place; with the keyboard, select the small handle that appears in its corner and use the arrows. **Reset order** puts them back in the standard order.

New metric builds one of your own without writing code: combine two measures (return over volatility, for instance), average a rolling series, or count the days on which a condition holds. The preview shows its value on the portfolio on screen before you save it. A saved metric is a file in the metrics folder, and it appears in the reports too.

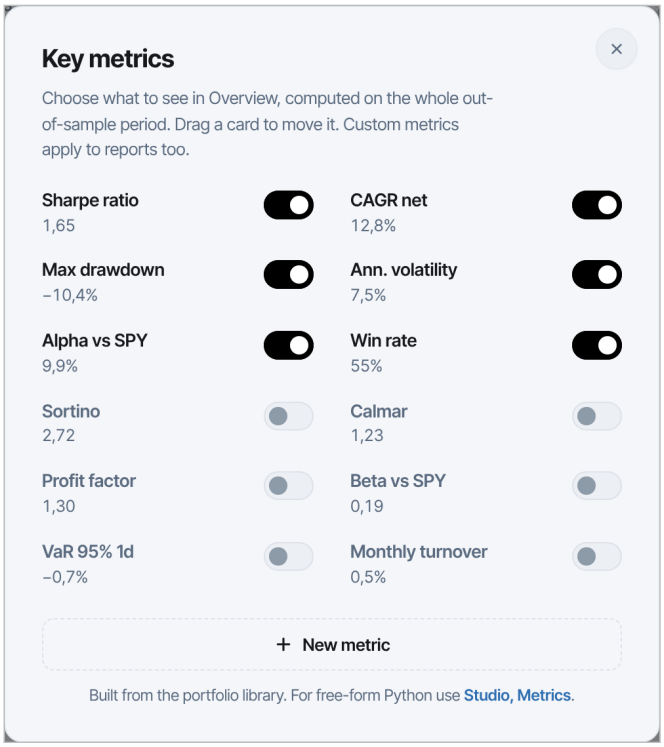


Figure 2.3. The metric list behind **Customize**: the six on screen are switched on.

2.3 Allocation, years and next steps

Below the metrics, three tabs about the same portfolio. The page opens on **Allocation** when there is a project, on **Next steps** when there is not yet one, and after that on the tab you chose last.

Allocation
: The weight of each stream at the last rebalance, the largest first. Underneath, three facts about the mix: the average correlation between its streams (the lower, the more they diversify each other), how often the weights are recomputed, and how far the portfolio is below its last high.

Yearly
: Every calendar year: return, deepest drawdown and Sharpe. The current year is marked ‘to date’.

Next steps
The four steps from an idea to a running strategy: build a stream in Studio, validate it with walk-forward, combine streams in a project, go live. They tick themselves off as you do them, and the next one carries the button that takes you there. It is the path this guide follows in the next chapters.



Figure 2.4. The right column with the **Yearly** tab open.

2.4 Streams

The list of every stream with results, one per row. Click a row to open its page; **New stream** leads to Studio, where streams are built.

Equity
The out-of-sample curve, whole period: green if it ends higher than it started, red otherwise.

Sharpe, CAGR
As above, for the single stream.

Max DD
The deepest drawdown; in red beyond the 15% limit for capital.

SPY corr.
The correlation of its daily returns with SPY: close to 1 means it moves with the market, close to 0 that it goes its own way. The correlation between the streams of a portfolio is in the **Allocation** tab.

Walk-forward
How many walk-forward folds closed in profit, one mark per fold. A dash means the stream

has not been through walk-forward yet.

Under the name, the timeframe and the project the stream belongs to. The **Eligible** label marks the streams that pass every out-of-sample check for capital: Sharpe at least 1, return above zero, drawdown within 15%, every walk-forward fold in profit. Press it to see the four checks one by one, and the next step: a live track record. The star marks the streams you want to watch.

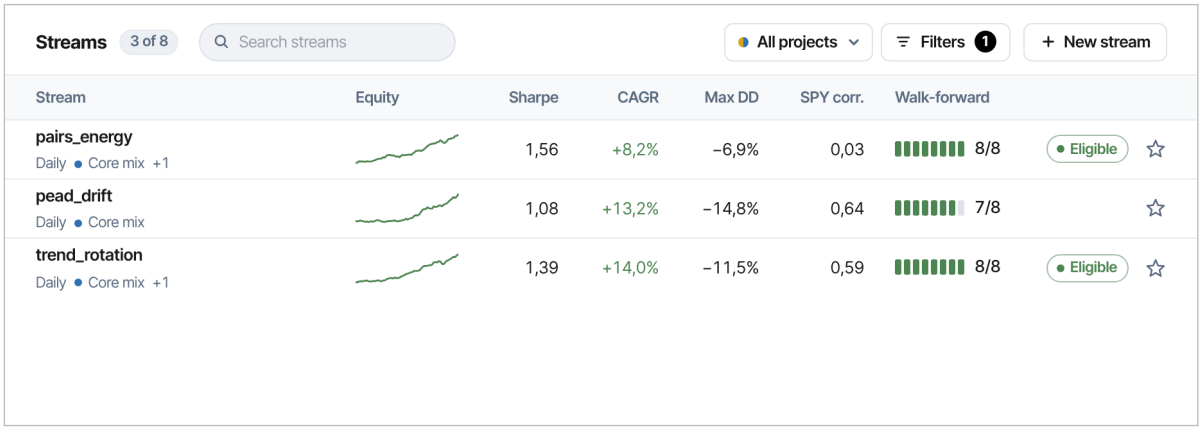


Figure 2.5. The list filtered to the streams with a Sharpe of at least 1: three of eight.

2.4.1 Sorting, searching, filtering

Click a column title to sort by it: highest first, then lowest first, then back to the original order. The search box finds a stream by name. Above the list, two menus:

All projects
: Shows the streams of one project, with their weight in it and a strip with the project's numbers, or those that are in no project yet. The × on the strip returns to all of them.

Filters
: A minimum Sharpe, a minimum return, only the streams eligible for capital, only the ones you watch. The button counts the active filters; **Clear filters** removes them all.

Sorting, filters and stars are remembered on this computer.

2.5 On a smaller screen

On a wide window the Overview fits the screen: the page does not scroll, only the lists do. Below about 1 100 pixels the four areas stack one under the other and the page scrolls; the list of streams scrolls with it, and its column names stay in view at the top. On a phone the list keeps the essential columns.

3

Studio

Studio is where things are built. You get there from **Studio** in the menu, or from **New stream** on the home screen.

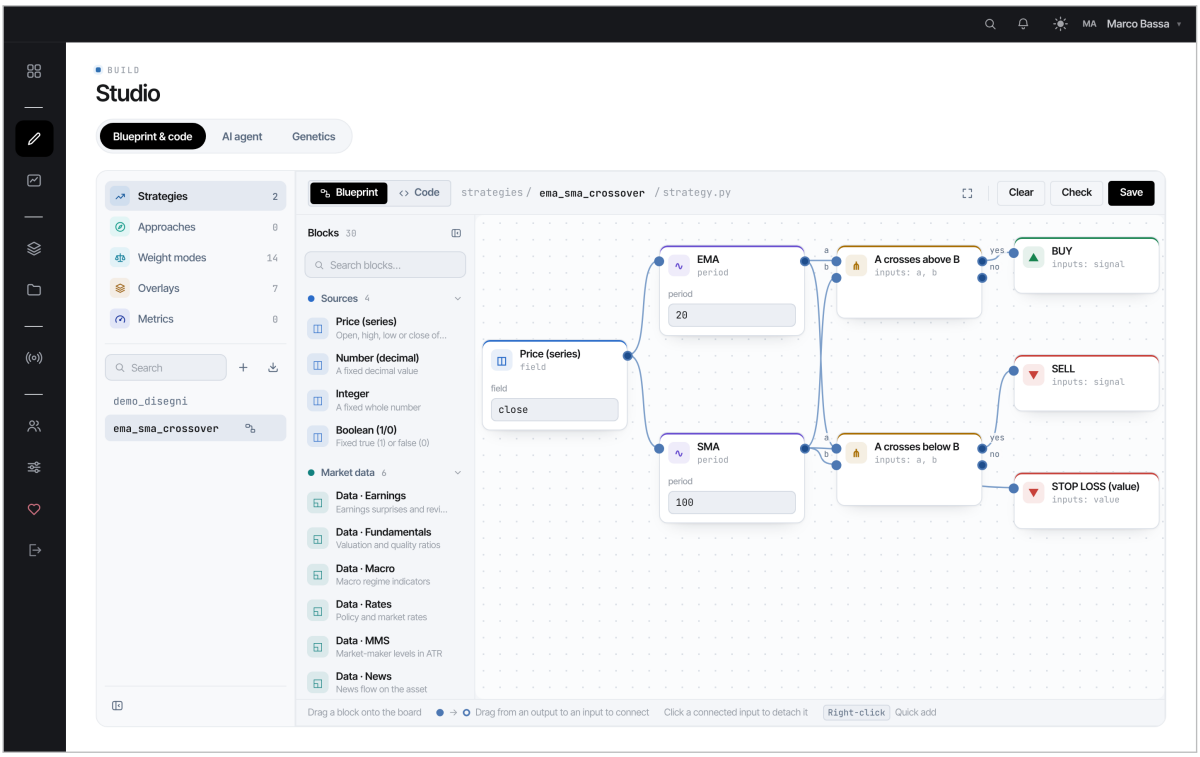


Figure 3.1. Studio, **Blueprint & code**. On the left the five families, with how many of each are in the system, and the strategies; on the right a strategy built with blocks, open on the board next to the block library.

The page is one surface as tall as the window. On the left you choose *what* you work on; on the right you work on it: a header that says how you are writing and where the file goes, the board or the code below it, and a line at the bottom that reminds you of the rules of what you are writing.

3.1 The five families

The column on the left says *what* you are building, and next to each family how many are already in the system. They are five different things, and it is worth knowing which one you need before you start.

Family	What it is
Strategies	A strategy: it decides what to buy and sell, and when. It is the thing you build most often.
Approaches	An approach: it skips the whole engine and produces the series of gains and losses directly. It is for people who already have a calculation of their own and just want to bring it into the system.
Weight modes	A rule for sharing capital among several strategies. Fourteen come already installed.
Overlays	A brake: it reduces exposure when the market behaves in a certain way. It does not earn, it protects.
Metrics	A measure of your own, a number or a chart, shown on the page of every stream next to the default ones. It can read all the data loaded in the system (Section 3.4).

3.2 What is in the system

Under the families is the list of the chosen one: what is in the system, by name.

- **Open** — a click on a name. The name goes into the header, and the list marks it. The small mark with two blocks next to a name says that it was built with blocks: it opens on the board as well as in code.
- **Search** — the field above the list filters it as you type, and shows the part that matches.
- **New** — (+) asks for the name first, at the top of the list: lowercase letters, digits and underscores, starting with a letter. A name that is taken is refused there, before anything is written. `Enter` creates it, `Esc` lets it go.
- **Import** — the button with the arrow into a tray brings in an artifact written elsewhere (Section 3.7).
- **Delete** — the bin appears when the mouse is over a name. Studio asks first; the backtest results stay.

The arrows move through the list, and `Delete` on a name deletes it, after asking. The button at the bottom of the column reduces it to the icons of the families, to give the board or the code more room. Studio remembers the family you were on and how you left the column.

3.3 The three ways to build

The switch at the start of the header, **Blueprint** or **Code**, chooses how you write; the **AI agent** has a tab of its own, next to **Blueprint & code**. The result is the same — a code file installed in the system — and only the road changes. Next to the switch the header shows where the file goes: the folder, the name — which you write in place — and the file.

3.3.1 Blueprint: with blocks

You drag blocks from the library on the left onto the board and connect outputs to inputs. The library is divided into categories: prices, numbers, external data such as earnings and fundamentals. The colour of a block says its category: sources in blue, market data in teal, indicators in violet, calculations in slate, conditions in amber, orders in green and red.

Check turns the board into code and checks it, without saving anything. Save does the same and, if the check passes, puts the strategy in the system: it appears in the list with the mark of the blocks, and in **Run backtest**. The result shows under the board — what is wrong, if anything, and the code made from the blocks under **Generated code** — and the header says when the board has changes that are not saved yet. Clear empties the board, after asking.

The quickest way to see how it works

Press Load an example: an example appears, already wired (the crossing of two moving averages with a stop). Take it apart and put it back together: in ten minutes you understand the mechanism without writing a line.

3.3.2 Code: by writing

Code switches to the text editor. A strategy or an approach is a folder with several files: they are listed on the left of the code, and + file adds one. Save keeps the file, and Save & test also runs a backtest and shows what it writes, in a panel under the code. `Ctrl+S` saves whatever you are looking at.

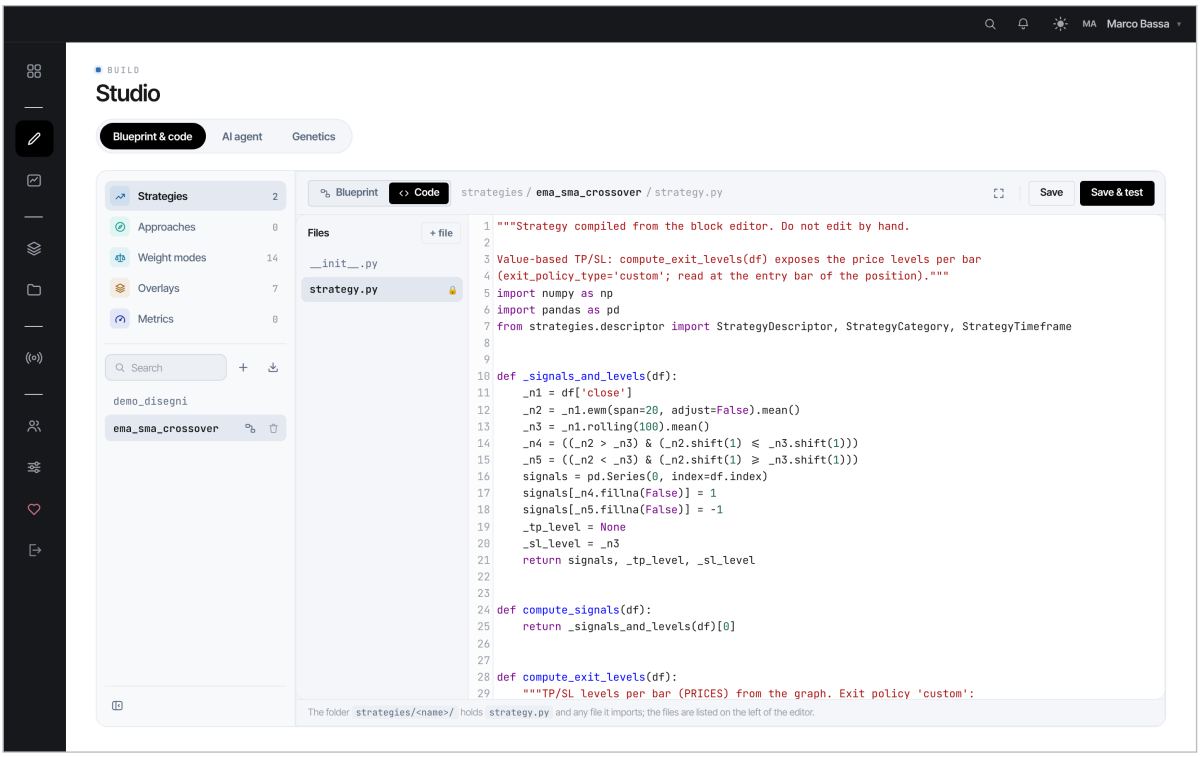


Figure 3.2. The same strategy in **Code**: the files of its folder on the left, the code made from the blocks on the right, Save and Save & test in the header.

The board and the code each keep their own artifact: you can have one open on the board and another in the code, and the list marks the one of the view you are looking at. A click on a name opens it in that view; a strategy written by hand has no blocks, so it opens in the code, and Studio says so. The other families are written in code only.

A strategy written in code can also draw what it computes, to see it later in the **View** next to its trades (Section 4.3.2). `chart(symbol)` gives the surface of an asset; each call adds the point of the current bar:

Drawing what the strategy looks at

```
from core.chart import chart

ch = chart(symbol)
ch.plot(fast.iloc[-1], "SMA 20", color="#4c8dff")
ch.plot(rsi.iloc[-1], "RSI 14", pane="RSI")            # a pane of its own
ch.hline(70, "Overbought", pane="RSI", color="#ef5350")
if crossed_up:
    ch.label(None, None, "cross", shape="arrow_up", position="below")
```

There are also `fill`, `bgcolor`, `line`, `box` and `vline`; the strategy guide in the code (`strategies/GUIDE_STRATEGIE_CUSTOM.md`) lists them all. Drawing never changes the results and does nothing in live trading: the same code runs everywhere.

3.3.3 AI agent: in words

The **AI agent** tab opens a conversation: you describe the strategy in words and the model writes it, installs it, and can run a backtest and compare it with what you already have. It needs a model installed locally, through Ollama; without one, the tab says so and the other two ways keep working.

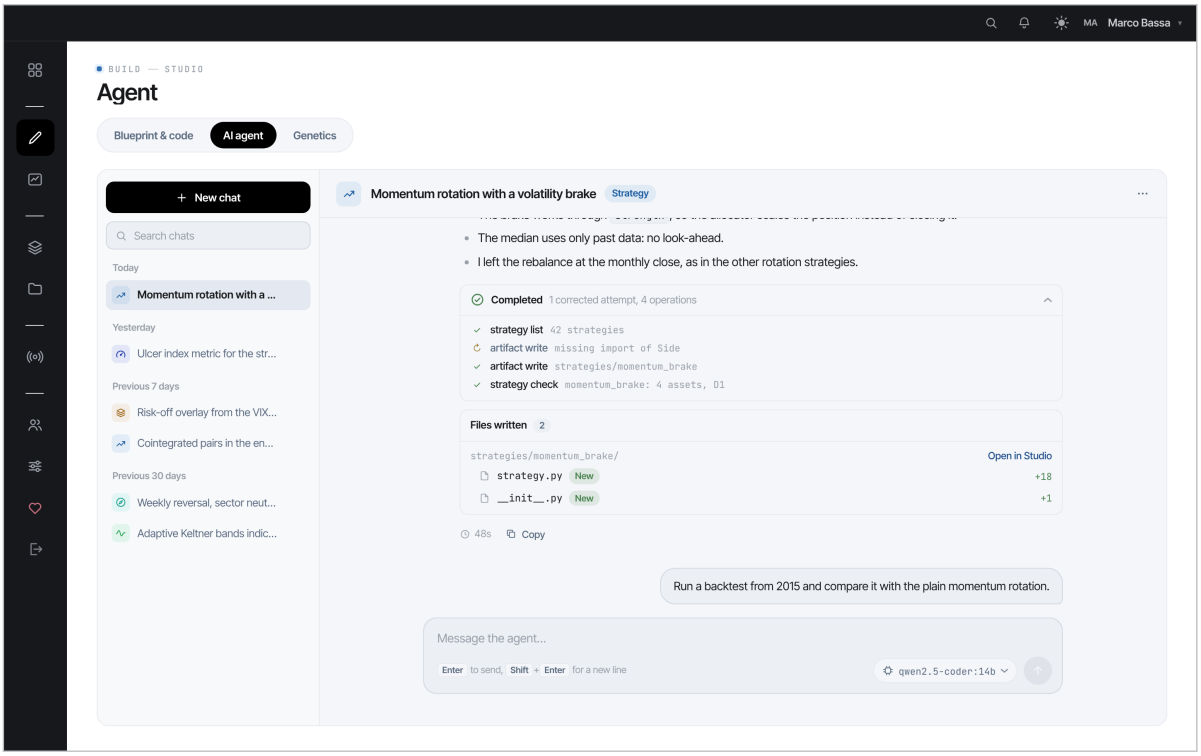


Figure 3.3. A conversation with the agent. On the left the chats, grouped by date; on the right the open one, with what the agent did under each reply: the operations, and the files it wrote with a link to open them in Studio.

The page fits the window: the conversation scrolls, the field where you write stays at the bottom.

- **The chats** — in the left column, grouped into **Today**, **Yesterday**, **Previous 7 days** and so on, each with the icon of what it is about (strategy, metric, overlay, indicator...). The field above filters them by title. **New chat** starts a new one. They are saved on your

computer, in `data/chat/`.

- **The title** — the model writes a short one after your first message. To change it, double-click it in the header, or use the `...` menu, which also deletes the chat.
- **The replies** — they read like pages: headings, lists, tables, and code with the name of its file, coloured, with a Copy button. A comparison between streams arrives as a chart with its table of numbers.
- **What the agent did** — under the reply, a line says whether the work was **Completed** and how many operations it took, including the attempts it had to correct. Click it to see each step. Below, the **Files written**, new or modified, with the lines added and removed, and Open in Studio.
- **Writing** — `Enter` sends, `Shift+Enter` goes to a new line. The model is chosen in the field itself, from the ones installed in Ollama.

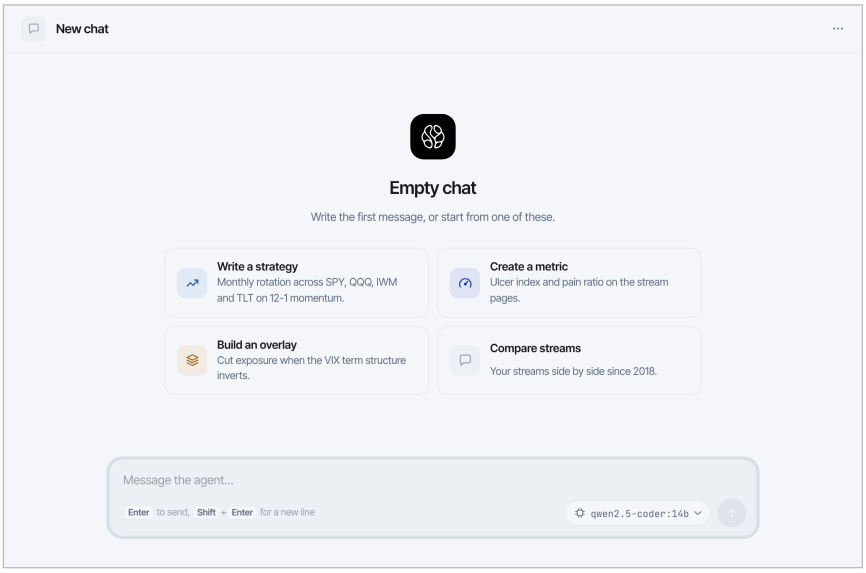


Figure 3.4. A new chat. Four starting points for the most common requests; one click writes the request in the field, ready to be edited and sent.

Read what the assistant writes

The generated code is installed in the system and runs on your data — and one day, perhaps, on your account. The assistant does not know whether a strategy makes financial sense: it knows how to write code that does what it was asked. The **Code** tab is there for this too.

3.4 Metrics

A metric is a few lines of Python on the results of a strategy. Two fields under the header say what it produces: **Output** — **Number** or **Chart** — and, for a chart, **Chart type**. **Figure** is the type for a chart with several series, points and an axis of numbers instead of dates.

A number: beta against the S&P 500

```
r_m = data.assets.returns("SPY", timeframe="D1")
both = returns.notna() & r_m.notna()
result(returns[both].cov(r_m[both]) / r_m[both].var(), "Beta vs SPY", fmt="ratio")
```

`returns` is the strategy's own series; `data` is everything loaded in the system: prices, earnings, fundamentals, rates, macro series, news, and the categories you created in Settings → Data. Where the data comes in several timeframes, the metric reads the one of the strategy, the lowest if it uses more than one; `timeframe="D1"` picks another. The full list, with a CAPM chart as an example, is in the Template of the editor and in the guide `metrics/METRIC_TEMPLATE.md`.

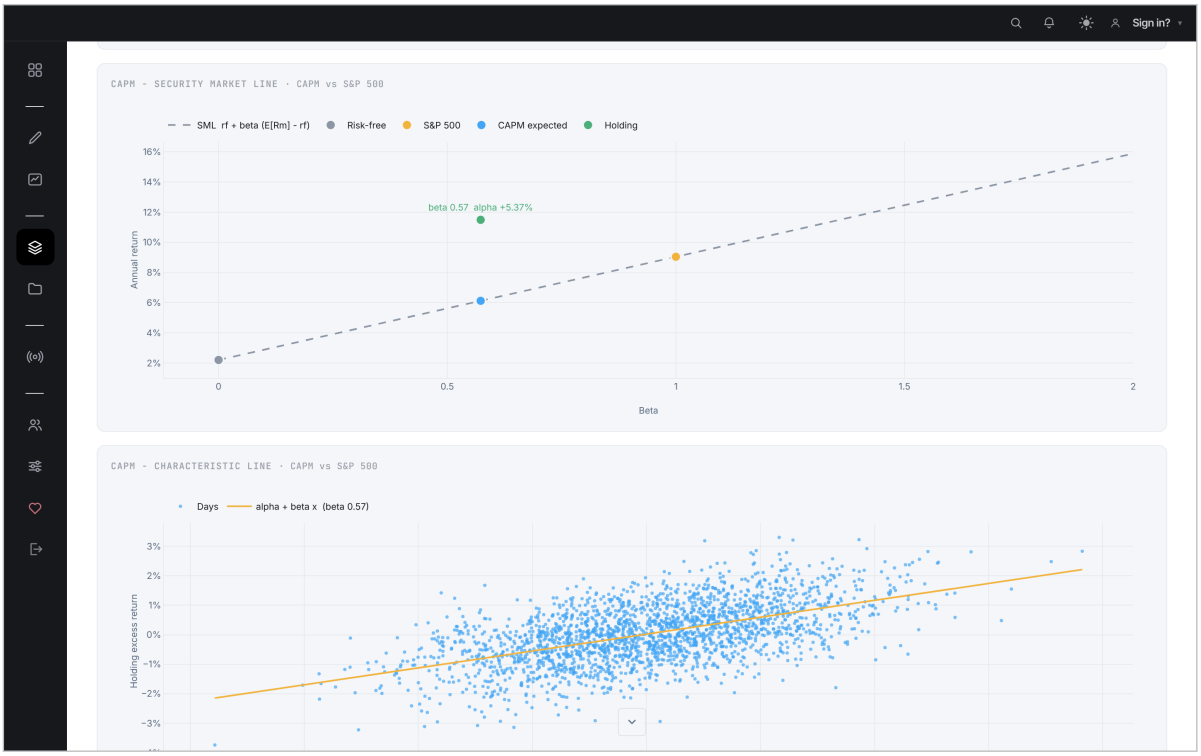


Figure 3.5. A **Figure** metric on the page of a stream: the CAPM against the S&P 500. The dashed line is what the market pays for each unit of beta; the strategy, above it, earned more than its beta was worth.

Save checks the code and saves it in one go: if something is wrong, the reason appears under the editor and nothing is saved. Validate checks without saving. If the code draws charts while **Output** says **Number**, or the other way round, the output is corrected by itself, and **Chart type** follows what the code draws.

> **Saving a metric whose data is missing**
>
> If the metric reads something that is not in the system yet, Studio says so and saves it anyway: it is not an error in the code. On the stream page it shows **Data not available** until the data arrives.

3.5 Genetics

The **Genetics** tab opens an area of its own: instead of writing one strategy, you let many of them be born and keep those that survive. Each candidate is an expression built from indicators

— the *building blocks* — and the evolution combines, mutates and selects them generation after generation, scoring each one on real data.

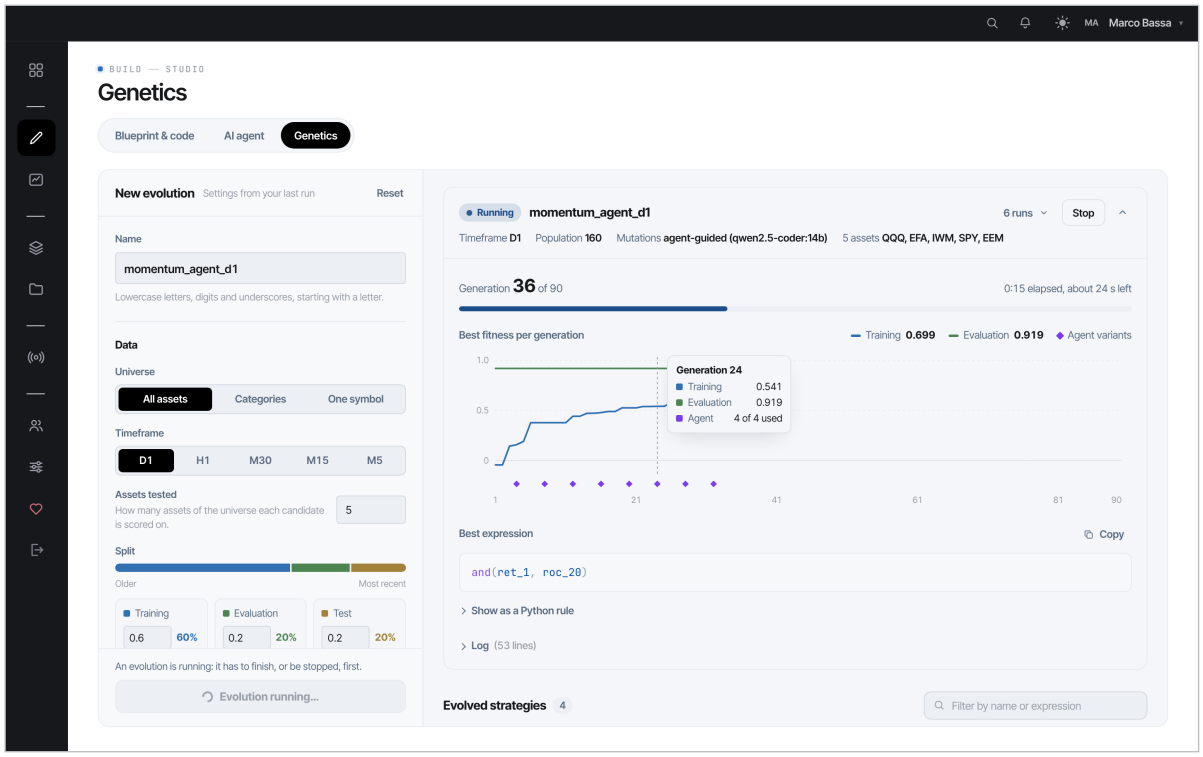


Figure 3.6. Genetics while an evolution runs. On the left you prepare the next one; on the right you follow this one: the generation, the time left, the best fitness on the training and on the evaluation data, the diamonds where the agent proposed variants, and the best expression so far.

3.5.1 Preparing an evolution

The column on the left is read from top to bottom; the **Start evolution** button stays at the bottom, and the line above it says what is still missing, if anything is (a name, a building block, a number that does not add up).

- **Name** — lowercase letters, digits and underscores, starting with a letter. It becomes the name of the strategy.
- **Data** — the **Universe** is **All assets**, some **Categories** (those without market data are greyed out), or **One symbol**. Then the **Timeframe**, and how many assets of the universe each candidate is scored on (**Assets tested**).
- **Split** — each asset's history is cut in time order: the evolution learns on the first part (**Training**), chooses on the second (**Evaluation**) and confirms on the last (**Test**). The three colours are the same everywhere on the page: in the bar, in the chart and in the results.
- **Building blocks** — the indicators the evolution can combine, grouped by family: trend, momentum, mean reversion, volatility, breakout. The indicators of the system that export a primitive are listed too. At least two.
- **Evolution** — the **Mutations** are **Heuristic** (random changes to the trees) or **Agent-guided**: every few generations a local model proposes variants of the best expression, and those outside the grammar are thrown away. Then **Population**, **Generations**, **Signal threshold** and **Seed**: the same seed with the same settings gives the same result. **Favour decision**

rules puts an if-then-else at the root of the tree, so the result reads as a rule instead of a formula.

- **Approval filter** — what a strategy must pass to be installed: how many assets must pass the checks (at least one per category, and at least two, because a rule that works on one asset may work by chance), the minimum number of trades, the risk/reward with the win rate it implies, and the share of Monte Carlo simulations it must survive.

The next time you open the page, the form has the settings of your last run (**Settings from your last run** next to the title); Reset brings back the defaults.

3.5.2 Following the run

While the evolution runs, the box on the right shows the generation, the time elapsed and the time left. The chart is the part to watch: two lines, the best fitness on the **training** data and on the **evaluation** data. Move the mouse over it to read each generation.

> **The distance between the two lines**
>
> If the training line keeps rising and the evaluation line stays where it was, the evolution is learning the training data, not the market: the rules it finds fit the past they were made on. The evaluation line is the one that counts — it is computed on data the evolution never learned from.

Below the chart, the **best expression** so far: functions, building blocks and numbers in different colours, with Copy. **Show as a Python rule** shows the same thing as the code that will be installed: reading it *before* it enters the system is the way to know what it does. Stop ends the run, and the **Log** keeps every line the evolution wrote. With more than one run in the session, the menu at the top of the box goes back to the others.

> **The run lives in the application**
>
> The evolution runs on your computer, inside the application: you can leave the page and come back, and the run is where you left it. Closing the application stops it.

3.5.3 Reading the result

At the end the box adds two panels: the **Results**, as averages per asset on the three parts of the data, and the **Approval filter**, category by category, with a bar that says how close each one came to its quota. A strategy that passes is installed by itself, and appears in **Run backtest** and in **View** like any other.

A strategy that does not pass is not thrown away: Add to the system anyway installs it, marked as such. It may have failed because a category had too few assets, or because the evaluation period was unfavourable; a backtest says more. Open in Studio and Run backtest take you to the strategy just installed.

3.5.4 The evolved strategies

Under the run, the table of every strategy born here: Sharpe on the evaluation and test data, the return and the drawdown on the test data, the approval — **Approved** with how many assets passed, or **On request** — and when it was created. The columns sort with a click; with more than one strategy, a field filters them by name or by expression.

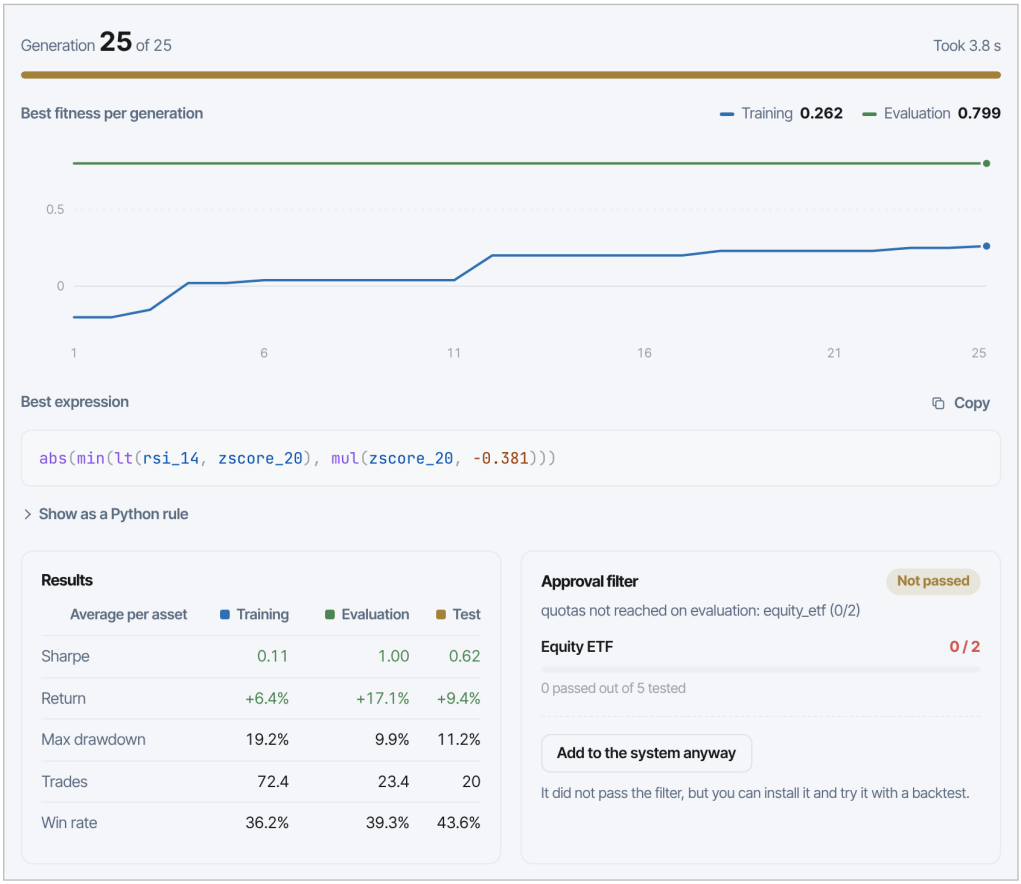


Figure 3.7. A run that did not pass the filter: no asset of the category passed the checks on the evaluation data. The numbers are still there to read, and the strategy can be installed anyway.

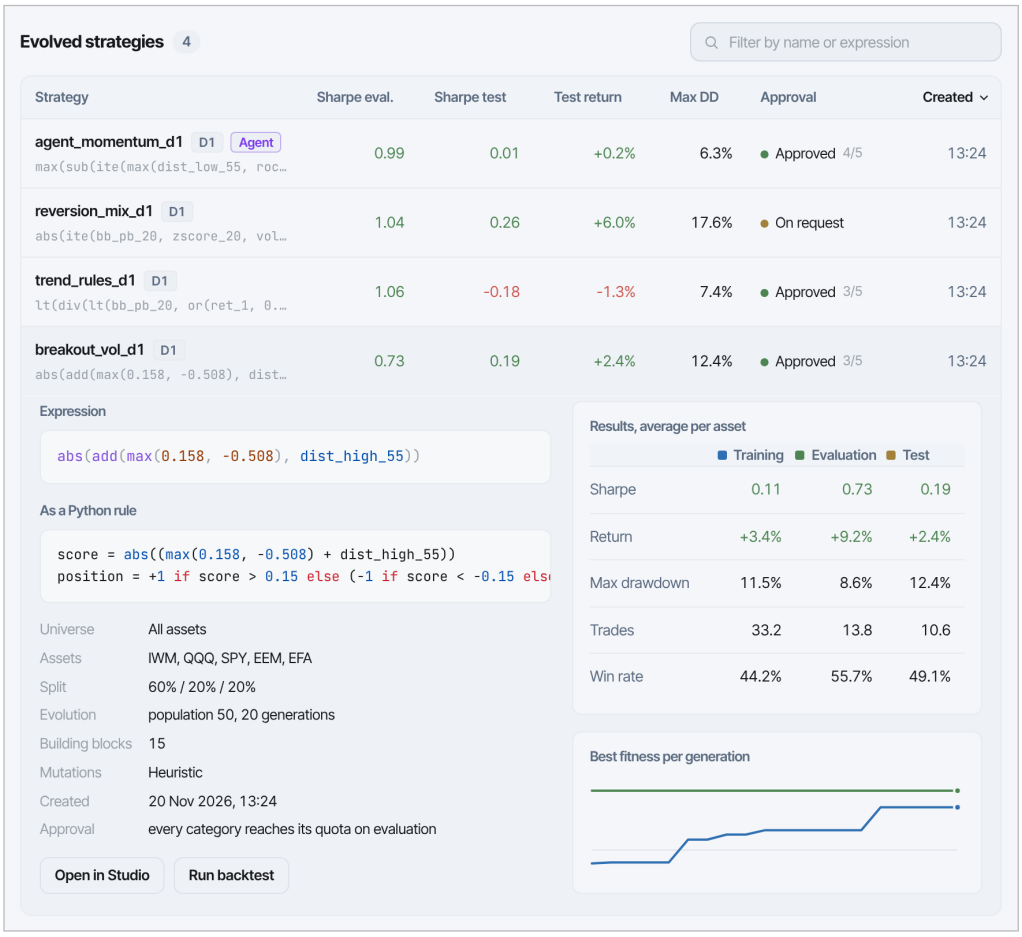


Figure 3.8. The evolved strategies, with one of them open: the expression and the rule, how it was evolved, the results on the three parts of the data and the fitness curve of its run.

3.6 What happens when you save

The artifact is written to a folder of the system, and from then on it appears in the lists: a strategy among those you can run, a weighting mode among those you can choose in a project, an overlay among those you can switch on.

Before writing, the system checks the code: it must not read or write files, must not open connections and must not run commands. If it contains any of these, the installation is refused and the reason is shown.

3.7 Bringing in an artifact from outside

An artifact written somewhere else — in another editor, received as a `.zip`, downloaded — does not have to be copied by hand into the right folder. Each family has its import button next to the search, in the column on the left (Section 3.2). The window shows the shape that family expects and what you can choose.

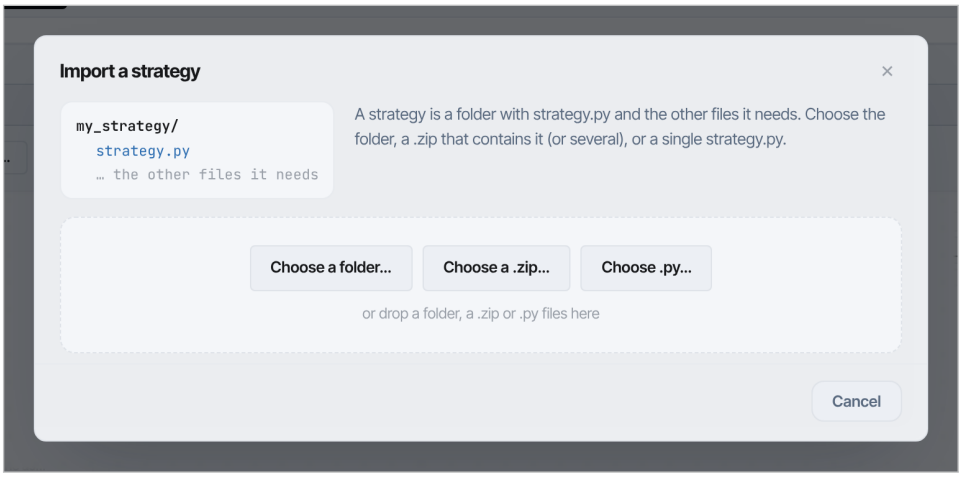


Figure 3.9. Importing a strategy: a folder, a .zip or .py files, chosen with the buttons or dropped on the dashed area.

Family	You can choose	What the import looks for
Strategies	a folder, a .zip, .py files	every folder with `strategy.py`, taken with all its files and subfolders; or a `.py` that declares the strategy, around which the folder is made
Approaches	a folder, a .zip, .py files	the same, with `approach.py`
Weight modes, **Overlays**	a folder, a .zip, .py files	the folder with `weight_mode.py` or `overlay.py` (and its `data_port.py`), or a single `.py` with `META` and `compute()`
Metrics	.py files only	each file is a metric, named after the file: a metric is never a folder

Nothing is written yet when you choose. The window lists what it found, and for each one whether it is new, already in the system, or cannot be imported — with the reason: a syntax error with its file and line, a `strategy.py` that declares no strategy, a name the system cannot

use. If the choice also holds artifacts of another family, it says so, with a link to the right section.



Figure 3.10. What a .zip holds: four strategies, all new, and a metric to import from **Metrics**.

Untick what you do not want, then **Import**. Something with the same name as an artifact already in the system is unticked: ticking it replaces the old one, which is kept as a copy in the `.studio_import/sostituiti` folder of the application. A strategy that a live runner uses, or whose backtest is running, is not replaced. At the end the window says, for each artifact, whether the system loads it — strategies are in **Run backtest** at once — or why not yet, for example a package that is not installed; then it is in the system anyway, and you open it in Studio to fix it.

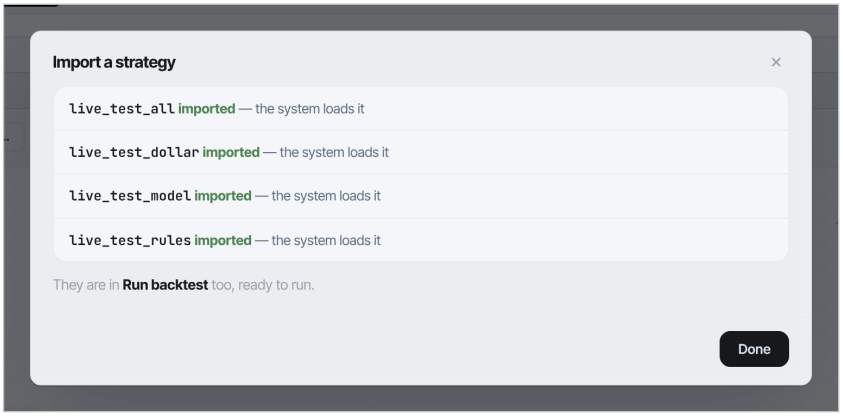


Figure 3.11. The four strategies imported: the system loads them, and they are already in **Run backtest**.

Import only code you trust

An imported artifact is code like the one you write in the editor: it is checked for its shape, not limited like the code the agent generates. As soon as it is imported the system loads it, and from then on it runs with your data and your accounts. When its code reaches beyond the data Lyra gives it — files, the network, programs, the inside of Python — the window says so before you import, with the first line it found: 'This code can reach beyond the data Lyra gives it … Import it only if you trust where it comes from'. Code bought in the marketplace may not do any of this (Section 7.4.2); code you import yourself may, and it is your call.

4

Streams

A **stream** is the result of a strategy: the series of its gains and losses over time. The **Streams** page shows them all, one card each.

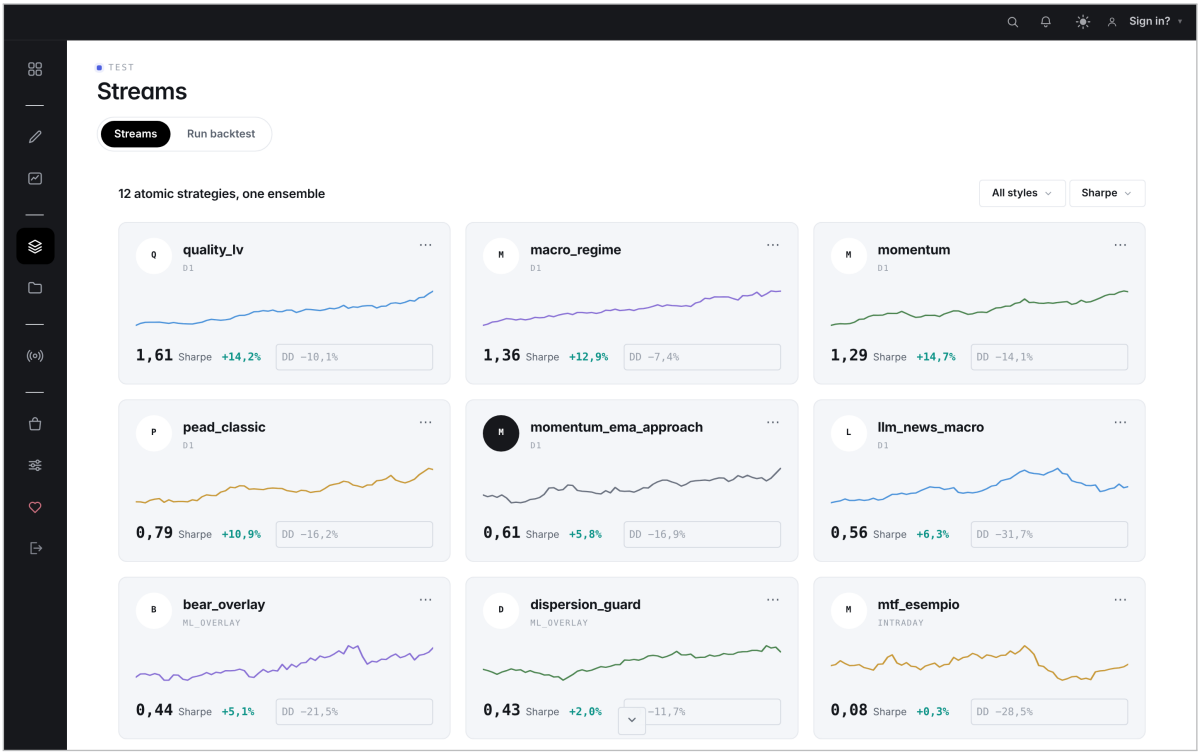


Figure 4.1. The list of results. Each card carries the performance, the Sharpe ratio, the return and the maximum loss.

4.1 Running a backtest

The **Run backtest** tab, next to **Streams**, lists everything that can be run: the installed strategies and approaches, each with its own description and a button. What you no longer use can be put aside or deleted from the card itself (Section 4.1.2).

Running it, and then

1. Press the button of the strategy you want.
2. The page shows the log live: you see what it is doing, line by line.
3. You can close the page. The job carries on, and the bell at the top right tells you when it is done.
4. The result appears in **Streams**.

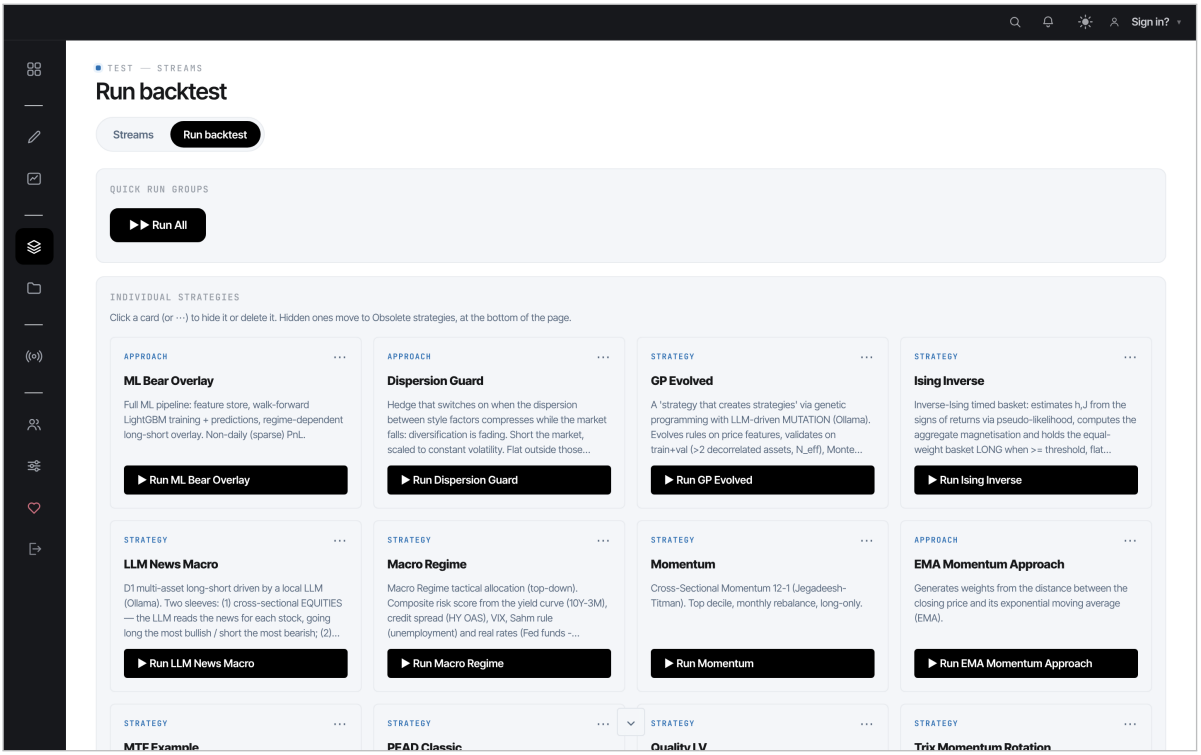


Figure 4.2. What can be run. **Run All** queues them all, one after the other.

> **One job at a time**
>
> Backtests run in a queue, not in parallel, even if you press several buttons. It is not a limit to work around: two calculations at the same time would fight over the same files and produce mixed-up results.

4.1.1 Running on the live bars

After a strategy has run live, you may want to check that its backtest does what the account did. A live runner decides on the bars its broker gives it; a backtest reads the files on the Data page, downloaded from a provider. They are not the same prices: on the 15-minute bars of the currencies, Yahoo's and Dukascopy's returns have a correlation between 0.95 and 0.97, and in a two-day test the backtest on Yahoo's bars made 13 trades where the live had made 21, only 9 of them the same.

So when the live data hold the bars a broker gave the runner for every symbol of a strategy, at its timeframe, the menu of its card offers **Run on the live bars**. The line under it says which bars: the timeframe, the broker and the period, in UTC.

A confirmation says what will happen before the job starts: the backtest uses the broker's bars for the symbols it has, and the Data page for the others — the benchmark, for instance, if the broker does not quote it. The files of the Data page do not change, and the result remembers which bars it was run on.

> **The live bars are kept only if you keep them**
>
> The bars stay after the runner stops only with **Keep the data the live downloads** on, in Settings → Data → Live data (Section 8.1.5). Turn it on before starting the runner you want to check. They do not need to be merged into the Data page: the backtest reads them where they are.

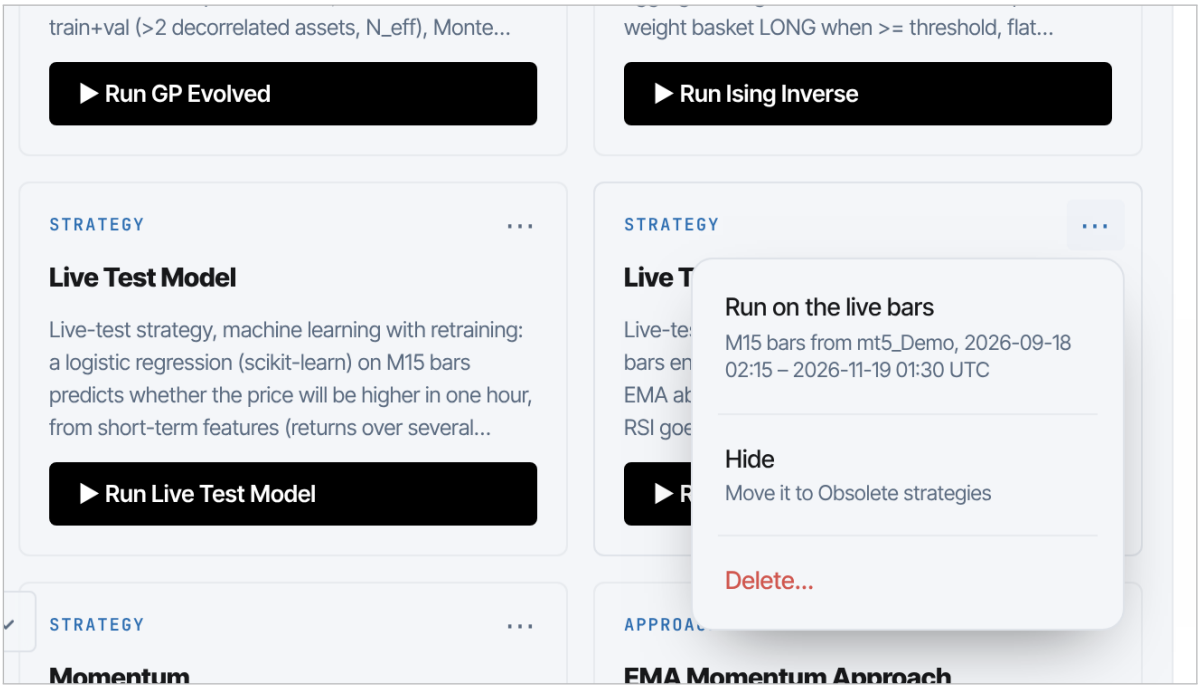


Figure 4.3. The menu of a card after a live run: the broker's 15-minute bars of the demo server, from the first to the last bar the runner received.

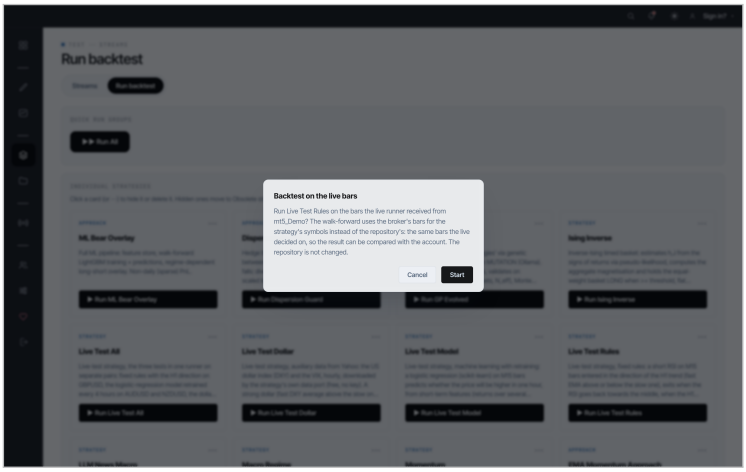


Figure 4.4. The confirmation of **Run on the live bars**.

4.1.2 Hiding a strategy, or deleting it

A strategy or an approach you no longer use does not have to stay in this list. Click its card, anywhere but on its button, or the ··· in its corner: a small menu offers two ways out.

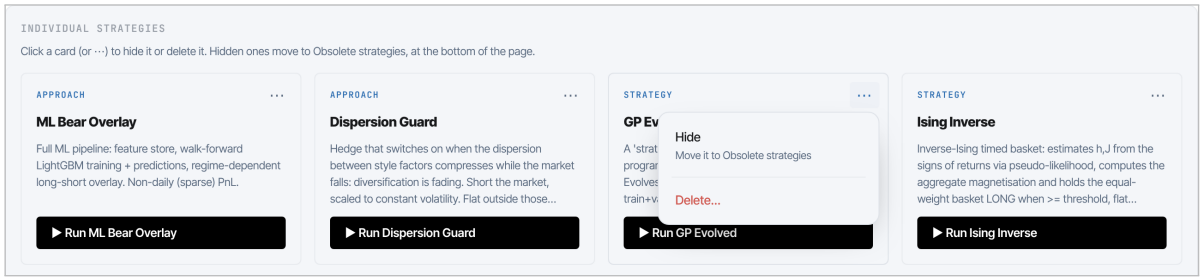


Figure 4.5. The menu of a card. **Hide** puts the strategy aside and keeps everything; **Delete** removes it for good.

Hide is the one to reach for first. The card moves to **Obsolete strategies**, at the bottom of the page, and the strategy is no longer offered here or in **Live**. Nothing is lost: its code and its backtest results stay where they are, and the message that confirms it has an Undo. If the strategy is part of a project, the message says so: show it again before running that project in **Live**.

Delete removes the strategy together with its backtest results, so its stream leaves **Streams** as well. Before anything happens, a confirmation lists what will go: the folder of the strategy, each results file and the projects that contain it.

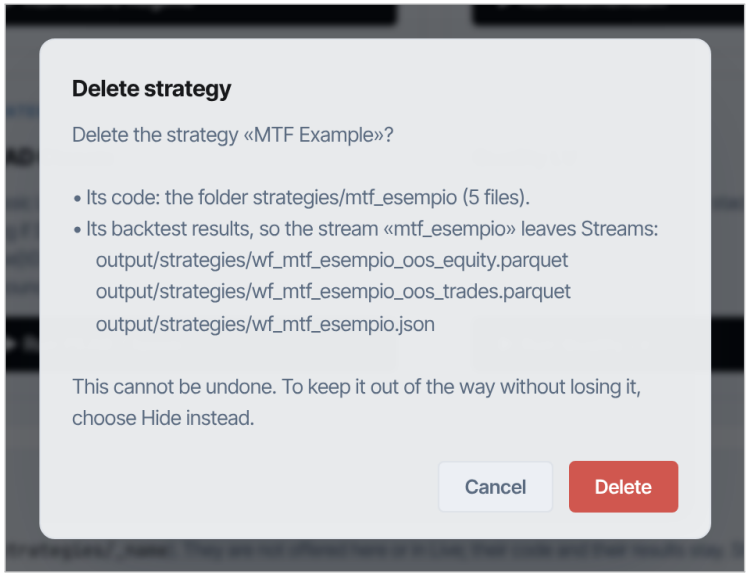


Figure 4.6. The confirmation of **Delete**. The focus starts on Cancel: pressing Enter out of habit deletes nothing.

When the page says no

A strategy that a runner in **Live** is using, on its own or inside the runner's project, can be neither hidden nor deleted: the runner could not start again without it. The same holds while one of its backtests is running. The message names the runner: stop it, or make it run something else, and try again.

4.1.3 Obsolete strategies

The section at the bottom of the page lists only the strategies and approaches you have hidden. It stays closed until you open it, the number next to its title tells you how many there are, and the page remembers whether you left it open.

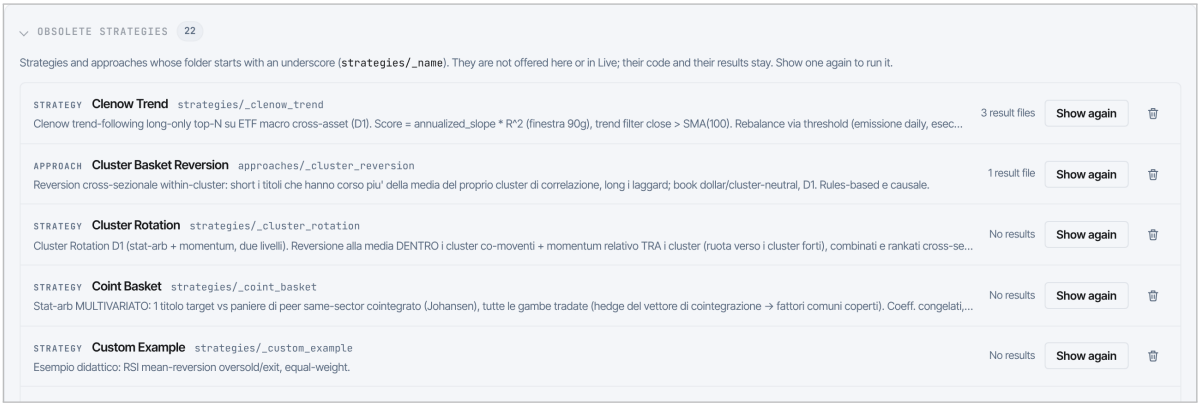


Figure 4.7. Obsolete strategies. For each one its type, its folder, its description and how many results files it still has; **Show again** puts it back in the list above, the bin deletes it.

> **What 'obsolete' means on disk**
>
> A hidden strategy is a folder whose name starts with an underscore: `strategies/_momentum` instead of `strategies/momentum`. It is the rule the system has always followed, so renaming the folder by hand does the same thing, and a folder renamed by hand shows up here too.

Two cases are worth knowing:

- **Show again** is greyed out when a visible strategy with the same name already exists: there would be two folders with one name. Delete one of the two first; deleting the hidden copy leaves the results to the visible one, which uses them.
- A strategy written long ago may no longer load, because of an import that has since disappeared, for instance. Shown again, it goes back to its folder but not to the list, and a message gives the error: correct it in **Studio**, where it now appears, or put it back among the obsolete ones with **Hide it again**.

4.1.4 An empty stream: a strategy in a project before its backtest

A project is made of streams, and a stream is born from a backtest: a strategy you have not backtested yet cannot go in a project, so it cannot run in **Live** with the others. When you want it there anyway — a live test, or to watch it trade before measuring it — give it an *empty stream*: **Create an empty stream** in the menu of its card.
The strategy now has a stream in **Streams**, drawn with a dashed border and without a curve, and you can put it in a project — together with streams that have results, or with other empty streams only (Section 5.6). A runner in **Live** runs its code like any other strategy: what it needs is the code, not the backtest.

What an empty stream cannot have, by definition, is numbers:

- **No metrics.** Its page explains what it is instead of showing empty panels; in a project, the metrics are those of the streams with results.
- **No Marketplace.** An empty stream cannot be published, nor can a project that contains

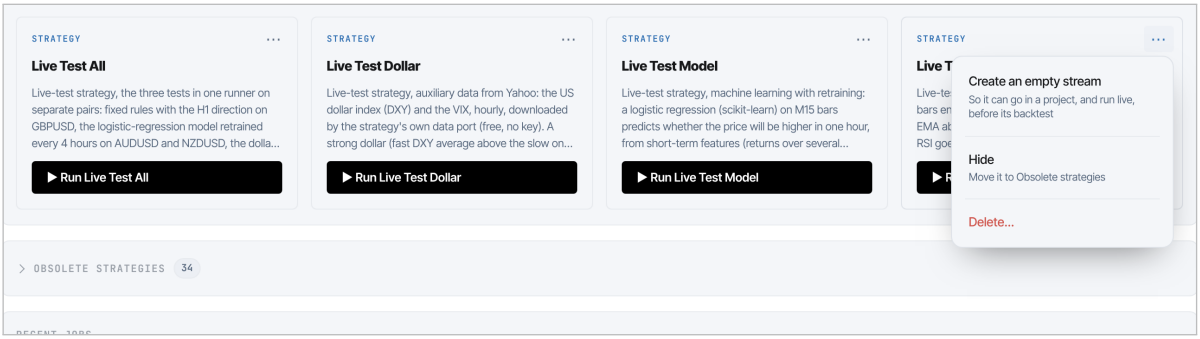


Figure 4.8. The menu of a strategy without a backtest. Once created, the card carries the **Empty stream** label, and the menu offers to remove it.

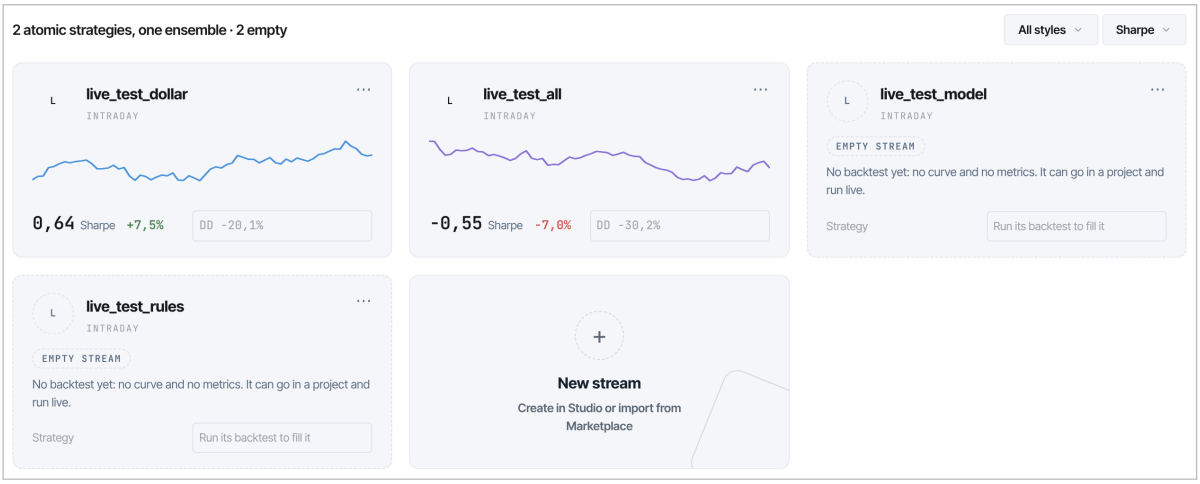


Figure 4.9. Two streams with results and two empty ones. The empty ones come last, whatever the order chosen, and the line at the top counts them apart.

one (Section 7.5).

It fills itself. Run its backtest as usual: its results replace the empty stream, and from then on it is a stream like the others, in **Streams** and in every project that contains it. To take it away without a backtest, choose **Remove the empty stream** in the same menu, or on its page: the strategy is not touched, and the projects that use it keep it, without a stream — the confirmation names them.

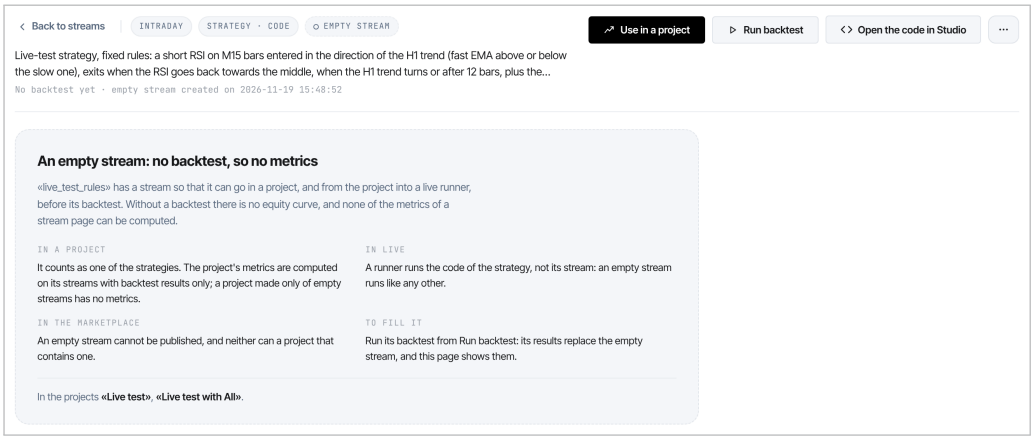


Figure 4.10. The page of an empty stream: what it is, what it can be used for, and the projects that contain it.

> **When the menu says no**
>
> **Create an empty stream** is greyed out for a strategy that does not say where its backtest results go (`output_path_pattern` in its descriptor): without that, it cannot have a stream, not even an empty one. A strategy that already has results does not offer it at all.

4.2 The page of a result

Clicking on a stream opens its page.

4.2.1 How the positions were sized

A strategy that proposes the same trade on two assets that move together — long the euro and long the pound, both against the dollar — is exposed twice to what is largely the same risk. The sizing measures the correlations between the assets, and they decide two things: how much risk the whole book takes, and how that risk is split between the positions.

How it is split depends on the strategy, not on the assets. Its signals on different assets can be:

- **the same idea, repeated** — the same rule on assets that move together, a trend followed on every currency. Two signals that agree confirm the idea; they do not add a second one. Two that disagree are a disagreement, not a hedge. The positions keep the proportions of conviction and volatility;
- **independent information** — different models or data for each asset. Two correlated positions share their risk and are reduced, the more the closer they move; two opposite ones become a position on the difference, which risks little and gets more room.

Next to the other labels at the top of the page, a backtest made with the default sizing says

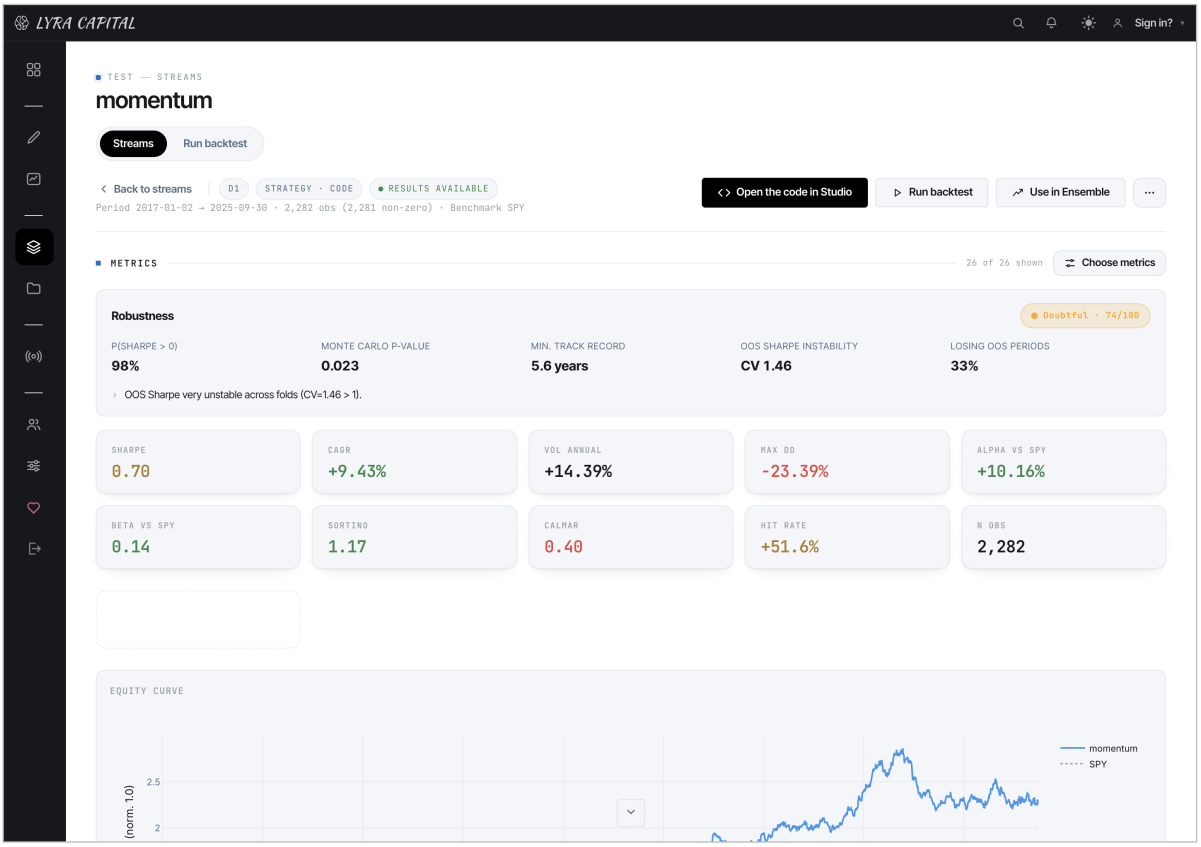


Figure 4.11. The page of a stream. Under its name the period and the benchmark; then the robustness verdict, the metrics and the curve, with SPY next to it.

where the strategy stands between the two.

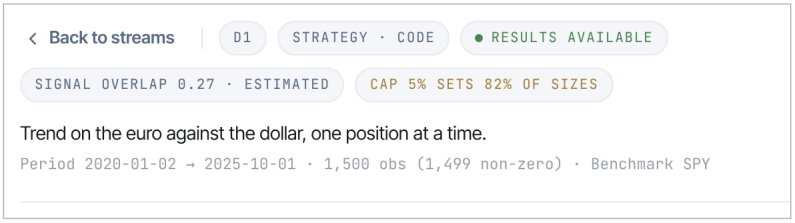


Figure 4.12. How the last backtest sized the positions: the signal overlap, estimated from the strategy's history, and a warning that the cap per position, not the sizing, decided most of the sizes.

Signal overlap 0.27 · estimated
: 1 means the same idea repeated; close to 0.1, independent information. The author of a strategy can declare it; otherwise the backtest estimates it from the strategy's own history, period by period and only from the past. It starts from 1, the safe side, because treating as independent two signals that are the same idea would lever their disagreements; it moves down only when the history proves it. The live runner uses the last value until the next backtest. The tooltip says how many days of history it rests on.

Cap 5% sets 82% of sizes
: Shown only when it happens: the cap per position stopped most of the positions, so they all took the same size and the correlations could change nothing. The cap is meant as a safety limit. Raise **Max weight per position** in Config (Section 8.2) — or ask the author to raise it for this strategy alone — and run the backtest again.

> **No label on an older backtest**
>
> The labels come from the backtest. A result made before this version, or with another way of sizing (Section 8.2), shows neither: run its backtest again.

4.2.2 Robustness: the panel to read first

It is the most important part of the page, and it is the one that answers the question 'did this strategy work, or was it lucky?'. The score on the right sums up the five numbers in one.

Indicator	What it tells you
P(Sharpe > 0)	The probability that the result is not noise. Below 95% the strategy has proved nothing.
Monte Carlo p-value	The same question asked another way: shuffling the data, how often would a lucky case have done just as well. The smaller, the better.
Min. track record	How long this strategy would need to run for its result to be credible. A value longer than the period tested means: there is not enough history yet.
OOS Sharpe instability	How much the Sharpe ratio changes from one period to the next. High means the strategy worked in some periods and not in others.
Losing OOS periods	The percentage of periods closed at a loss.

> **Why a 'Doubtful' verdict is not an error**
>
> It means the data are not enough to tell this strategy apart from chance. Not that it is wrong: that it has not been proved. It is the honest answer over a short period, and the way to change it is to have more history — not to change the parameters until the verdict improves, which is the most common way of fooling yourself.

4.2.3 The metrics

Ten numbers, unless you choose otherwise (see below). Besides those already seen on the home screen (Chapter 2):

Beta vs SPY
: How closely the strategy follows the market. Close to zero: it does something of its own. Close to one: it moves like the index.

Sortino
: Like the Sharpe ratio, but it counts only the downward swings. Rising fast is not a flaw.

Calmar
: Annual return divided by maximum loss. It answers 'how much does it earn for each point of pain'.

Hit rate
: The percentage of periods in profit.

N obs
: How many observations. Few observations make everything else less reliable.

4.2.4 The charts

Below the numbers, the charts. The first is the **Equity Curve**: what one unit invested at the start would have become, day by day.

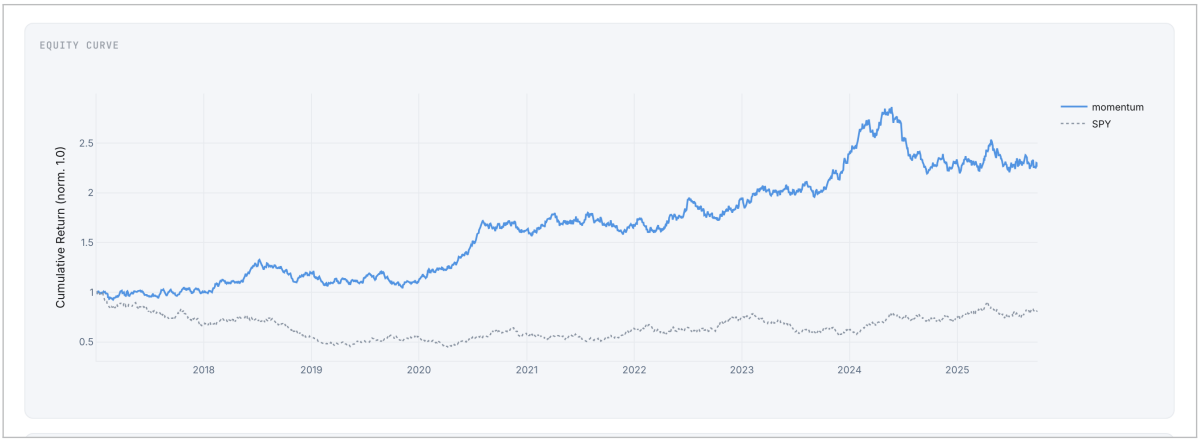


Figure 4.13. The equity curve of a stream. The dashed grey line is SPY over the same period, starting from the same point.

The dashed grey line is **SPY**, the fund that tracks the S&P 500, over the same period and starting from 1 like the strategy: the comparison you would make anyway, already drawn. For a strategy tested with the walk-forward it is SPY over the very same test periods. It is also the benchmark behind **Alpha vs SPY**, **Beta vs SPY** and the correlation chart. The results of a

project draw it in the same way (Section 5.2).

Drawdown (Underwater)
: How far below its previous peak the strategy was, day by day. The deepest point is the **Max DD**.

Rolling Sharpe Ratio
: The Sharpe ratio over the last 90 and 252 days. A line that keeps falling is a strategy losing its edge.

Rolling Correlation vs SPY
: How closely it moved with the market, period by period.

Yearly Returns, Monthly Returns Heatmap
: The result year by year and month by month.

Stress Test Periods
: How it went in periods everyone remembers: the end of 2018, the Covid crash, the rate shock and the bear market of 2022, the recovery of 2023.

4.2.5 Choosing the metrics

Every metric on the page can be hidden: the numbers, the robustness panel, the charts, the tables, and the metrics you wrote yourself in Studio. Move the pointer over one and a small eye appears in its corner: pressing it hides that metric, and the message that follows has an **Undo**. To see the whole list, press **Choose metrics**, at the top of the metrics.

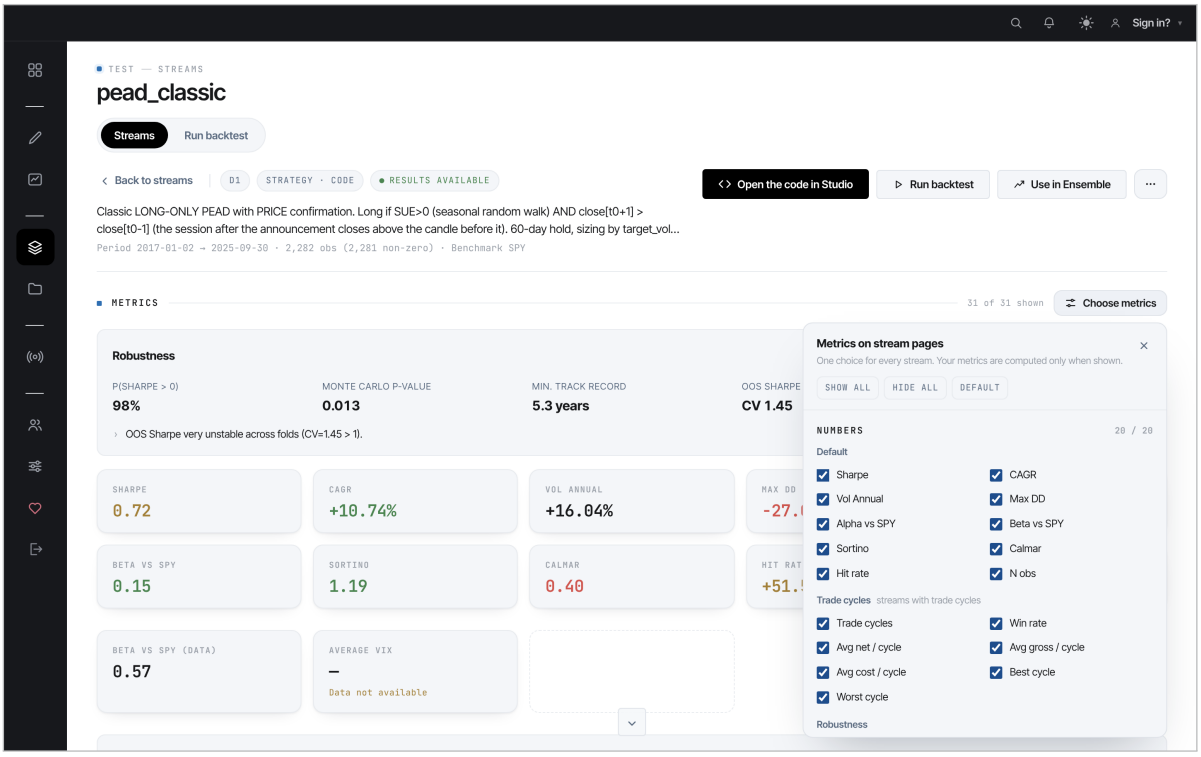


Figure 4.14. **Choose metrics**. Numbers, charts and tables, the default ones and yours, each with its tick. Next to the button, how many are shown.

- The choice holds for *every* stream: you make it once. Projects have a choice of their own (Section 5.2).

- Show all, Hide all and Default change everything at once. **Default** is how the page looks before any choice: the default metrics and your active ones.
- A metric you create after making your choice appears by itself; you hide it like the others if you do not want it.
- Your metrics are computed only when they are shown, on the results of the stream and with its benchmark.

One of your metrics can show a word instead of a value. **Data not available**: it reads data that is not in the system yet — load it in Settings → Data and the metric works. **Error in the code**: the metric has to be fixed in Studio, and the word takes you there. **No results yet**: the stream has never been run.

Adding a metric from here

Move the pointer after the numbers, or at the very bottom of the page, below the charts and the tables: New number metric and New chart metric appear. They open the **Metrics** section of Studio with the output already chosen, a template and a free name; save, and the metric is on the stream page (Section 3.4) — your numbers after the default ones, your charts at the bottom, just above New chart metric. The pencil on one of your metrics opens it in Studio, and Studio has a link back to the stream.

4.2.6 Comparing a live with its backtest

The charts of a stream are daily. For a strategy that trades every few minutes, the few days of a live run are a handful of points: too few to lay next to the account's curve. A metric can draw the equity bar by bar instead — `metrics/METRIC_TEMPLATE.md` explains how, and has the example ready to copy.

Laying the backtest next to the demo account

1. Keep the live data on (Section 8.1.5) and run the strategy live on a demo account for a few days.
2. Stop the runner. Its journal says on the first line when it started: that is the start of the comparison.
3. In **Run backtest**, open the menu of the strategy's card and choose **Run on the live bars** (Section 4.1.1).
4. In the metric write the start time as the journal shows it (`START`), your distance from UTC (`UTC_OFFSET` : 1 in Italy in winter) and the account's balance at that moment (`CAPITAL`); then choose it on the page of the stream (Section 4.2.5).
5. The chart is the backtest from the last bar closed when the live started, rebased to the balance: lay it next to the balance curve of the account's report.

The backtest counts in UTC; the metric takes your local time and the chart shows it. Two differences remain even when everything matches: the account trades at the price of the minute in which the runner's cycle runs, a few seconds after the bar opens, and pays the broker's spread and commissions, which the backtest only estimates.



Figure 4.15. A 15-minute strategy after a two-day live: the backtest equity from the hour the live started, rebased to the demo account's balance.

4.2.7 Arranging charts and tables

Charts and tables — the default ones and yours — are panels you arrange like windows on a desktop. They stay in rows, up to three side by side.

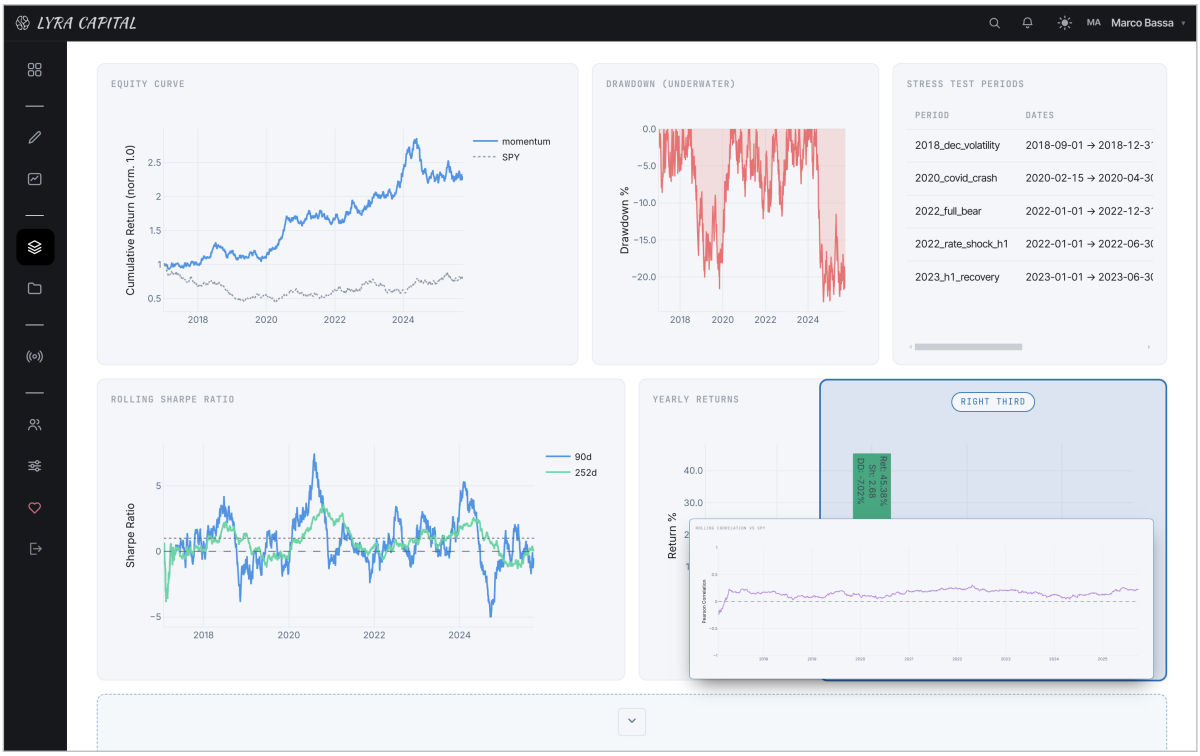


Figure 4.16. Moving a panel. Where it was, a dashed outline; under the pointer, a small copy; the blue frame shows where it will land — here the right third of the row, next to two other panels.

- **Move.** Take a panel by its title bar — the strip with its name — and drag it. A blue frame shows where it will go: next to the panels of a row (**Left half**, **Middle third**…), or a line **New row** above or below a row. Let go, and the panels slide into place. **Esc** cancels. Near the top or the bottom of the window the page scrolls by itself.
- **Make it bigger.** Pull the bottom edge of a panel down: the whole row grows, because panels side by side keep the same height. Between two panels side by side, the divider

changes their widths. Double-click the edge or the divider: back to the normal size.

- **Maximize.** Double-click a panel's title: it takes the whole width and the height of the window. Double-click again: it goes back to where it was, as it was.
- **With the keyboard.** Among the buttons that appear on a panel there is one with four arrows. With the focus on it, the arrow keys move the panel, **Shift** with the arrows change its height and width, **Enter** maximizes it and brings it back.

Like the choice of the metrics, the arrangement holds for every stream. Reset layout, in Choose metrics, brings back the original order: one panel per row, at the normal size. One of your metrics that draws several charts makes one panel for each; a new chart of yours appears at the bottom, just above New chart metric, and you move it where you want. In a narrow window the panels go one below the other.

4.2.8 Exporting to Excel or CSV

Export, next to Choose metrics, puts the data and the charts of the page in a file. The numbers are the ones on the page, not a new calculation.

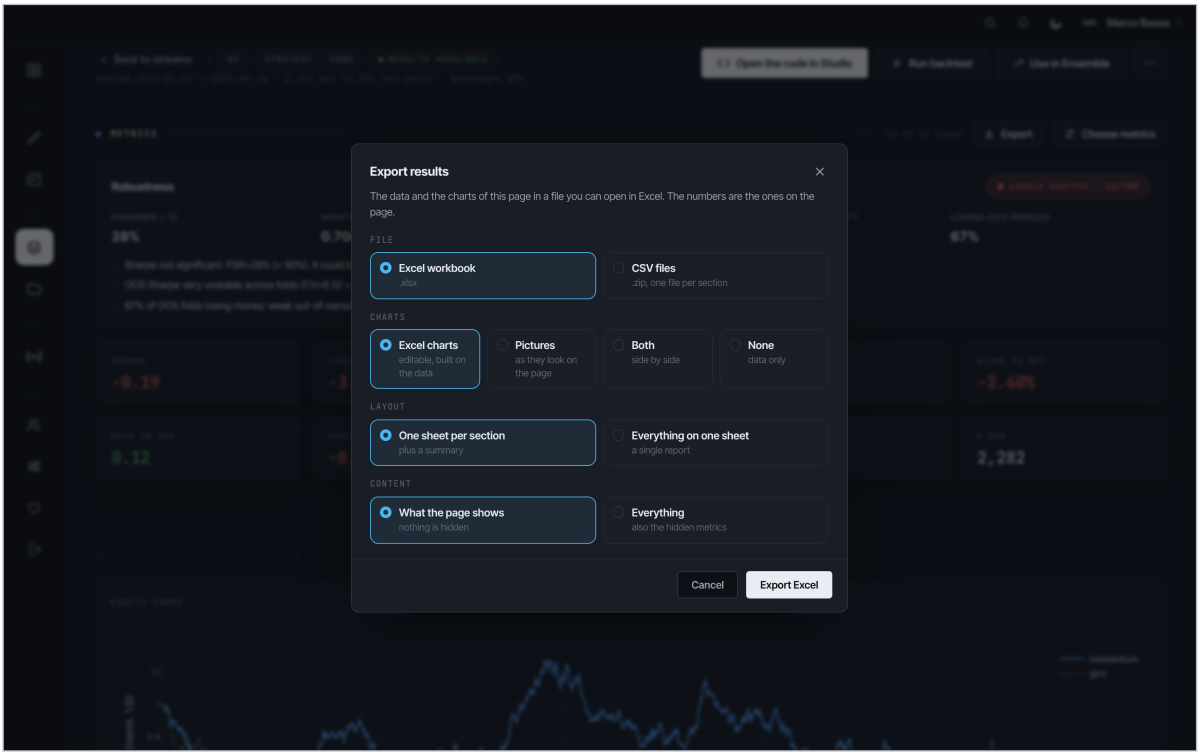


Figure 4.17. Export. Each choice says what it does; the ones that do not apply to the chosen file disappear.

- **File** — an Excel workbook (**.xlsx**), or CSV files in a **.zip**, one per section.
- **Charts** — **Excel charts**, built on the data in the file, that you can change in Excel; **Pictures** of the charts as they look on the page; **Both**, side by side; or **None**. The heatmaps (monthly returns, correlations) become tables coloured like the page, because Excel has no heatmap chart. With CSV the pictures are PNG files next to the data.
- **Layout** — **One sheet per section**, with a **Summary** sheet first (the details, the metrics, the robustness panel, your numbers and a link to every sheet); or **Everything on one sheet**, a single report with the metrics and the charts on the left and the long series on the right.

- **Separator** (CSV) — **;** with the decimal comma, the way Excel in Italian opens it, or **,** with the decimal point.
- **Content** — **What the page shows**, or **Everything**: also what you hid with Choose metrics. Your hidden metrics are computed for the file.

In Excel the percentages are real percentages (12.3% is the number 0.123 with the percentage format) and the dates are dates: you can compute with them. In the CSV files they are in points, with **(%)** in the column name. The sheets follow the order of the panels on the page, and the choices stay for the next time.

4.2.9 The three buttons at the top

- Open the code in Studio — opens the artifact in the editor.
- Run backtest — runs the calculation again.
- Use in Ensemble — takes this stream into the project builder (Chapter 5).

4.3 View: seeing what it did

The **View** item in the menu shows prices with the strategy's trades on top: where it went in, where it came out, and what it was looking at when it decided — the lines it compared, the levels it watched, the zones it waited for, if its code draws them.

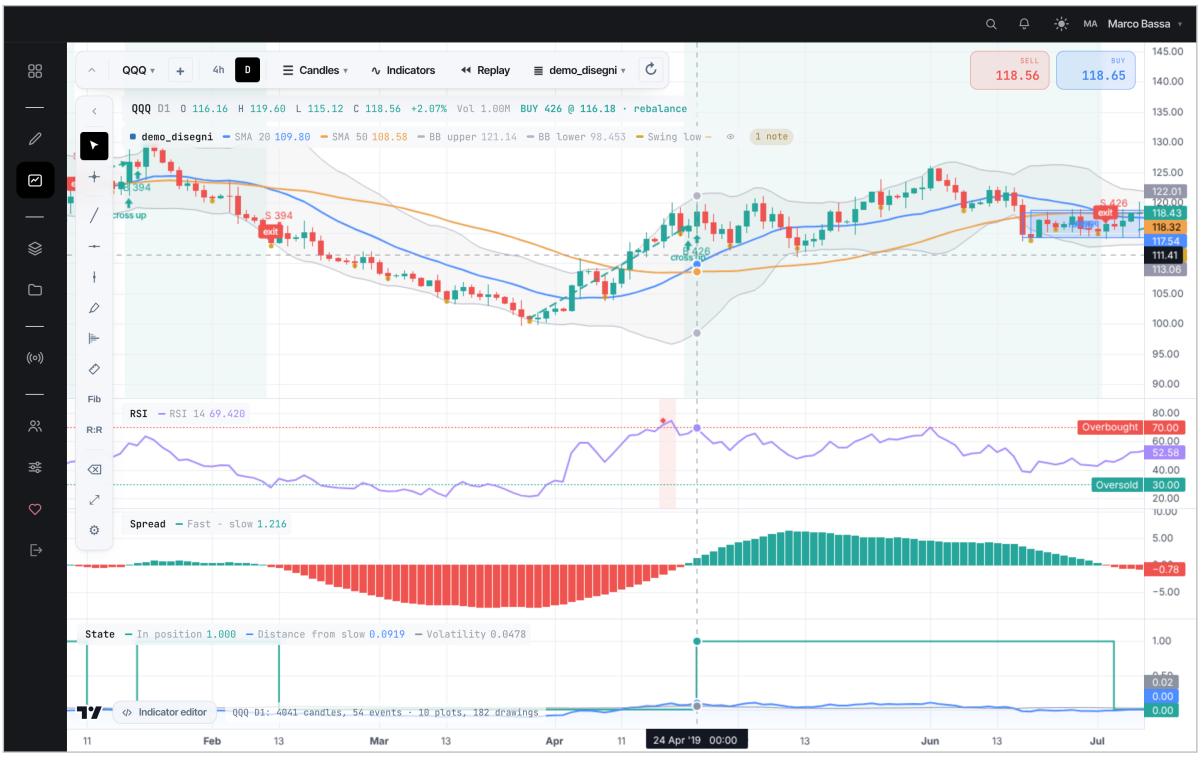


Figure 4.18. The chart view with a strategy that draws: two moving averages and the Bollinger bands on the price, its labels and ranges, the RSI, a histogram and the position in panes of their own. The cursor is on a trade: its price and reason are in the legend.

It is for something the numbers do not say: *whether the strategy is doing what you thought.* A curve that rises nicely can hide twenty sensible trades and a single huge lucky one.

4.3.1 The page

The chart takes the whole page. Three panels sit on top of it, and each of them can be reduced when you want to see what is underneath:

- **The bar at the top** — the instrument, the timeframe, the type of chart, **Indicators**, **Replay** and the strategy whose trades you are looking at.
- **The drawing tools**, down the left side: lines, the ruler, Fibonacci, the Volume Profile over a range and, at the bottom, the gear that opens the settings.
- **The indicator editor**, a panel along the bottom of the page where you write an indicator in Python and see it on the chart.

The arrow at the start of a bar reduces it to a small tab in the same place: a click on the tab brings it back, without moving the mouse. The legend moves into the free corner, so that nothing covers it.



Figure 4.19. The indicator editor at the bottom of the page, with a preview on the chart: the bands on the price and the ATR in a pane of its own, just above the editor.

The editor opens from Indicator editor at the bottom left and takes the lower part of the page, the whole width, as on TradingView. The chart gets shorter above it instead of disappearing underneath: the dates, the panes of the preview and the TradingView logo stay in sight. Drag the top edge to make it taller or shorter (or use the arrow keys when the edge has the focus); the button with the two arrows makes it fill the page up to the bar at the top, and brings it back; the arrow next to it closes it into the button again. A double click on its header does the same as the two arrows. Preview (`Ctrl+Enter`) draws what the code plots without saving it; Save (`Ctrl+S`) keeps it under the name you gave it, and from then on it is in **Indicators** and in **In the system**, where you open it again or delete it. Template brings back the example, with the series and the functions you can use.
The browser remembers how you left the panels, and how tall the editor was.

4.3.2 What the strategy drew

A strategy can draw on each asset what it computes, the way an indicator does on TradingView: lines, histograms, levels, coloured backgrounds, arrows, rectangles, labels. It is a few lines in its code (Section 3.3); the backtest records them bar by bar and the View shows them with the trades, for the out-of-sample years. Nothing changes in the results: the drawings are only something to look at.

- **The panes.** What belongs to the price sits on the candles; the rest gets a pane of its own below them, one per name the strategy chose ('RSI', 'Spread'…). Drag the line between two panes to change their height.
- **The legends.** Each pane has its legend at the top left: the name of every line with its value under the cursor, or the last one. A click on a value hides that line, a second click brings it back. The eye next to the strategy's name hides everything it drew.
- **The notes.** If something in the drawing code was wrong — a colour that is not a colour, a style that does not exist — the legend says **1 note**. A click shows what and how many times. The backtest ran all the same: drawing never stops it.
- **On and off. Indicators** → **Strategy** and the **Strategy plots** group of the panel (the gear) turn off the drawings, or only the lines, rectangles and labels.
- **The replay.** During the replay you see only what existed at the bar of the cursor: a stop the strategy removed at the exit is there while the position is open, and goes with it.
- **Other timeframes.** The chart opens on the strategy's timeframe. On another one the drawings follow: a daily value sits on the last candle of its day, a daily background colours all the candles of the day.



Figure 4.20. A note from the drawing code: the strategy asked for a line style that does not exist, and got a plain line.

4.3.3 The trades on the chart

Each trade is an arrow: up and green to buy, down and red to sell, with the side and the quantity (`B 364`). Price and reason are in the legend at the top when the cursor is on the candle. **Trade details on the chart** in the **Positions** group of the panel writes them next to every arrow instead.

4.3.4 The colours of the candles

The panel that opens from the gear at the bottom of the left toolbar starts with **Candles**: four colours, one for each part of a candle.

Colour	What it paints
Buy	The body of the rising candles, those that closed above their open.
Weak buy	Their wicks: the high and the low.
Sell	The body of the falling candles.
Weak sell	Their wicks.

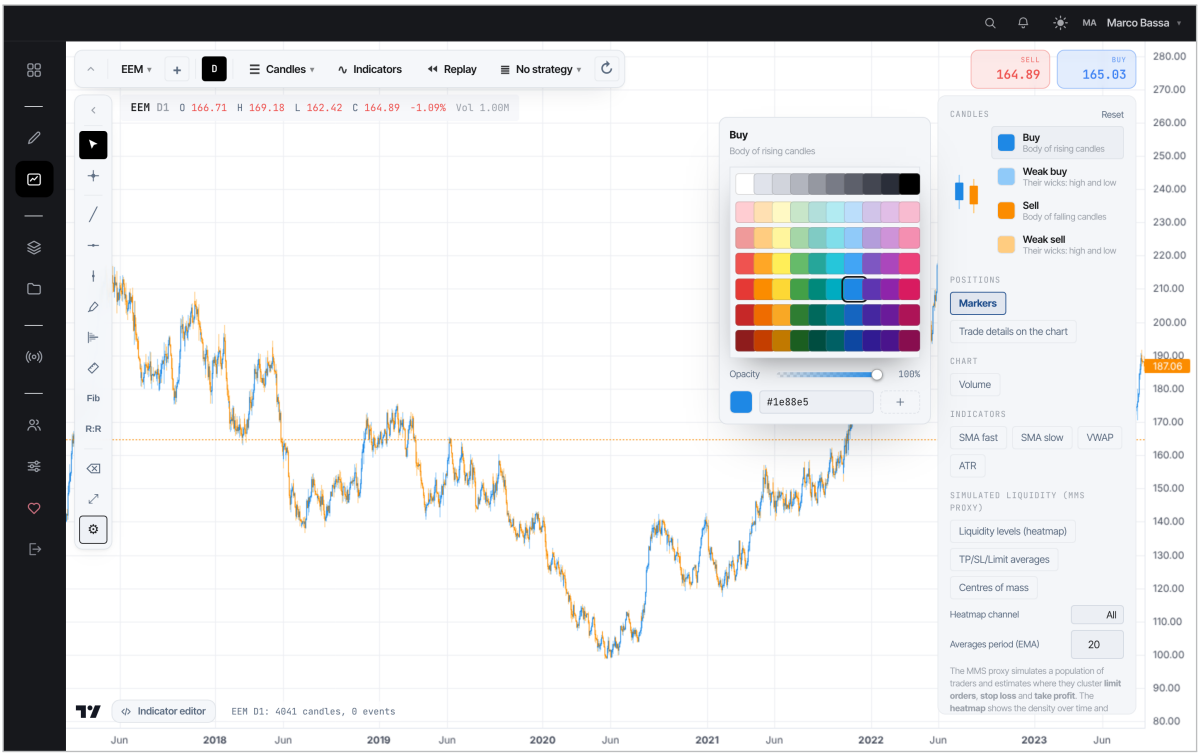


Figure 4.21. Choosing the colour of the rising candles. The chart changes while you choose; the small candles next to the four colours show the result in miniature.

A click on a colour opens the palette: a swatch, the **Opacity**, the hexadecimal code to type (`1e88e5` or the short `f5a` both work), and + for any other colour. The arrow keys move through the palette, `Esc` closes it. The bar charts take the colours of **Buy** and **Sell**, the volume the same two, lighter.

The colours are saved in the application, not in the browser: they stay the same after a restart and in every window. Reset next to **Candles** brings back the default ones.

5

Projects

A **project** is a combination of several strategies, with the rule that decides how much each one weighs and any brakes. It is what you put into production: running a single strategy is rarely a good idea.

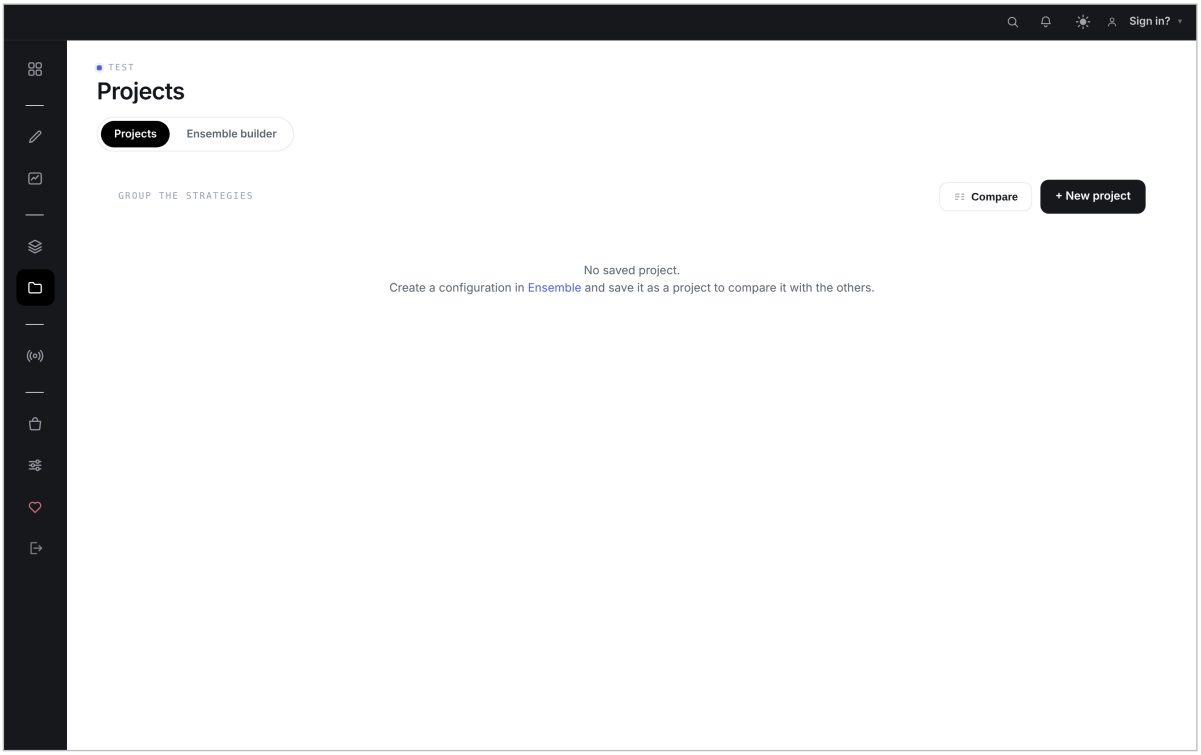


Figure 5.1. The list of saved projects.

5.1 The builder, in four steps

It opens with **Ensemble builder**, or with Use in Ensemble from the page of a stream. The four steps at the top always say where you are.

5.1.1 Step 1 — Strategies

You choose the streams to combine. You need at least two: with one there is nothing to combine, and the Next button stays off until two are chosen.

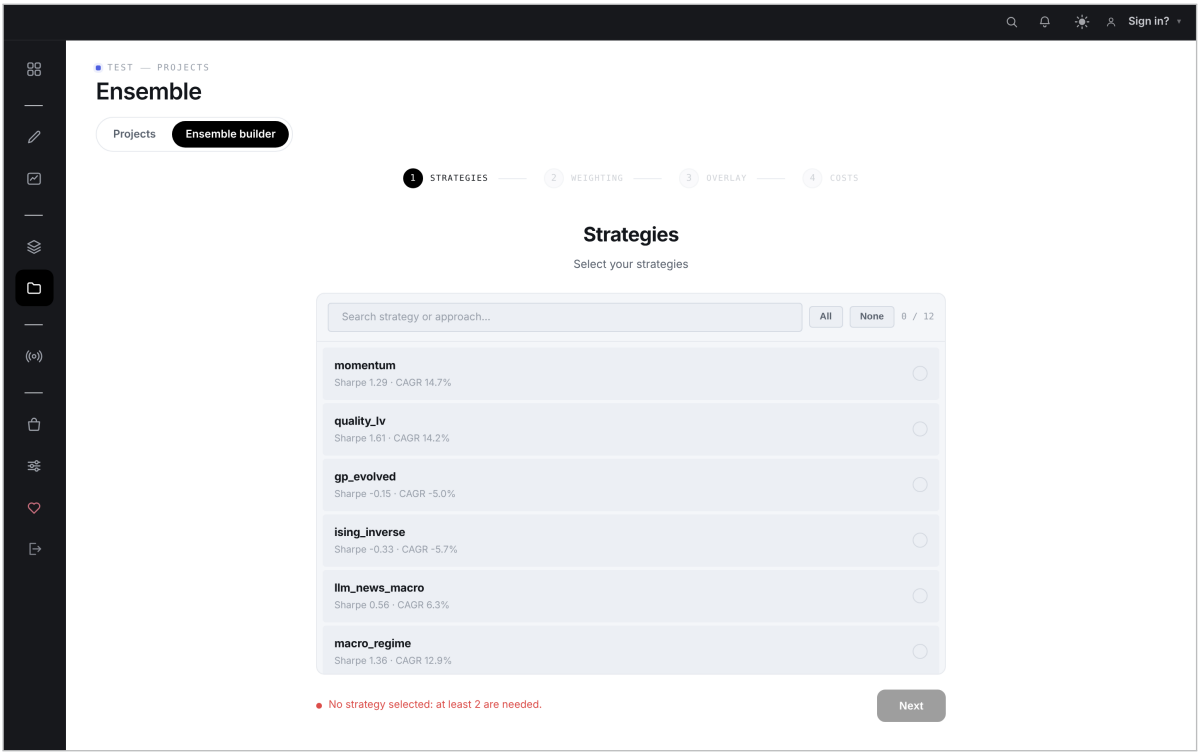


Figure 5.2. Choosing the streams. Next to each one are its Sharpe ratio and return, so you do not have to look them up elsewhere.

Do not pick the best three

The temptation is to take the three highest Sharpe ratios. It is almost always the mistake: if the three strategies make the same bet, putting them together reduces nothing — you are just making it three ways. The home screen helps here: **SPY corr.** in the list shows the strategies that are the market in disguise, and the **Allocation** tab gives the average correlation between the streams of a saved project (Section 2.3). Two strategies with low correlation, even with modest Sharpe ratios, do better together than three good strategies that are all alike.

5.1.2 Step 2 — Weighting

How to share the capital among the chosen streams. Fourteen methods, grouped by family.

Family	The idea
Naive	All the same. **Equal**, an equal share each. Always worth trying: it is the benchmark the others are measured against.
Risk / covariance	Weights by how much each one swings and how much they move together. Whatever swings less weighs more.
Robust	As above, but without trusting the estimated correlations too much — on little data they are unstable.
Tail-risk	Looks at extreme losses instead of the average swing.
Return-aware	Also takes into account how much each one is expected to earn, not only how much it risks.
Regime-adaptive	Changes the weights according to the state of the market.

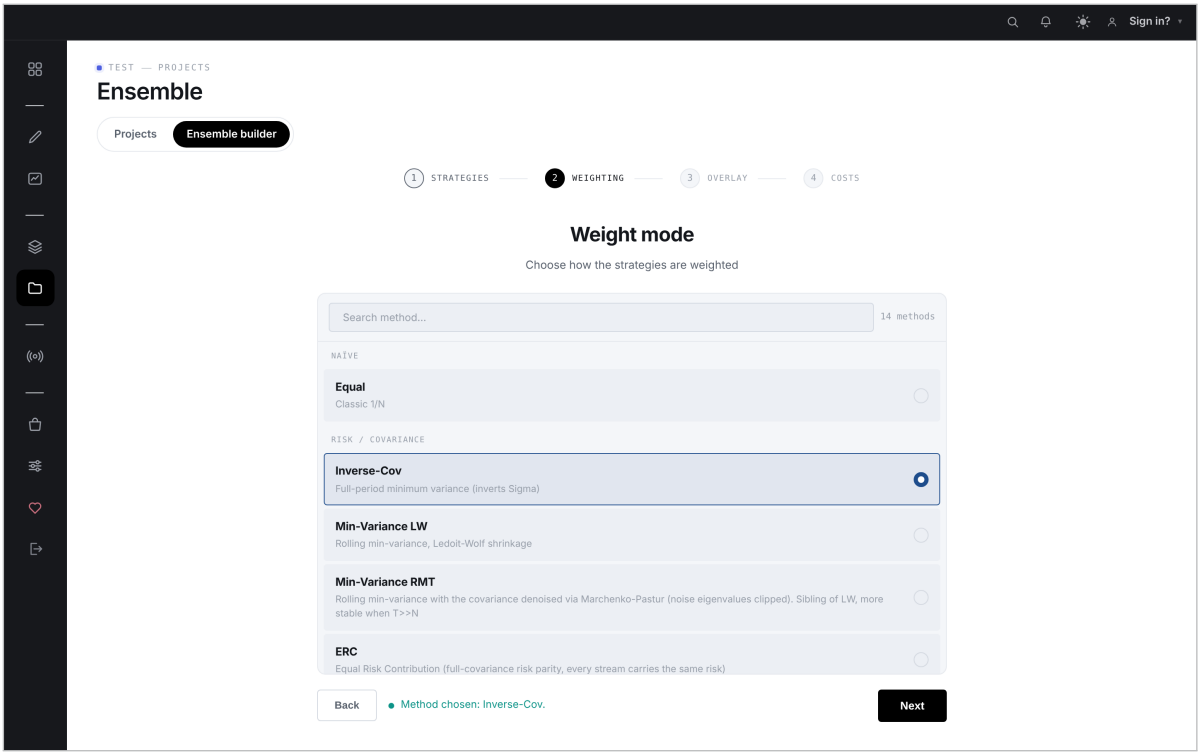


Figure 5.3. The weighting methods, by family. Each one carries its line of explanation.

Where to start

Equal. Then try **Min-Variance LW** or **ERC** and see whether the drawdown really falls. If a complicated method does not beat equal weighting, equal weighting is the answer: it has one parameter fewer to get wrong.

5.1.3 Step 3 — Overlay

Overlays reduce exposure when something happens. Several can be switched on together: their effects multiply, and the order in which you choose them does not change the result.

Overlay	Reduces exposure when...
Vol-Regime	realised volatility rises.
Jump-Vol	there is a sudden jump: it reacts within a day.
Corr-Gate	the strategies start moving all together, that is when diversification disappears.
Regime-GMM, **Iso-Stress**	market conditions look like those of past periods of stress.
Macro-Regime	the macroeconomic signals worsen.
News-Pressure	the volume of news goes above normal.

An overlay also takes away gains

A brake that reduces exposure reduces everything: the losses and the gains. It is worth looking at the two numbers together — how much the drawdown fell, and how much the return fell. Switching on five because 'it can't hurt' leads to a portfolio that is almost always out of the market.

5.1.4 Step 4 — Costs

The costs: commissions and slippage. The backtest subtracts them.

> **Raising the costs is the most honest check you can make**
>
> A strategy that stays good with the costs doubled is robust. One that stops working was earning inside the spread, and on a real account it will not earn.

5.2 The results

After the last step the page shows what the combination would have done. It is the same page that opens when you choose a saved project.

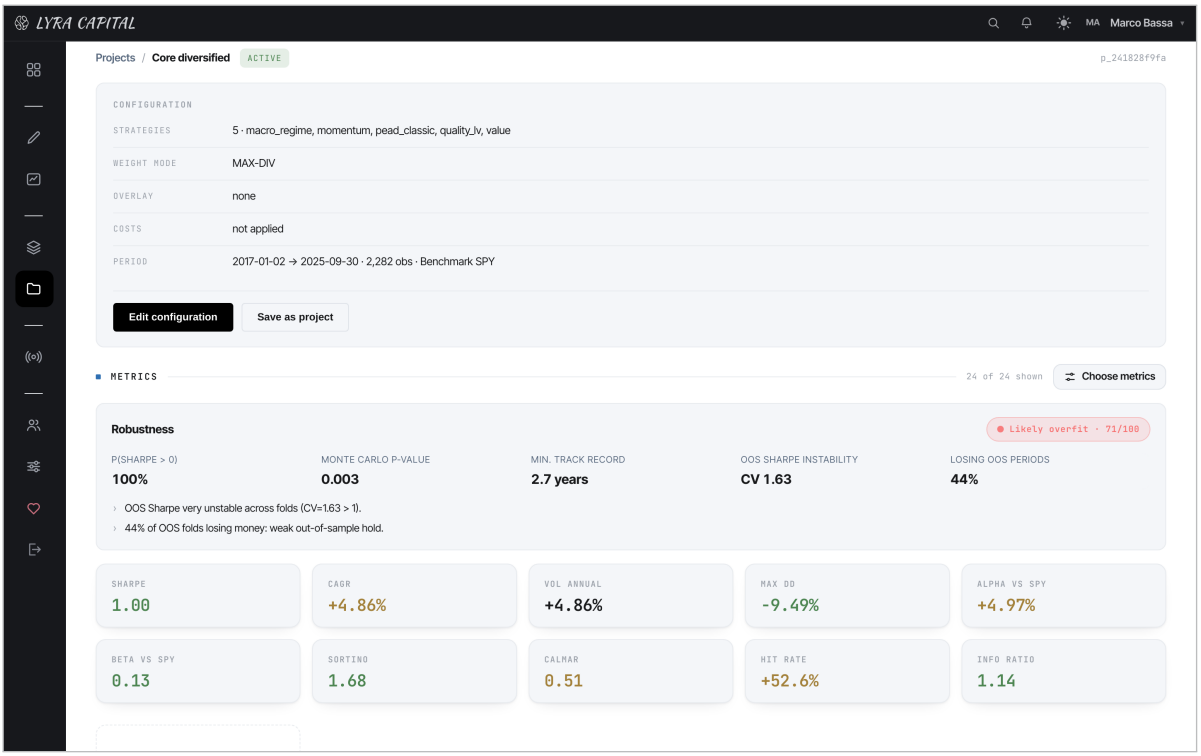


Figure 5.4. The results of a project. At the top the configuration that produced them, then the same parts as the page of a stream, with Choose metrics.

- **Configuration** — the strategies, the weighting, the overlays, the costs and the **Period**: the days the strategies have in common, on which every number below is computed. Edit configuration takes you back to the four steps.
- **Metrics** — the robustness panel, the numbers, the charts and the tables of the page of a stream (Section 4.2), with the same look and the same names: what you learn to read there, you read here. The equity curve has SPY too.
- **What only a combination has** — how much each strategy contributed, how they move together, the weights, which strategies are behind the current drawdown, the factor breakdown of the alpha, the table of the strategies. The highest correlation between two of them over the last 90 days is the one to watch: above 0.8 they are making the same bet.

You choose and arrange them as on the page of a stream (Section 4.2.5, Section 4.2.7): the eye on a metric hides it, Choose metrics lists them all, and charts and tables move and resize like

windows. The choice and the arrangement hold for every project and for the Ensemble builder. They are separate from those of the streams, because here there are panels a stream does not have. By default the charts that go together sit side by side: drawdown and rolling Sharpe, yearly and monthly returns, contributions and correlations, the two attributions. Reset layout brings that back. Export writes the results to Excel or CSV, as for a stream (Section 4.2.8); the file also carries the configuration.

A metric of yours on a project

Your metrics appear here too, computed on the results of the combination and again every time you change it. New number metric and New chart metric, after the numbers and at the bottom of the page, open Studio. Its link back takes you to the project, which is computed again, or to the Ensemble builder with the same configuration, computed again.

5.3 The weights are decided with the past only

A weighting method is re-estimated at fixed dates — every 63 trading days unless the method says otherwise, from the first day with a year of common history — and on each of those dates it sees only the days before. Until the first date the weights are equal. A method estimated once on the whole period would know the future of the very returns it is weighting, and its project would look better than it can ever be in practice.

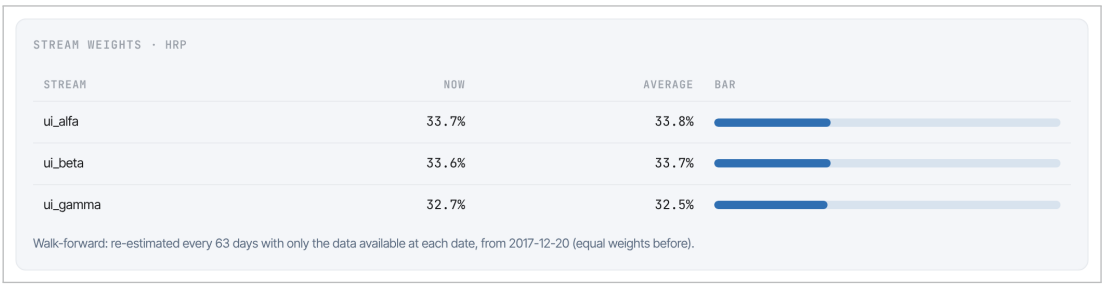


Figure 5.5. The weights of a project. **Now**: the weights in force today, decided at the last re-estimation; **Average**: their mean over the period; below, the calendar of the walk-forward.

- **Now** — the weights decided at the last re-estimation, with the data available then: the ones a runner uses today, to the decimal.
- **Average** — their mean over the period: how the capital was really shared while the results above were made.
- The note under the table says how often the weights were re-estimated and from when, and in how many re-estimations the method gave no valid weights, so that equal weights were used instead.

The overlays pass the same test. Before applying one, Lyra computes it again on shorter histories and checks that its decisions up to each cut do not change: an overlay that looks ahead is not applied, and the page says so in red under the weights — 'Overlay … not applied: …'.

Kelly and Black-Litterman fall

Estimated on the whole period, these two methods gained about 0.13 of Sharpe over the same method re-estimated with the past only — what they knew of the period they were judged on. The methods that look only at risk lose almost nothing. Run again the projects you saved: their numbers are now the ones the method can really reach.

5.4 When strategies overlap

A project weighs its strategies by looking at their curves. Two strategies whose curves did not move together over the year are treated as two independent bets. But that is an average. One trading the euro and one trading the pound, each on its own reasons, have curves that move together little on average — and yet on the days when both are long, they are almost the same bet on the dollar, and the project risks more than it believes.

Day by day, the **overlap correction** compares the risk the project really takes, from the strategies' positions and the correlations between the assets, with the risk its weights assume. When the real one is larger, it scales down the strategies that overlap until the two are equal; the others stay as they are. It never scales up: when two strategies' positions offset each other more than usual, adding leverage would be right only if they were truly independent, and that is exactly what cannot be known on that day. The backtest and the live runner apply it the same way.

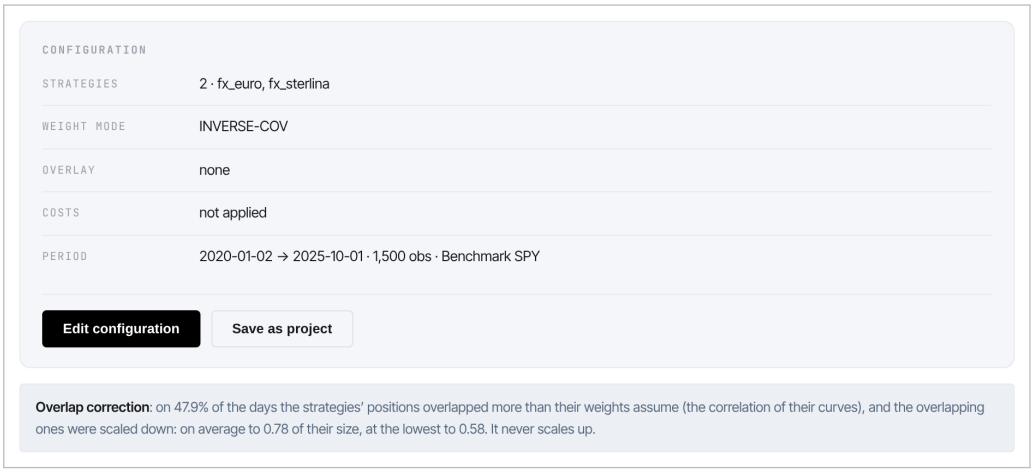


Figure 5.6. Under the configuration, how the correction acted: on how many days the strategies overlapped more than their weights assume, and how far the overlapping ones were scaled down.

When it cannot apply, the note says why: a strategy whose backtest predates this version has no saved positions (run it again), an approach produces only its curve and is left as it is, and without daily prices for an asset the correction is off for that project. You can turn it off in Config (Section 8.2).

5.5 Saving

At the end you give it a name and save it. From then on the project appears in the list, the home screen can draw it — when you choose it, or by itself when it has the highest Sharpe (Section 2.1) — and, most importantly, it can be chosen on the **Live** page (Chapter 6).

5.6 Projects with empty streams

The list of step 1 also has the strategies with an *empty stream* (Section 4.1.4), under their own heading and with a dashed border. They count like the others — a project still needs two strategies — but they have no backtest, so they cannot have numbers:

- **Some empty streams.** The metrics, the charts and the curve on the home screen are those of the streams with results, and the home screen says how many strategies are left out. The configuration at the top lists them under **Not in the metrics**.

- **Only empty streams.** There is nothing to measure: instead of the metrics, the page says why. The project is saved all the same, and on **Projects** its card reads **No metrics** · **empty streams**.

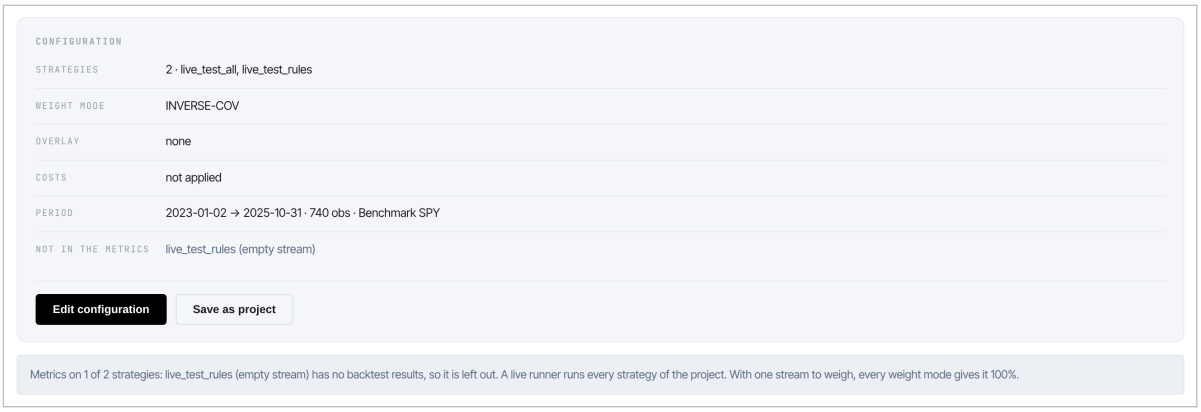


Figure 5.7. A project with one stream with results and one empty stream: the metrics are those of the first, and the note says so.

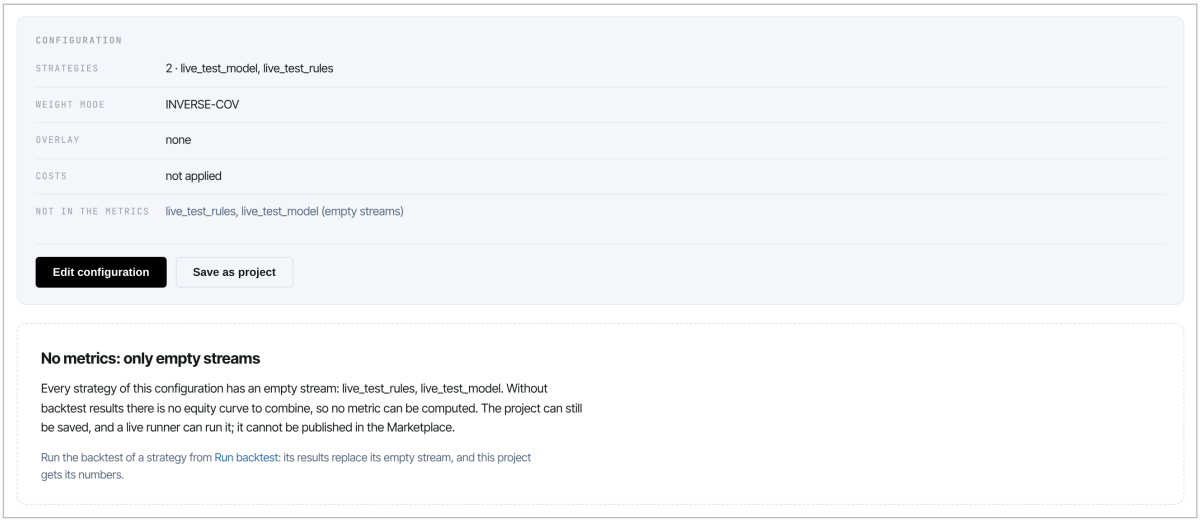


Figure 5.8. A project made only of empty streams: no metrics, and it can be saved and run live.

In **Live** these projects run like any other: the runner runs every strategy of the project, with or without a backtest (Section 6.1). As the backtests arrive, the empty streams fill up and the project gets its numbers, with nothing to redo. A project that contains an empty stream cannot be published in the Marketplace.

6

Live

This is the page where a strategy stops being an exercise. A runner always starts in simulation: it works out the orders on the connected account and writes them to its journal, without sending them. Real orders are switched on runner by runner, when you decide, with an explicit confirmation. From then on, money moves.

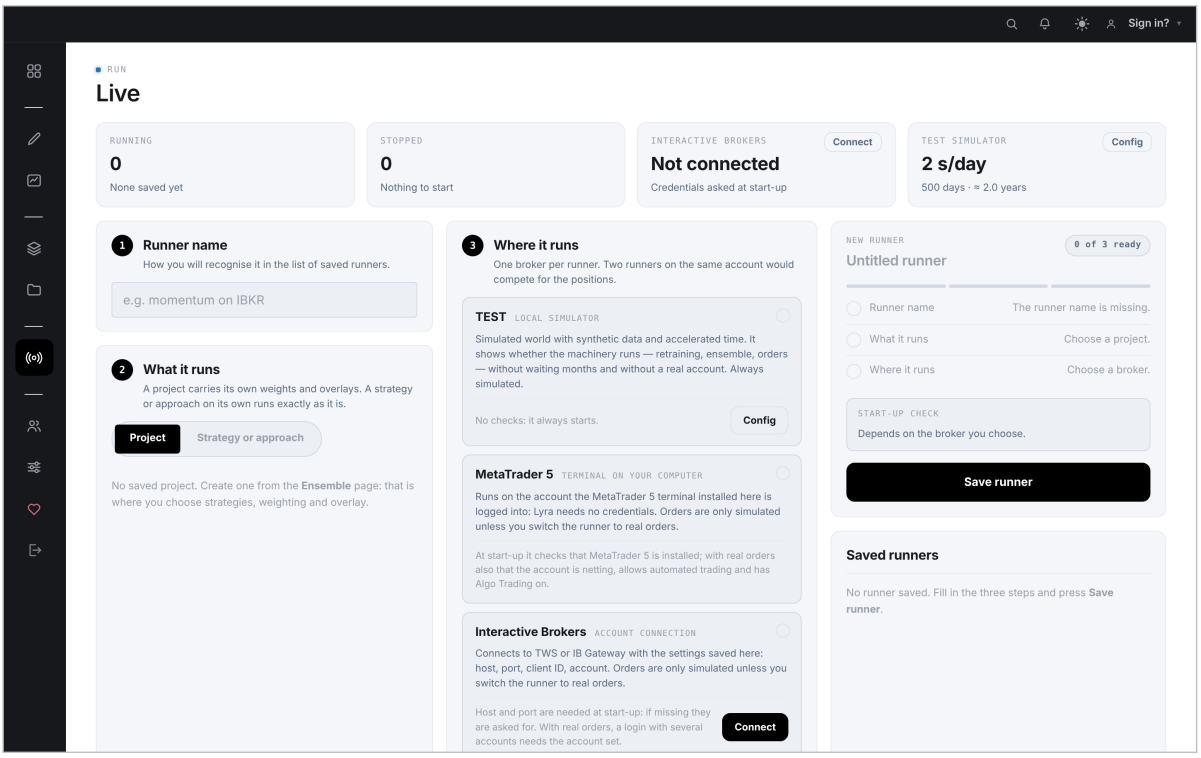


Figure 6.1. The Live page. At the top the status, below it the three steps to create a runner, on the right the summary of what is missing and the saved runners.

6.1 What a runner is

A **runner** is what runs a strategy on an account. It is made of three parts, and they are exactly what the three steps ask for:

1. **A name.** It only serves to recognise it in the list.
2. **What it runs.** A project (with its weights and its overlays), or a single strategy.
3. **Where it runs.** Which broker. For MetaTrader 5 and Interactive Brokers there is also the choice of orders: simulated or real (Section 6.4).

The panel on the right shows which of the three are in place: the **Save runner** button stays off until all three are. Below it, a line says what the orders will be like, already in the colour they

will have on the runner's card.

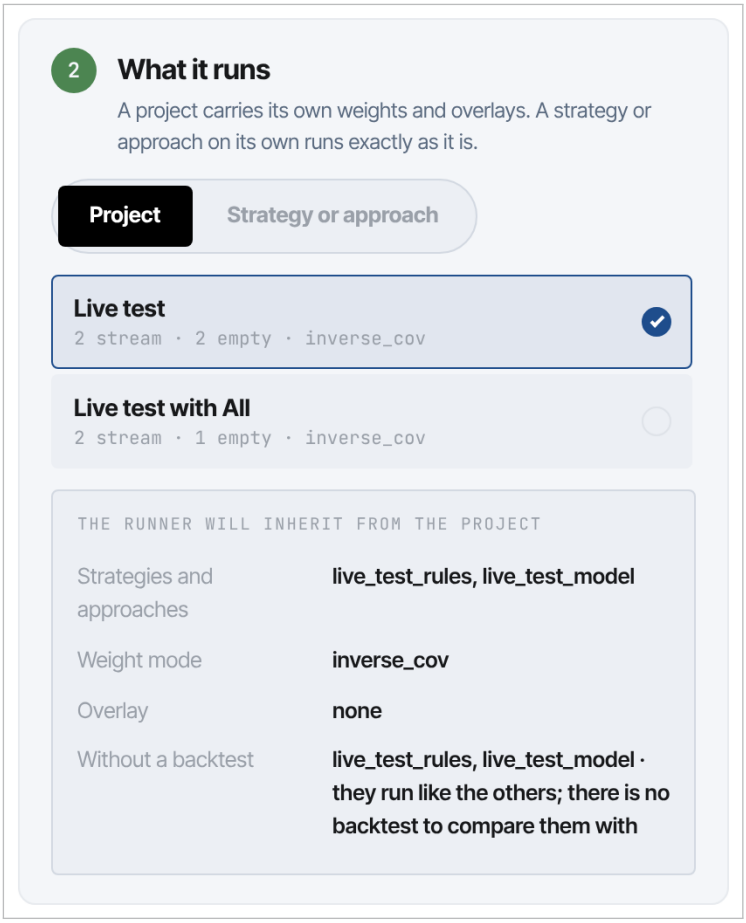


Figure 6.2. A project made of two empty streams. The list says how many streams of each project are empty, and the summary names them: they run like the others, with no backtest to compare them with.

A project can contain strategies with an *empty stream*, or be made only of them (Section 5.6): this is the way to run in **Live**, together, strategies that have no backtest yet. The runner runs their code as it runs any other; what is missing is only the backtest to compare the live with. Names too long for their card end with '…': the full name is in the tooltip.

6.2 The three brokers

Broker	What it does
TEST	A simulated world with synthetic data and accelerated time. It needs no account and no credentials. It shows whether the machinery works — retraining, composition, orders — without waiting months. It is always simulated, but it runs the same engine as a real runner: the project's streams with their parameters and data, as in the backtest.
MetaTrader 5	Runs on the account the MT5 terminal installed on the computer is logged into: Lyra needs no credentials. At start-up it checks that MT5 is there; with real orders, also that the account is *netting*, that the broker allows automated trading and that **Algo Trading** is on. The names of the symbols on the broker are found by themselves; the few that are not go in Symbols (Section 6.9.1).
Interactive Brokers	Connects to TWS or IB Gateway with the host, port, client ID and account saved with Connect. If they are missing, it asks for them at start-up.

The order to follow, without exceptions

1. **TEST** for a few simulated days. It shows that the machinery starts, that orders come out, that the journals get written.
2. **The real broker, in simulation.** The runner reads the account's equity and positions and writes to the journal the orders it would send. You find out which instruments the broker does not trade and which quantities do not add up.
3. **A demo or paper account, with real orders on.** Real orders, play money: you find out about trading hours, rejections and missing permissions.
4. **A real account**, with a fraction of the capital.

Skipping a step saves no time: it only means finding the same problems later, with the money in.

6.3 Where the prices come from

On MetaTrader 5 and Interactive Brokers the runner gets its bars by itself. Every part of what it runs declares the history it needs: a strategy, the bars its indicators look back over and the stretch it replays to rebuild its state; the regime, the series that drives it in each market (SPY for shares, the dollar index for currencies, Bitcoin for crypto), with its whole history; the **IsoForest-Stress** overlay, the market stress measured on the stock panel.

1. The runner asks the **broker** first: the first time all the bars, then only the new ones.
2. What the broker does not have — a symbol it does not quote, history it does not go back to — comes from the **data of the backtest** and from the **providers**: Yahoo Finance, Dukascopy and Polygon (Polygon with `POLYGON_API_KEY` in the `.env` of the application). Each is asked only for what it can give: Yahoo has hourly bars for two years and minute bars for a few weeks, Dukascopy only currencies, Polygon only the markets of your plan.

3. Before an older stretch is joined to the broker's bars, the runner checks that the two move together on the days they share; a broker's CFD that quotes at another level is brought to the same level. Only closed bars are used.

The first start can take a few minutes: the runner's card says how far it has got ('Context: 182 of 254 series'), and the first cycle starts when the bars are in. If a series stays short even so, the runner uses what there is and the journal says which series and why — for instance a Polygon key that was rejected. You do not need to update prices in Settings → Data for a runner on a real broker.

What the runner downloads does not touch your backtests

The bars, the category rows and the series the runner downloads or computes go into `data/live`, not into the data your backtests read; at the next start it picks up from there. You merge them into the backtest data, checking for the days the application was off, or delete them, from Settings → Data → Live data (Section 8.1.5). If you choose not to keep them, they go to a temporary folder that, while the application stays open, is cleaned of what the runners no longer need.

6.3.1 Each strategy runs as in its backtest

A strategy does not just send signals: in the backtest it has its own portfolio, risk limits and exits, and the runner gives it the same. At start-up its state is rebuilt by replaying the backtest engine itself over the last year of bars, then it moves one closed bar at a time, deciding at the close as in the backtest. The positions it holds after a restart are the ones it would hold had it never stopped.

- **Stops and take-profits** are checked every minute on the broker's prices, not only at the cycles. A position opened at a bar is watched from the next one, as in the backtest.
- **The regime** is worked out live on its drivers, which the runner keeps up to date.
- **Small adjustments** — a position that would change by less than 0.2 % of the account — are not sent; openings, closings and reversals always are.
- A strategy whose data are missing (the macro series of **macro_regime**, the earnings of **pead_classic**) says so once in the journal and waits; it starts when the data arrive.

6.4 Simulation or real orders

Once MetaTrader 5 or Interactive Brokers is chosen, the **Orders** panel appears below the broker cards, with two positions: **Simulation**, the starting one, and **Real orders**.

Switching real orders on

1. In step 3, choose **MetaTrader 5** or **Interactive Brokers**.
2. In the **Orders** panel, press **Real orders**. A window opens that says what changes.
3. Read it, tick the box and press **Enable real orders**: the button stays off until the box is ticked.
4. **Save runner**. The **REAL ORDERS** label appears in red on the runner's card.

For a runner that is already saved, use **Switch to real orders...** on its card, with the runner stopped: the window is the same.

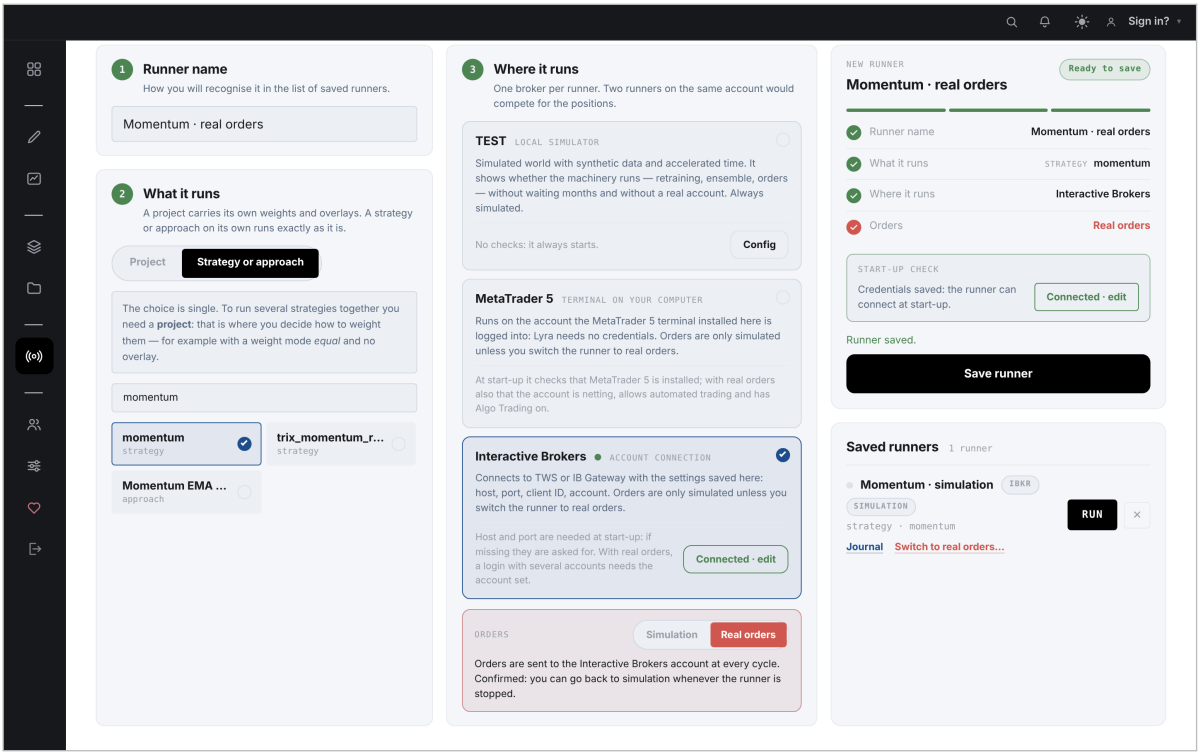


Figure 6.3. A runner with real orders on: the **Orders** panel and the summary line turn red. On the right, a runner already saved in simulation.

The runner manages the whole account

It sizes the orders on the equity of the whole account, not on a part of it, and in the symbols it trades it closes the positions that are not among its targets: a share bought by hand on that account, if it is in the strategy's universe, can be sold at the next cycle. That is why the account must be dedicated to Lyra — one per runner, no manual trading.

Currency pairs with the dollar first, and crosses

An order on a currency pair is a number of units of the first currency. The runner sizes it in dollars, and converts with the price of the second currency in dollars: on EURUSD a unit is worth the price, on USDJPY a dollar, on EURGBP the price times GBPUSD. Before, every pair was treated like EURUSD, and USDJPY received about one hundred-and-fiftieth of the exposure. If the rate it needs is not available, the order does not go out and the journal says why.

First reduce, then increase

In every cycle the runner first sends the orders that reduce positions, then those that increase them; a reversal (from short to long, or the other way round) becomes two orders, the close and the open. If part of the book cannot go out — old data on a stock, an order still open at the broker, a holding with no price — or a reduction does not go through, the orders that increase exposure wait for the next cycle. A cycle left halfway never leaves the account more exposed than before.

The mode is always visible: on the runner's card (**SIMULATION** or **REAL ORDERS**) and on

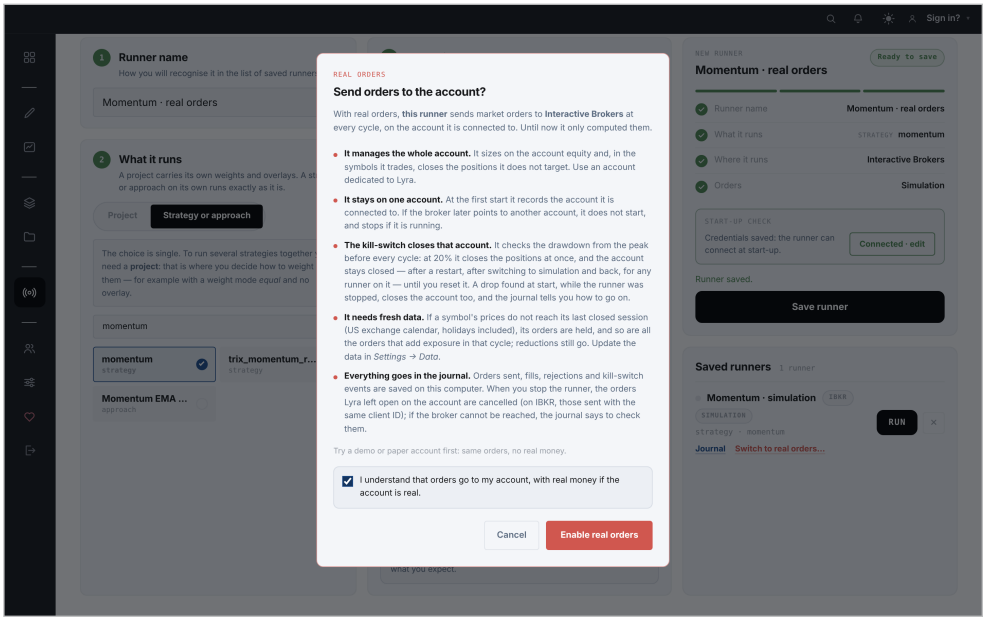


Figure 6.4. The confirmation of real orders. The button turns on only after the box is ticked.

every line of the journal. It can only be changed with the runner stopped, and going back to simulation with **Back to simulation** asks for no confirmation. The **RUN** button of a runner with real orders is red and asks once more before starting.

The confirmation stays on record

The runner records when real orders were confirmed, and the journal reports it. Going back to simulation deletes the confirmation: to switch them on again, you confirm again.

6.4.1 The confirmation is for one account

At the first start after the confirmation, the runner records the account it is connected to — on MT5 the number and server, on IBKR the number — and from then on it does not move. If later the MT5 terminal is logged into another account, or the **Connect** settings point to another IBKR account, the runner does not start and the journal says which account it expected and which one it found. If it happens while it is running, it stops before sending anything else. The account is shown on the runner's card and in the start window.

Moving to another account

1. With the runner stopped, **Back to simulation**.
2. Connect the broker to the new account (the MT5 terminal, or **Connect** for IBKR).
3. **Switch to real orders...** and confirm: at the next start the runner binds itself to the new account.

Accounts in another currency

Lyra sizes the orders in dollars, on the prices of its own data. An account in euros is fine: the equity is converted with the exchange rate the broker gives (on MT5, the EURUSD price in the terminal). If there is no rate, real orders do not start and the journal says so.

6.5 The journal

Every runner keeps a journal, which opens with **Journal** on its card. Everything the runner does goes into it, one line per event, with the time and the mode in which it happened:

- starts and stops, with the account it connected to: number, type (demo, paper or real), terminal; with real orders, also the check that it is the confirmed account and Lyra's orders left open, cancelled at start and at stop;
- the orders. In simulation, those it would send; if they are the same at the next cycle, a single line says so. With real orders, those sent and their outcome: filled with quantity and price, partly filled, still working (*pending*), cancelled, rejected with the reason the broker gives;
- warnings, for example a symbol the broker does not trade, or data that are too old;
- the kill-switch events (Section 6.6);
- maintenance: retraining and data updates. A strategy that retrains its own model, inside its code, is said to do so once; then each refit appears when it happens, written by the strategy itself ('model refit at 2026-11-17 08:00 UTC on 5344 bars…');
- what the strategies report while they run: a refit, a datum that is missing, a filter that stopped them trading;
- the data and the regime: which symbols the broker has, where each series came from, which are short and why, each strategy rebuilt, the regime of each market.

At the start, the maintenance lines say which data the runner keeps up to date: one line per category, with the components that read it, the source, how often — and, if something is missing, what: 'data · macro (macro_regime): FRED and Yahoo · macro series, every day — needs FRED_API_KEY in the .env of the app'. The updates then run in the background, without holding up the cycles; when one brings new rows, the journal says how many and the runner decides again straight away with them. The same error is written once, not at every attempt. The cards on Settings → Data show the same state (Section 8.1.4), and the new rows wait in `data/live` (Section 8.1.5).
The filters at the top separate orders and fills, kill-switch, warnings and errors, data and regime. While the runner is running, the journal updates by itself.

> **It lives on the computer**
>
> The journal is a file in the application folder, `data/runners/<id>/journal.jsonl`, one line per event; beyond 5 MB it continues in `journal.1.jsonl` and so on, up to ten files. It stays after the application is closed and even after the runner is deleted: it is the account's logbook, and it is worth keeping.

6.6 The kill-switch

Three levels, on by default, the same as in the backtest:

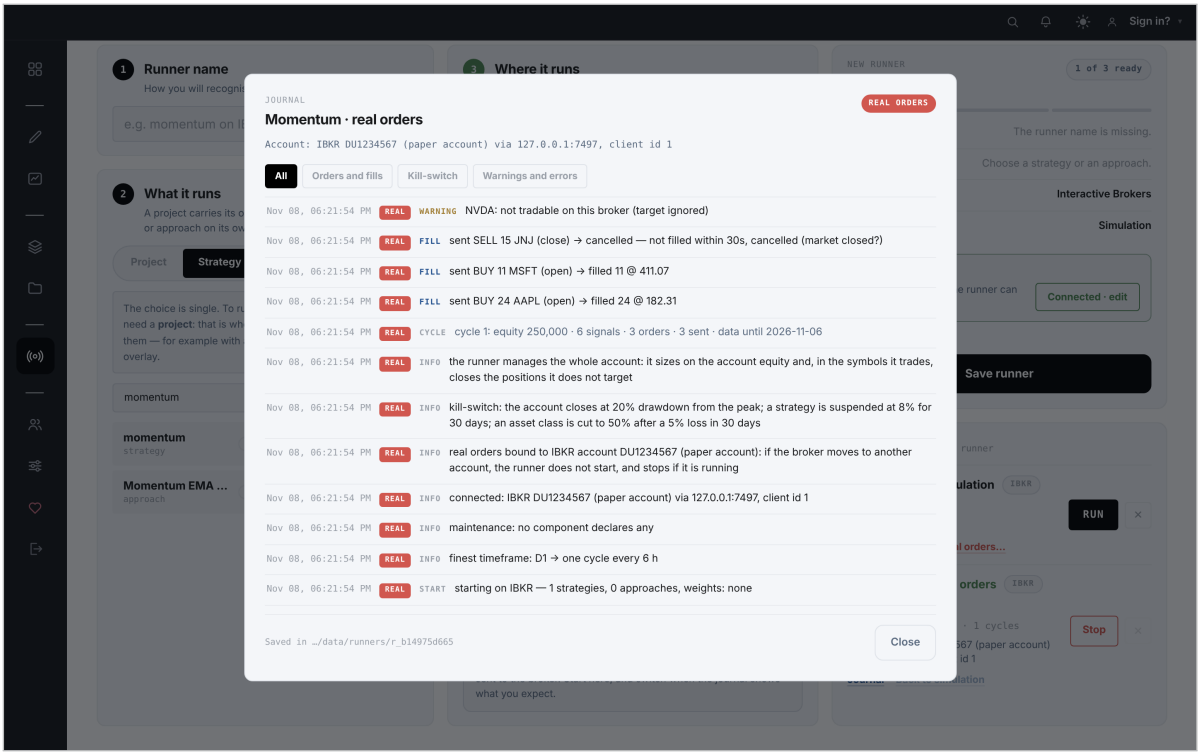


Figure 6.5. The journal of a runner with real orders: two orders filled, one cancelled because it was not filled in time, a warning.

Level	When it trips	What it does
Strategy	An 8% drawdown	Suspends it for 30 days, then it comes back by itself.
Asset class	A 5% loss in 30 days	Halves its exposure for 20 days.
Account	A 20% drawdown from the peak	Closes the positions in the runner's symbols and opens no others.

A suspended strategy and a halved asset class stay that way even after the runner is restarted, until they expire.

The account level, with real orders, belongs to the **account** and not to the runner. The drawdown is measured at the start of every cycle, before deciding: if the fall happened between one cycle and the next, the runner does not send another round of purchases, but closes the positions in the symbols it trades straight away. If something stays open (market closed, a rejection) it tries again at once, then every minute three times, then at the normal pace; once the positions are closed, it stops by itself.

From then on the account stays closed: after a restart, after going to simulation and back to real orders, for any runner with real orders on that account. Starting a runner on a closed account only serves to close the positions left: the start window says so, the runner closes and stops. While it is closing, not even repeated errors stop it.

Reopening the account

1. Stop the runner.
2. **Journal** → (Reset kill-switch), and confirm. The account reopens for every runner that uses it.
3. At the next start the peak starts again from the account's equity.

Why it belongs to the account, and is saved

The kill-switch protects money, and the money is in the account. When it belonged to the runner, a restart was enough to start again from the equity of the moment — after a 15% drawdown the 20% limit in fact became 32% — and going to simulation and back, or using another runner, was enough to reopen a closed account. The state lives in `data/accounts/`, one file per account; if it cannot be read, real orders do not start: a closed account must not reopen because of a damaged file.

A fall found at start-up, withdrawals and deposits

The kill-switch looks at the account's equity and does not tell a loss from a withdrawal. If at start-up the equity is already below the account's peak by more than the threshold — the fall happened with the runner stopped — the kill-switch trips and is recorded, but the runner sends no orders and says how to go on. If it is a loss, start the runner again: it closes the positions left and stops. If money was withdrawn, reset the kill-switch from the journal instead. If the account's equity cannot be read at start-up, real orders do not start.

In simulation

The kill-switch trips in simulation too, and the journal says what it would have done. There it closes nothing and does not touch the account: it belongs to the runner, and it is reset the same way.

6.7 When orders do not go out

When an order does not go out, or does not go as expected, the journal says so in one line. The most common ones, what they mean and what to do:

In the journal	What it means, and what to do
'orders NOT sent: the data are too old'	Some symbols did not reach the last closed session of their market: for US shares and ETFs, the New York Stock Exchange calendar, holidays included; for forex and crypto, one session is tolerated. On MetaTrader 5 and Interactive Brokers the runner keeps asking the broker and the providers for the missing bars, and the journal says why they are not in (a provider that does not answer, a key that was rejected); with the other brokers the prices are those of Settings → Data. The orders on the symbols that are behind wait, and so do those that increase exposure; reductions on the other symbols go out. The held orders go out at the first cycle with fresh data. The closes ordered by the account's kill-switch go out regardless.
'N symbols not tradable on this broker'	The strategy wants a position in symbols the broker does not offer: their targets are ignored, the rest of the book goes out. On MetaTrader 5, check the start-up line 'not found on this broker' (Section 6.9.1).

In the journal	What it means, and what to do
'real orders cannot start on this account'	The account cannot receive orders from Lyra, and the line says why: on MT5, a *hedging* account (a *netting* account is needed), trading disabled or not allowed to expert advisors, **Algo Trading** off, the Python API switched off in the terminal's options; on IBKR, a login with several accounts and no account given in **Connect**; on both, an account in another currency with no rate to convert it.
'the real orders of this runner were confirmed for . . . '	The broker is connected to an account different from the one the runner started on after the confirmation. If it is the right account, go back to simulation and switch real orders on again (Section 6.4.1).
'cycle skipped, nothing sent'	An obstacle that appeared while the runner was on, for example **Algo Trading** switched off or the account data not yet in sync. The runner skips the cycles without sending anything and resumes by itself when it goes away.
'kill-switch: this account was closed'	The account was closed by the kill-switch: the runner starts only to close the positions left, then stops (Section 6.6).
'an order is still working at the broker'	On that symbol there is an order still open — Lyra's or one placed by hand. While it stays open, the runner does not send another: when the first was filled, the position would double.
'order(s) that add exposure not sent'	Part of the book did not go out (one of the lines above, or a holding with no price in that cycle) or a sale did not go through: in that cycle only the reductions go out, and the purchases are worked out again at the next cycle.
An order *rejected*	The reason is the broker's, written in the line: permissions, market closed, quantity below the minimum. In TWS and IB Gateway the **Read-Only API** option must be off, otherwise every order is rejected.
An order *cancelled* on IBKR	It was not filled within 30 seconds, usually because the market was closed, and the runner cancelled it: left open, it would have been filled later while the runner was working out another one.
An order *pending*	The broker accepted it but has not filled it yet (MT5), or its cancellation is not yet confirmed (IBKR): it can still be filled. The runner sends no other orders on that symbol while it stays open, and cancels it when it stops.
An order *skipped*	The quantity is below the broker's minimum lot, and nothing went out: the difference is smaller than one lot.

In the journal	What it means, and what to do
‘the broker is not answering’	Connection dropped, TWS or MetaTrader 5 closed or restarting. The runner sends nothing and keeps trying without giving up, with waits that grow up to a quarter of an hour; when the broker answers again, it says so.
‘cycle failed (1 of 3 in a row)’	An error of another kind. The runner tries again within a minute; three errors in a row stop it, with the cause in the journal — except when it is closing an account closed by the kill-switch.
‘cannot connect to TWS/IB Gateway’	TWS or IB Gateway off or not logged in, API not enabled (Configure → API → Settings), wrong port, or another program is using the same client ID.

6.8 The simulator

The **TEST SIMULATOR** panel at the top says how long a simulated day lasts — in the screenshot, two seconds — and how much time it will cover in all. Pressing Config changes the pace.

> **What speeding up time is for**
>
> A strategy that retrains every thirty days, on a real account, retrains for the first time in thirty days. In the simulator it happens in a minute, and if the retraining has a problem you find out at once instead of next month.

6.9 Credentials

The broker credentials stay **on the computer of whoever uses the application**, in a local file. They do not go through the site, they do not end up in any database and they do not leave the machine.

For Interactive Brokers, Connect saves the host, the port, the client ID and the account. Port and client ID must be numbers, and the window says so at once. The usual ports are 7 497 for TWS in paper and 7 496 for real, 4 002 and 4 001 for IB Gateway. The account must be given if the login manages more than one.

> **Two consequences**
>
> The first: nobody can recover them for you: if they are lost, you set them up again with the broker. The second, more important: whoever has access to that computer has access to the account. The application is meant to run on a personal machine, not on a server others can reach.

6.9.1 Symbol names on MetaTrader 5

Brokers name the same instrument in different ways: `AAPL.US`, `#AAPL`, `EURUSD.r`, `BTCUSD`. Lyra finds them by itself, with the broker’s suffixes and prefixes, and at every start the journal says how many symbols of the runner it found and which it did not (‘not found on this broker: …’). A symbol that is not found gets its bars from the providers but cannot be traded.
If the broker does have it, under a name Lyra does not recognise, press Symbols on the

MetaTrader 5 card and write one line per symbol, Lyra's name on the left and the broker's on the right:

```
AAPL = Apple.US      BRK-B = BRK.B
```

A wrong line is stopped when you save it. The names stay on this computer, like the credentials, and count from the next start of an MT5 runner; the button shows how many there are (**Symbols · 2**).

6.10 Starting and stopping

Once saved, the runner appears in the list at the bottom right, with the commands to start and stop it. The two counters at the top — **RUNNING** and **STOPPED** — say how many are in each state.

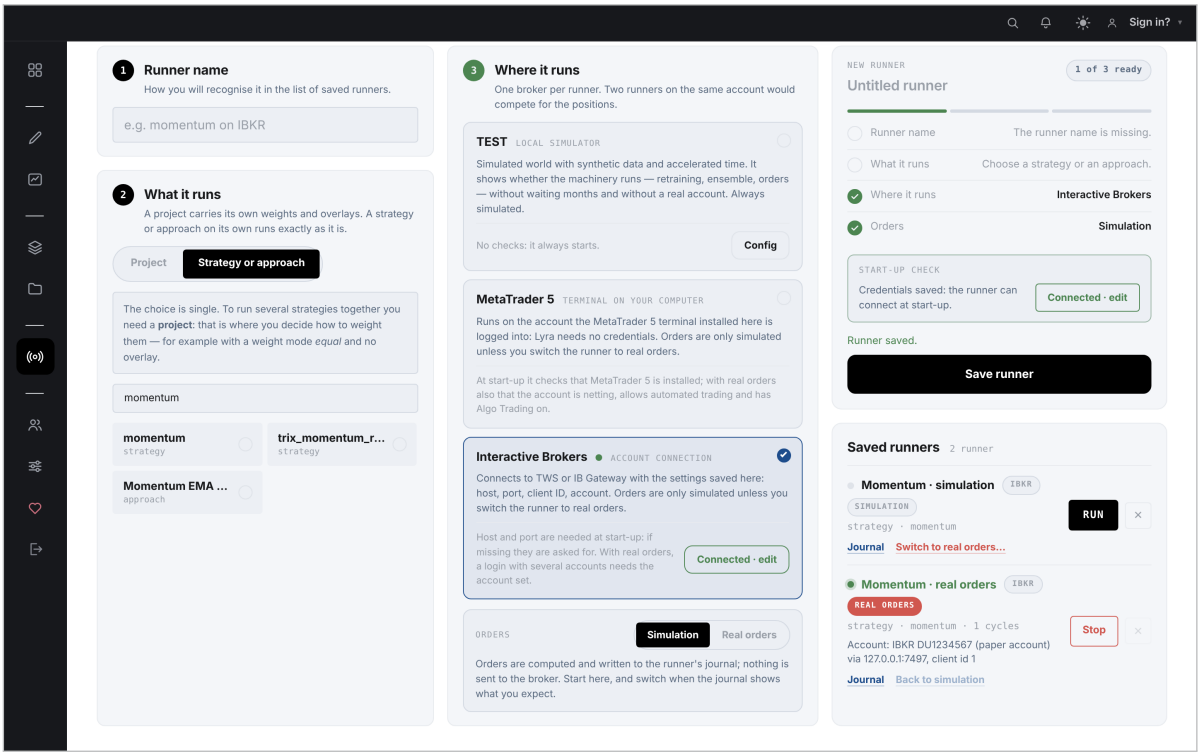


Figure 6.6. Two runners on the same model: one stopped in simulation, one running with real orders.

The first cycle starts at once. The following ones follow the shortest bar among those the runner watches, a quarter of a bar at a time: for daily strategies, one cycle every six hours. Asking the broker more often would produce identical cycles.
With real orders, **RUN** asks for confirmation once more and says which account the orders will go to; if the account was closed by the kill-switch, it says that starting will only serve to close positions.

Stopping: no new orders

Once **Stop** is pressed, the orders of the current cycle not yet sent do not go out any more; those already sent reach their outcome, which goes into the journal. With real orders, Lyra's orders still open at the broker are cancelled: none of them is filled later without the runner watching.

One broker per runner

Two runners on the same account fight over the positions: the first opens, the second sees a position it does not expect and closes it. The page says so in step 3, and does not let a second runner start on the same broker while the first is running.

7
Community

It is the area where users meet, and the only one in the application that needs an account. In the menu it sits under **Share**; the page has four tabs:

Tab	What it is for
Marketplace	Buying and selling strategies, approaches, overlays and weighting methods. It is the subject of this chapter.
Messages	Conversations with other users, for example with the author of a strategy.
Feedback	The reports and suggestions sent to the Lyra team.
My account	Profile, security and subscription (Chapter 9).

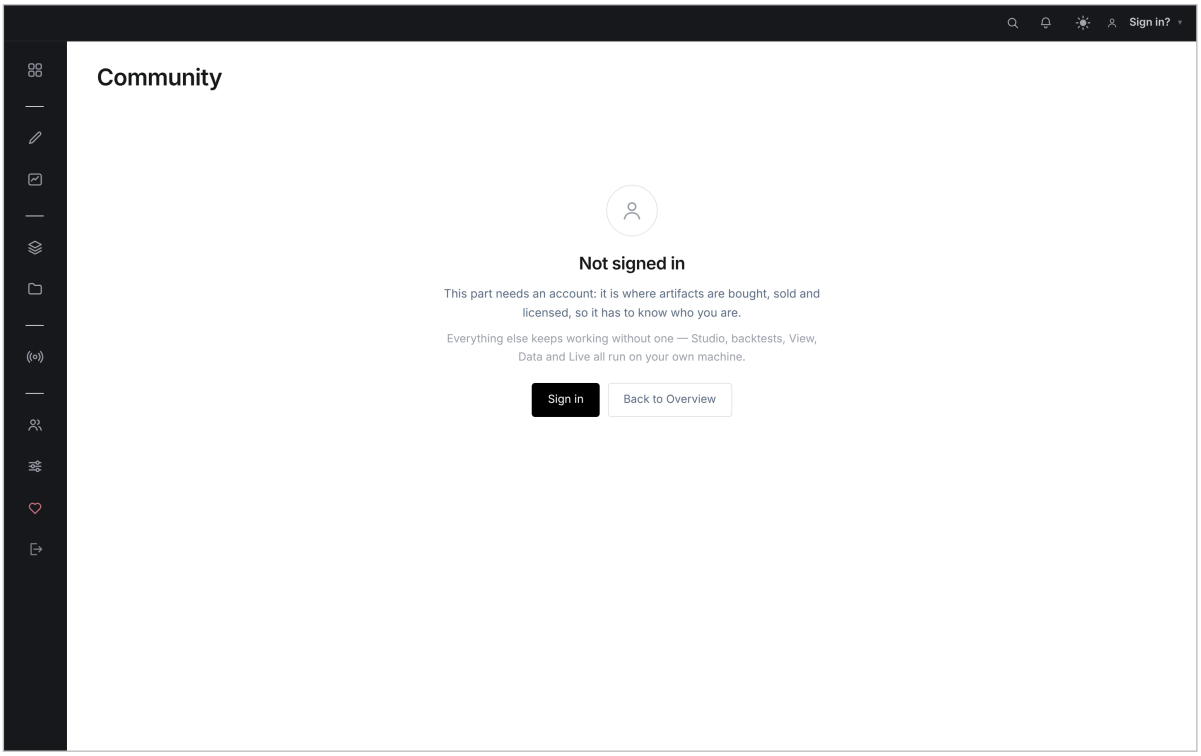


Figure 7.1. The **Community** without an account. The rest of the application keeps working: the page says so.

7.1 Why an account is needed here and not elsewhere

Because here there is someone on the other side. Building and testing strategies happens entirely on your own computer, and asking for an account to do it would make no sense. Buying, selling

and granting a licence require the system to know who is who.

7.2 Browsing the storefront

Every card starts with what the artifact did out of sample: its **equity curve**, green if it ended above where it started, red if below. The dashed line is the starting level. Building blocks that have no backtest of their own — overlays, weight modes, metrics, indicators, regime rules — show the icon of their type instead.

Under the curve come the name, the author and the version, the author's summary, and three numbers measured out of sample: **Sharpe**, **CAGR** and **max drawdown**. A blue badge after the author's name marks a **verified seller**: the payment provider has checked their identity and pays them. The author cannot set it; it comes from the payment account. At the bottom, how long ago it was published, likes, downloads and rating. A line in the warning colour says when the artifact needs auxiliary data (Section 7.4.1).

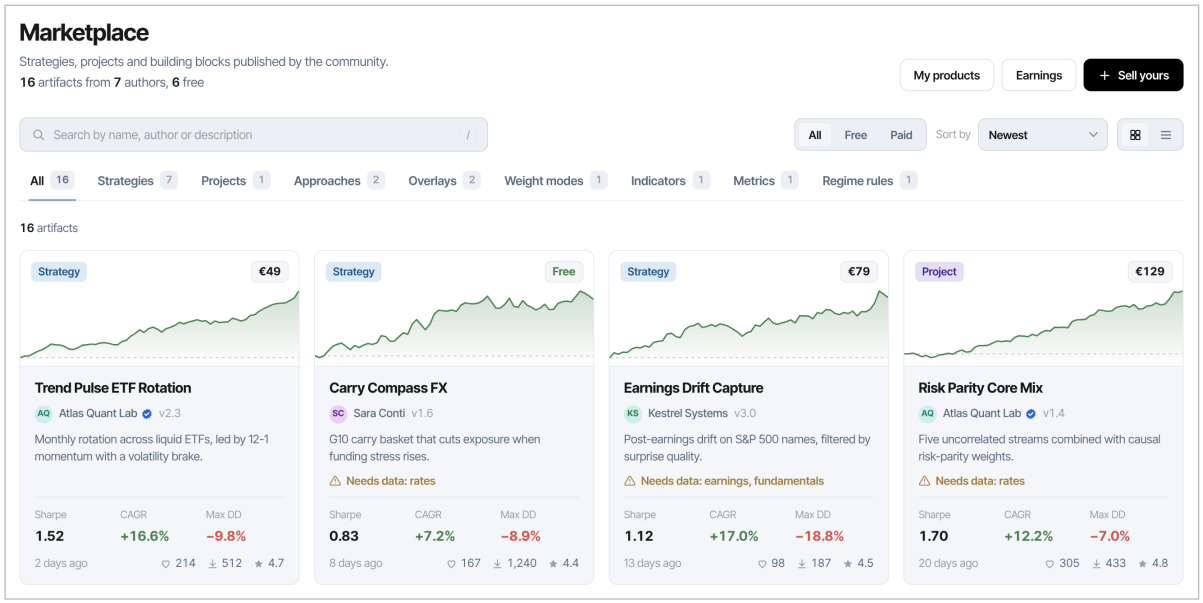


Figure 7.2. The storefront. The bar stays in view while you scroll; the type tabs say how many artifacts of each type match the search and the price.

The bar above the storefront stays in view while you scroll:

- **Search** — by name, author or description; the list narrows while you type. Press **/** anywhere on the page to jump to it.
- **Price** — **All**, **Free** or **Paid**.
- **Sort by** — **Newest**, **Highest Sharpe**, **Most liked**, **Most downloaded**, **Top rated**, **Lowest price**. Artifacts without a backtest go to the end of **Highest Sharpe**.
- **View** — cards or a table. The choice is remembered.
- **Types** — the tabs below, each with how many artifacts it holds for the current search and price.

The storefront remembers your choices. **Back** (Section 1.3) returns to the previous selection, and from an artifact's page it brings you back to the storefront as you left it: same filters, same sort, same view, same position on the page.

The **table** shows the same artifacts in columns: a small equity curve, Sharpe, CAGR, max

drawdown, price, likes, downloads, rating and date. Click a heading to sort by it — best values first, click again to reverse — and click a row to open the artifact. It is the quickest way to compare many strategies on the same numbers. On a phone the storefront always shows cards.

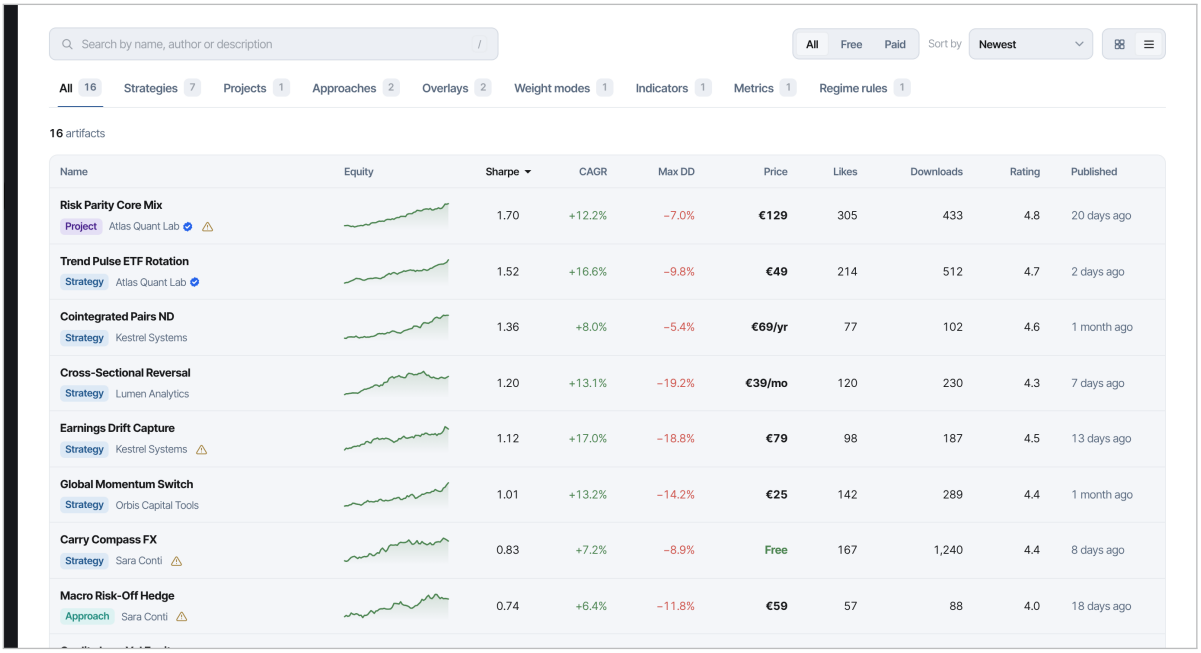


Figure 7.3. The table view, sorted by Sharpe. A dash means the artifact has no such number (it has no backtest of its own).

7.3 The artifact page

At the top, the type and the asset categories it was tested on, the name and version, the author's summary, and a line with the author — **Verified seller** when the payment provider has checked their identity — the date of publication, rating, likes and downloads. On the right, and at the top on a narrow window, the **purchase card**: the price, whether it is paid once or per month or year, and the button — **Buy for €49**, **Get for free**, or **Install into the system** once it is yours.

A bar lists the sections and follows you as you scroll: **Performance**, **Overview**, **Instruments**, **Data** (only when the artifact needs auxiliary data), **Content** and **Comments**.

- **Performance** — five numbers: Sharpe ratio, CAGR, max drawdown, total return over the whole out-of-sample period, and the share of positive months. Below them the equity curve: the scale on the right is the change from the start, and the label on the scale is the last value. Move the pointer over the curve: the crosshair shows the return from the start and the drawdown at that point. Under it the monthly returns; a bar shows its value under the pointer.
- **Overview** — the author's description, the steps of how it works and, if there is one, the presentation video.
- **Instruments** — the asset categories it works on and, for each, the instruments it actually traded out of sample, with result, yield, contribution and the period covered.
- **Content** — the files you get; the code itself after purchase.

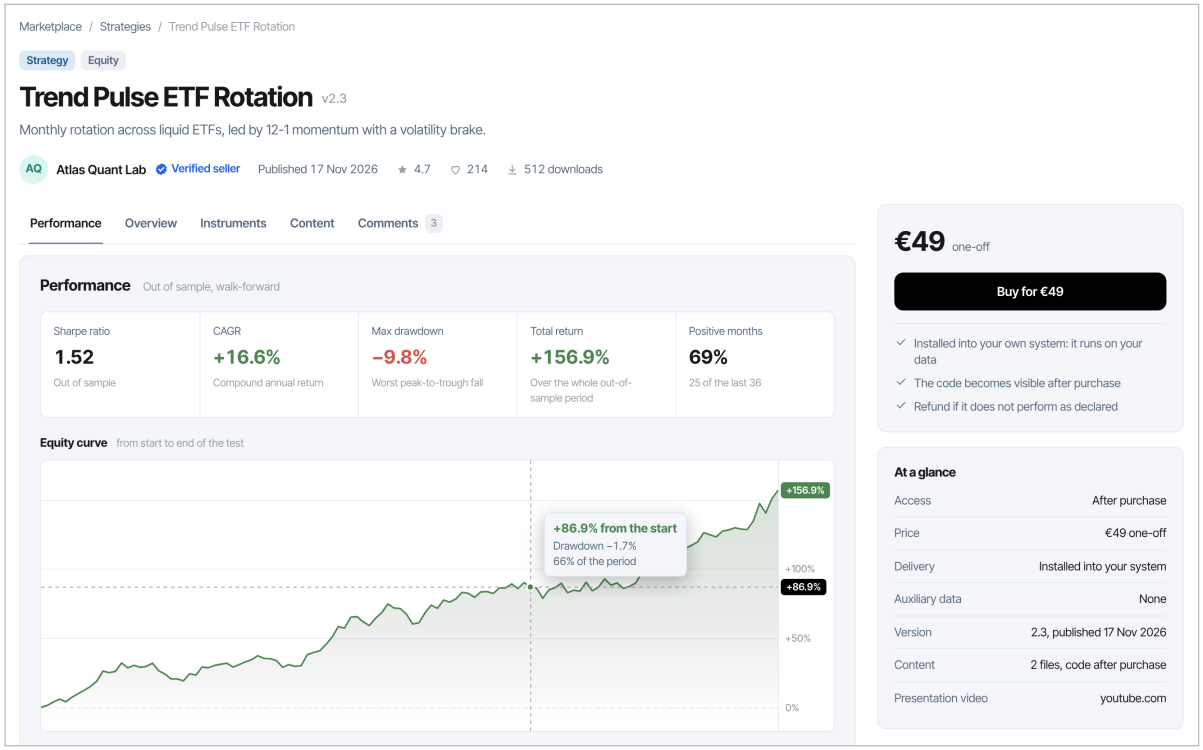


Figure 7.4. The page of a strategy, with the pointer on the equity curve: return from the start and drawdown at that point.

7.4 Buying

From the storefront to the system

1. Open the artifact's page (Section 7.3): out-of-sample results, description, instruments, data it needs, comments.
2. **Buy for ...** — **Get for free** when it is free, and access is immediate; if it is paid, the payment flow opens.
3. Once the payment is done you go back to the application. If access does not show as active yet, **it is not an error**: the confirmation arrives a moment later. The page says so and asks you to wait.
4. **Install into the system** — the artifact is written to the system, and from then on it appears in the lists like all the others.

Before installing, the conflicts

If something with the same name already exists in the system, the application says so *before* writing. It is worth reading: an installation that overwrites a strategy you were working on is lost work.

7.4.1 The data an artifact needs

Many strategies read only prices. Others also read **auxiliary data**: earnings releases, news, fundamentals, policy rates, macroeconomic series, the MMS features, or a category the author created on the **Data** page (Section 8.1). Without those files such a strategy still runs, but on empty series: what it does on your computer has nothing to do with the author's results.

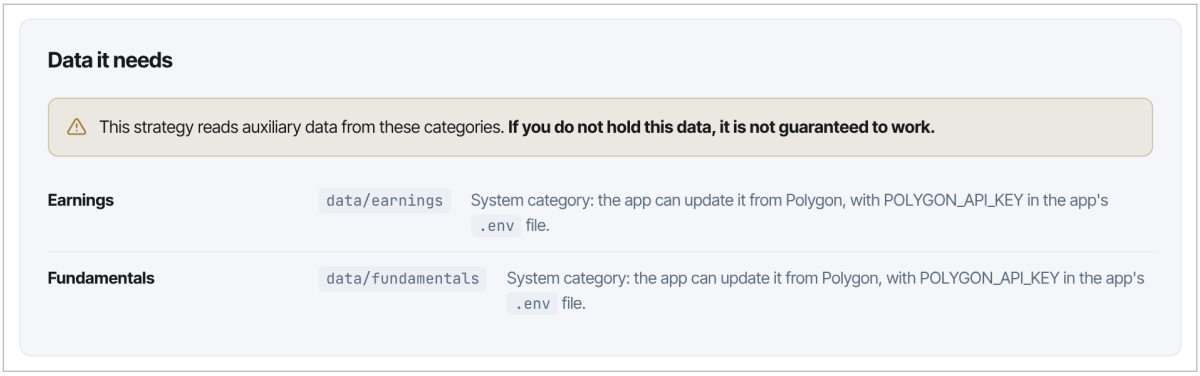


Figure 7.5. An artifact that reads two categories of auxiliary data. Before buying or installing, the page says which ones, where they go and who keeps them up to date.

- **In the storefront**, the card says **Needs data:** with the categories.
- **On the artifact's page**, the **Auxiliary data** panel lists them with the warning 'If you do not hold this data, it is not guaranteed to work': for each one the folder (**data/earnings**) and who keeps it up to date — a system category the application can download (with the provider's key, when it needs one), one the application computes from the prices, or one of the author's, whose files are not in the download. **At a glance** repeats them; **None** means the artifact reads only prices.
- **When you sell**, the page that publishes shows the same categories before you press the button: it is what buyers will read.

The categories are read from the artifact's own code, not declared by the author: for a project they are those of all its strategies and approaches.

7.4.2 What bought code can do on your computer

An artifact is code written by someone else, and it runs on the computer where your broker credentials, your account and your own strategies are. So the marketplace allows code that **computes on the data Lyra gives it**, and nothing else: no files, no network, no programs, no looking inside Python, no changes to the rest of the system. Three checks make sure of it.

1. **Before installing**, Lyra reads the code on *your* computer — whatever the site did before — and installs nothing if it does not pass. The page says why, file by file: 'Not installed: this artifact's code does things that marketplace code may not do… strategy.py:12 — 'open' is not available in marketplace code'.
2. **Every time it is loaded**, if one of its files has changed since it was installed, the file is checked again. A change of yours that stays within the rules is kept; one that does not stops that file from running.
3. **While it runs**, a guard stops anything it tries outside the rules, even through a function of Python or of Lyra: reading your files and keys, writing to disk, the network, starting programs. The backtest says which artifact tried what: 'Lyra blocked strategy … from the marketplace: it tried to use the network'.

The backtest of a bought artifact starts from Lyra's own launcher and without your keys in its environment. In a runner, a bought strategy reads the portfolio but cannot change it, and works on a copy of the data the orders are computed from.

Bought is not yours to sell

What you installed from the marketplace is someone else's work: it does not appear among the things you can publish, a project that contains it cannot be published, and automatic publishing leaves it alone.

7.4.3 If a purchase does not show up

There is a command for exactly this, on the artifact's page, that explains why it does not show as purchased. It tells apart two things that at first sight look the same: 'you did not buy it' and 'I could not ask'. They are opposites, and the second is solved by trying again.

If the payment went through but access does not open, there is a reconciliation command: it checks the payment again with the provider and, only if it shows as collected, completes the operation.

7.5 Selling

To publish you need two things, in this order:

1. **The marketplace must be open.** During the beta it is closed: nobody uploads anything. Items already in the storefront stay visible.
2. **A paid plan.** With the marketplace open, any paid plan will do, from Pro up (Chapter 9). On Free you can browse, download and buy, but not sell.

Why a subscription even to publish for free

It is not about the money. Whoever publishes puts their name next to a strategy that others will run with their own money: a filter at the entrance is worth more than any check after the fact. Someone without a subscription remains a full user — they browse, download, buy and use. They just cannot put things in the storefront.

A published artifact goes online at once: there is no approval queue. The description and the video can be corrected later; the bundle of files cannot.

Empty streams do not go in the storefront

A strategy with an empty stream (Section 4.1.4) has no backtest results to show whoever buys it, and a project that contains one would show the numbers of only part of what it sells. Neither can be published: the list marks them, the checks show the reason before you try, and Publish answers with the names of the streams — 'A project cannot be made of empty streams: in «Live test», «live_test_rules» and «live_test_model» are empty streams…'. Run their backtests, then publish.

The code must only compute

Buyers check the code before installing it (Section 7.4.2), and the page that publishes runs the same check first: the **Code check** row reads 'Passed: computation only' with the libraries the code uses, or lists the lines to fix, with file and line. Until it passes, Publish is refused — an artifact nobody could install is not worth putting in the storefront. What marketplace code may use: the standard library for computation, numpy, pandas, scipy,

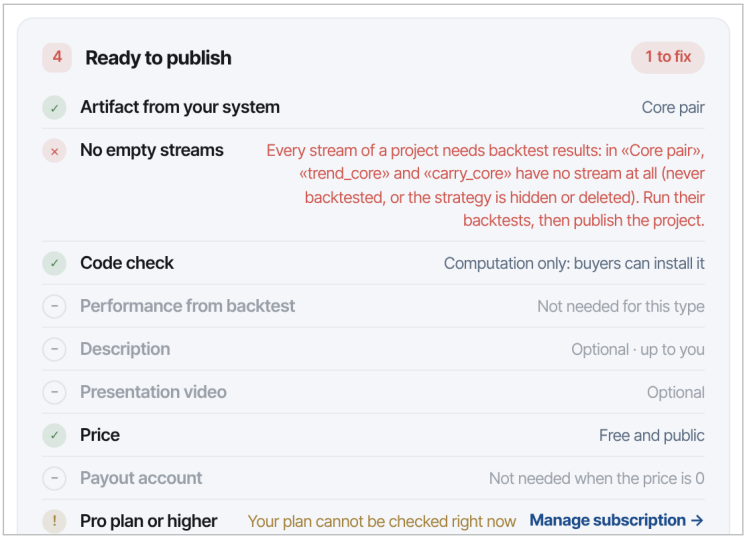


Figure 7.6. A project made of streams without backtest results: the checks name them before you try to publish, and the attempt is refused with the same reason.

> scikit-learn, statsmodels and lightgbm, and the parts of Lyra meant for artifacts (types, indicators, universe, market data, auxiliary data by category, metrics). The developer's guide has the full list.

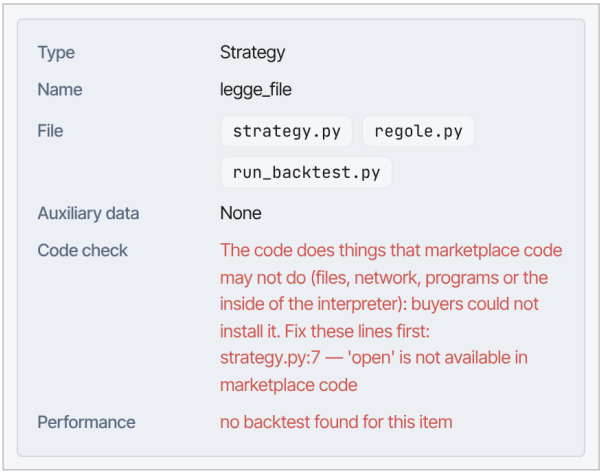


Figure 7.7. A strategy that reads a file of its own: the code check names the line, before you try to publish.

7.6 If an artifact does not perform as declared

You open a dispute from the artifact's page. It is not a discussion: the system runs two automatic checks.

Check	What it compares
Integrity	The installed files with those sold, one by one.
Performance	The metrics declared by the seller with those obtained by running the calculation again on your own system.

The case is a conversation between buyer, seller and the administration, with the results of the two checks attached. It can be withdrawn at any time.

> **Two different results do not mean a deception**
>
> A strategy can perform differently because the data are different: a different universe of instruments, a different period, different costs. The check is there to establish *by how much* they differ, which is the starting point of any sensible discussion.

8

Settings and data

The **Settings** item gathers three tabs: **Config**, **Data** and **Market rules**. They are the three places where you tell the system what to work with, and within which limits.

8.1 Data: bringing in the prices

Without data, every page of the application is empty. It is the first thing to do after installing.

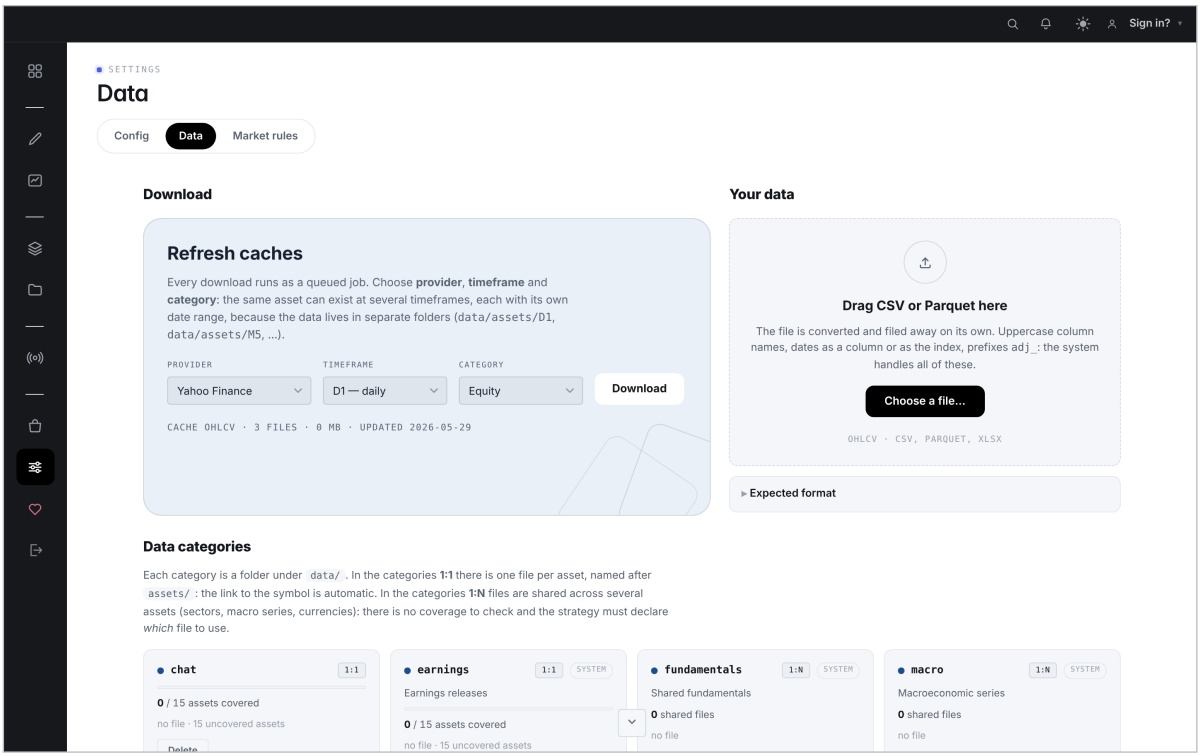


Figure 8.1. The Data page. On the left the automatic download, on the right the manual upload, below them the categories.

8.1.1 Downloading

You choose three things and press **Download**:

Provider
: Where to get them from. Some need a personal key, which you set up from the site.

Timeframe
: The time step. The same instrument can exist at several steps: daily data and five-minute data live in different folders and do not get in each other's way.

Category
What kind of data it is.

The job goes into the queue and the bell tells you when it is done. Below the fields, a line says what is already there: how many files, how much space, up to when they go.

Intraday bars from Yahoo Finance

Yahoo Finance is free and needs no key, but keeps intraday bars only for a while: the last 60 days for 5, 15 and 30 minutes, 730 days for one hour. The download takes what there is. For stocks it fetches the benchmark (SPY) first, because no backtest starts without it: a 15-minute strategy on currencies needs SPY at 15 minutes too.

Dividends: the whole bar is adjusted, not only the close

Stock prices are adjusted for dividends: the day a dividend is paid the price drops, and without the adjustment that drop would look like a loss. Lyra applies the adjustment to open, high, low and close alike — the backtest buys at the open and values at the close, and adjusting only the close made every entry and every exit jump. On a stock paying 2 % a year, traded in and out every month, the old way turned a Sharpe of 0.97 into 0.45. The live prices are adjusted the same way.

8.1.2 Uploading your own file

The **Your data** panel accepts CSV, Parquet and XLSX files dragged onto it. The system recognises by itself columns written in capitals, dates given as an index or as a column, and prefixes such as *adj_*. The **Expected format** panel shows what it expects.

8.1.3 The categories

Each category is a folder, and they come in two types.

Type	It means
1:1	One file per instrument, named after the instrument. The link is automatic, and the application can say how many instruments are covered and how many are not.
1:N	Files shared by several instruments: sectors, macroeconomic series, currencies. Here there is no coverage to compute, and the strategy has to declare which file to use.

'0 / 15 assets covered' is not a fault

It means the category exists but has no data yet for any instrument. It fills up by downloading, or by dragging files into it.

8.1.4 Keeping a category up to date

A strategy declares the categories it reads, and how often they must be refreshed. Each category has a *source* that keeps it up to date: the system one for earnings, fundamentals, news, rates, macro series and MMS features; for a category of yours, a file you write in its folder. The card of each category says who updates it and how it went.

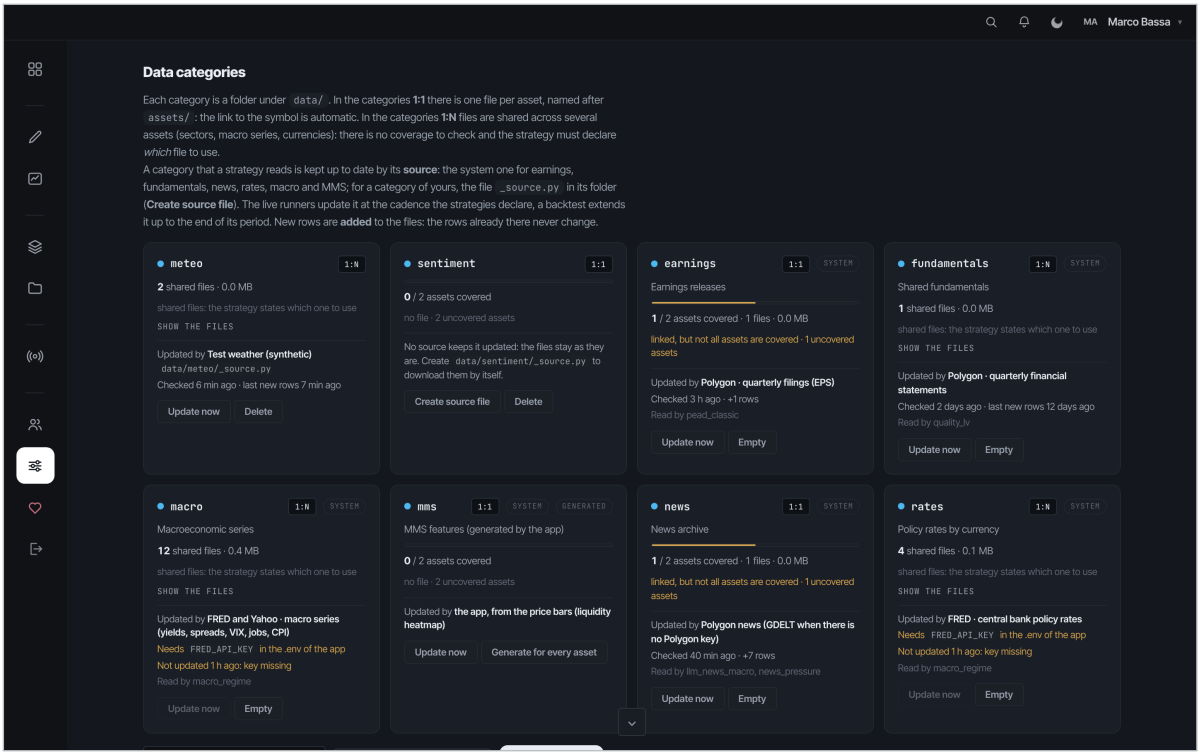


Figure 8.2. The categories with their sources. **macro** and **rates** wait for a FRED key; **sentiment** has no source yet; **meteo** has one written by the user.

Updated by
: The source: a provider (Polygon, FRED), the application itself (MMS), or your file `data/<category>/_source.py`.

Checked ... ago
: The last update and how many rows it brought. In amber when the last attempt failed, with the reason when you hover on it.

Needs ... in the .env of the app
: The key the provider needs is missing: *POLYGON_API_KEY* for earnings and fundamentals, *FRED_API_KEY* for rates and macro series. They go in the `.env` file of the application (FRED keys are free); the key typed on this page for downloading prices stays in the browser and is not used for the automatic updates.

Read by
: The strategies, overlays and weight modes that declare the category: they are the reason it is updated.

Update now updates the category straight away, as a job. **Create source file**, on a category with no source, creates the file to fill in, `data/<category>/_source.py`, with the instructions written inside it: one function, `fetch`, that returns the new rows of a file. **Empty** removes the data and keeps the source.

> **New rows are added, the old ones never change**
>
> An update *adds* what is new to each file: the rows already there stay exactly as they were, even if the provider has corrected them since. A backtest repeated tomorrow sees the same

> past as today, and a value the provider revises later does not rewrite what was known on that day.

You rarely need the button. A backtest extends the categories its strategy reads up to the end of its period, before starting; a live runner updates them in the background at the cadence the strategies declare (Section 6.5) — into a folder of its own, as the next section explains.

8.1.5 The data of the live runners

A live runner downloads and computes data while it works: the new rows of the categories, the bars of the instruments at every bar, the history it fetches from the broker and the providers to give the strategies their context, the market-stress and macro risk-off series. None of it goes into the files your backtests read. It goes into `data/live`, with the same folders as `data/`, and holds only what `data/` does not have yet. The runner reads the two together; backtests and every other page read only `data/`, so a runner never changes the data of a backtest — not even one running at the same moment.

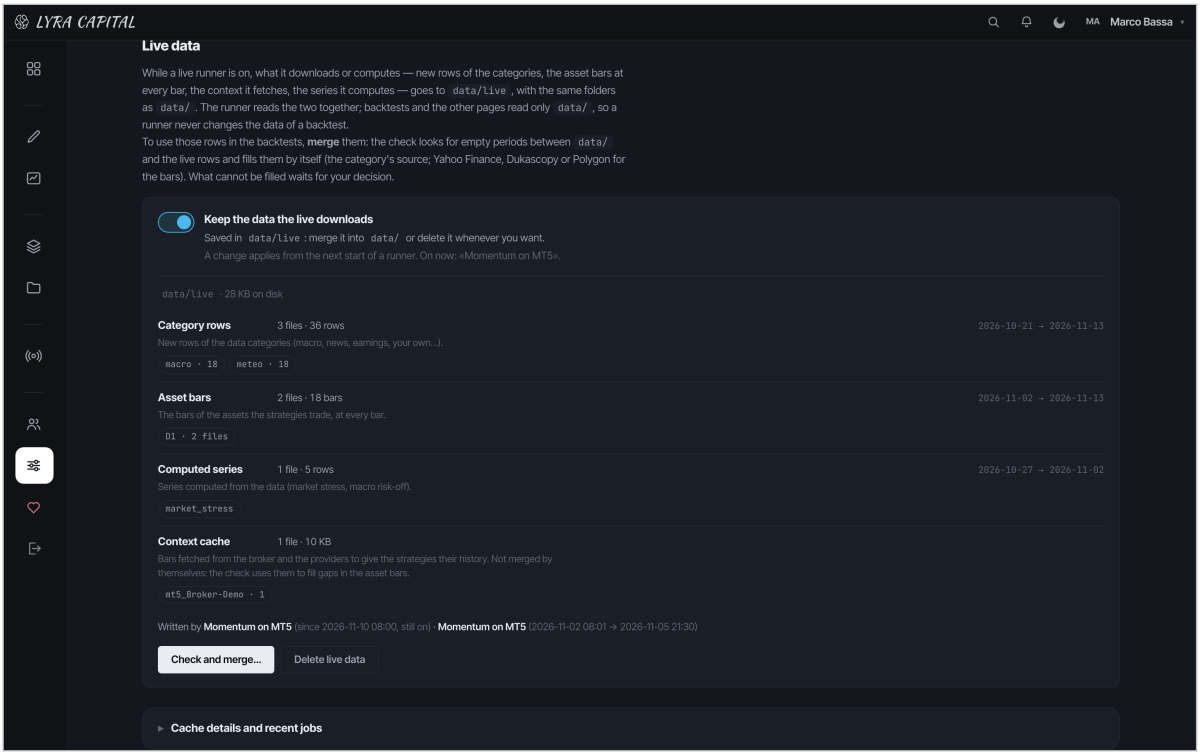


Figure 8.3. The **Live data** section of the Data page: the setting, what is in `data/live` by area, the runner sessions that wrote it.

Keep the data the live downloads
: On, the data stays in `data/live` and the runner starts again from there the next time. Off, the runner works exactly the same, but what it downloads goes to a temporary folder that is deleted when the application closes: nothing to merge. While the application stays open, what the runners no longer need is deleted from it at the cadence you choose (below). A change applies from the next start of a runner; the page says which runners are on.

The summary
: One line per area — category rows, instrument bars, regime drivers, computed series, the

context cache — with files, rows and period; below, the last merge and the runner sessions that wrote.

Delete live data
Removes `data/live`. A runner that is on downloads again what it needs, from where `data/` ends.

With saving off: what the runners no longer need. A runner can stay on for months, and so can the temporary folder. With saving off, **Delete what the runners no longer need** chooses when it is cleaned: **As soon as it is no longer needed** (checked every minute, and as soon as a runner stops), **Every hour** (the default), **Every 6 hours**, **Every day**, **Every week**, or **Only when the app closes**, as before — the folder then grows as long as the application stays open. The choice applies at once, also to the runners already on.

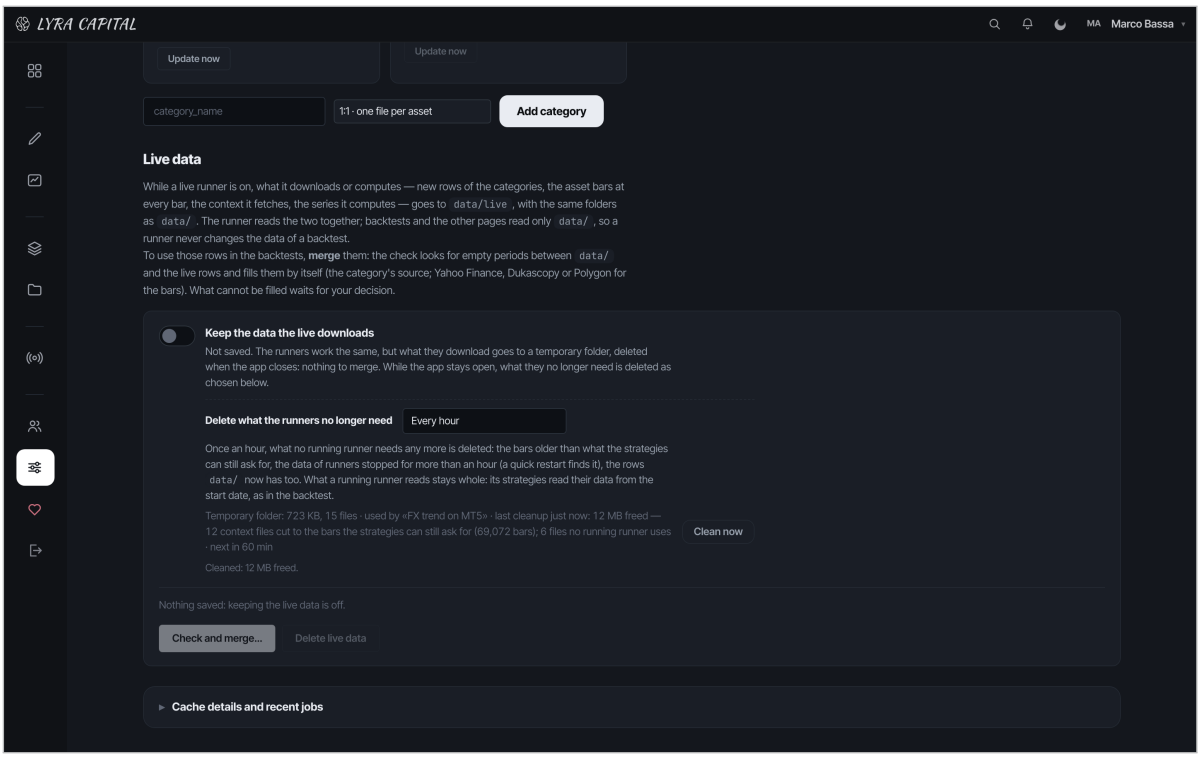


Figure 8.4. Saving off, cleanup once an hour. One runner is on; the one stopped three hours ago had left six files. **Clean now** removed them, together with the bars the strategies can no longer ask for.

What goes

The bars older than what the strategies can still ask for: every component declares how many bars it needs, and the most it may need, and one and a half times that stays. The data of the runners stopped for longer than the interval, unless a runner that is on uses it too: a quick restart finds its data, and a category that went is downloaded again as soon as a runner needs it. The rows `data/` has as well, after an **Update now**. With no runner on for longer than the interval, the whole folder.

What stays

Everything a runner that is on reads, whole. Its strategies prepare their data from the start date of the backtest, and an overlay like news pressure compares today's news with all of its history: deleting old rows would change their decisions, not only the space.

The line below

How much the folder holds, which runners use it, the last cleanup and the next. **Clean now** cleans at once, without waiting for the interval. A cleanup that deletes something says so in the log of the runners that are on.

Merging into the data of the backtests. **Check and merge...** compares every file of `data/live` with `data/` and looks for *empty periods*: the market sessions with no bar, the days with no row — typically the days the application was off. A holiday is not a gap. It then tries to fill them by itself: for a category, its source; for the bars, first the provider the file comes from (it downloads the period again, and is accepted if its returns match the file where the two overlap), then the bars of the live, then the other providers — Yahoo Finance, Dukascopy, Polygon — brought to the level of the file and checked on the bars they have in common. The bars a session should have are counted in New York time, so the change to and from daylight saving time, which moves the opening of Wall Street by an hour in UTC, is not a gap.

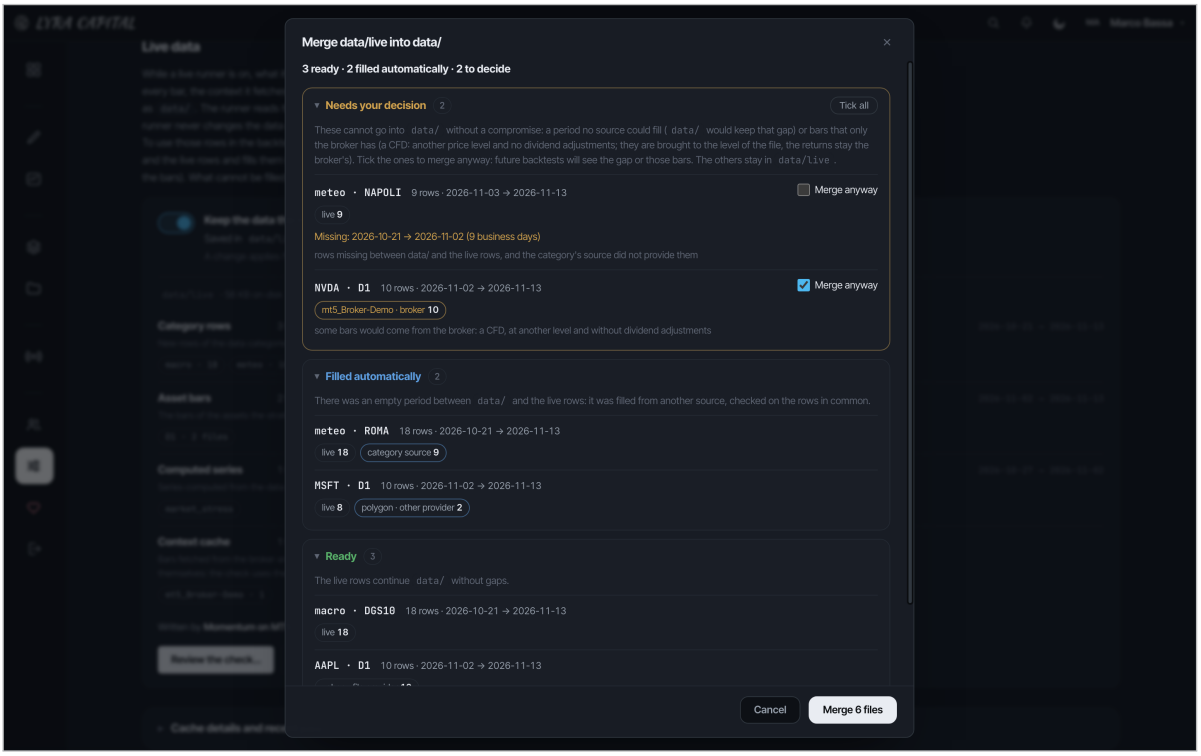


Figure 8.5. The check: two files to decide, two filled automatically, three ready. **NVDA** has only broker bars and is ticked to merge anyway.

Ready

The live rows continue `data/` with no gap.

Filled automatically

There was a gap, and another source filled it; the chips say how many rows came from where.

Needs your decision

A period no source could fill, or bars that only the broker has. Tick **Merge anyway** to accept it, or leave the file in `data/live`.

The merge adds the rows to `data/` without changing the ones already there, removes them from `data/live`, and the instrument remembers the merge (when, from which sources, which gaps

were accepted). The check and the merge run in the background: you can close the window and come back.

Merge anyway is a decision about every future backtest

A gap you accept stays in `data/`: the backtests over that period will find no bar, or no row, for those days. Broker bars are the prices of a CFD: they are brought to the level of the file, but the returns are the broker's, without the dividend adjustments of the provider. When in doubt, leave the file in `data/live`: a check the next day, when the provider has the period, may fill it.

8.2 Config: the limits

These are the parameters that apply to every calculation.

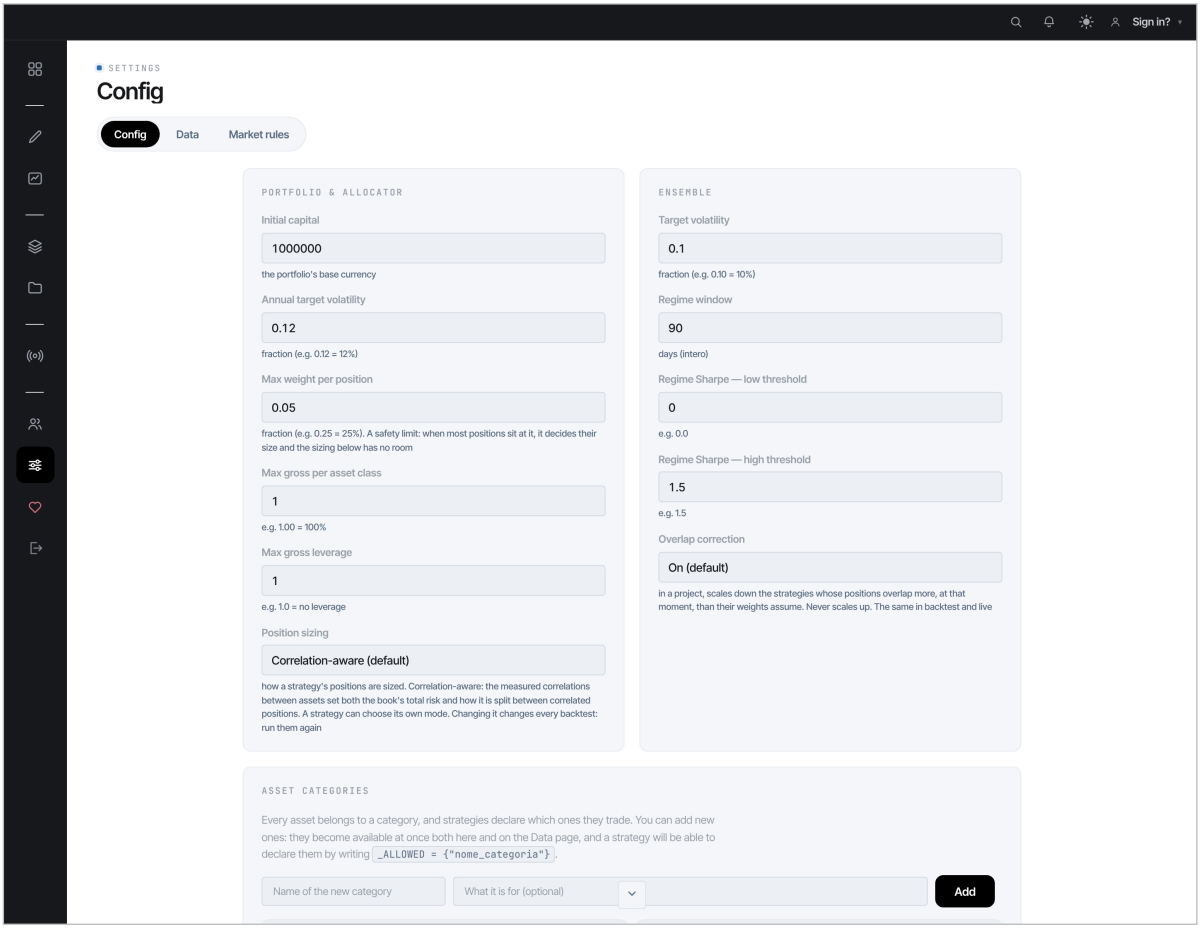


Figure 8.6. The parameters. Each field carries its unit below it, because the difference between 0.12 and 12 is a factor of a hundred. At the bottom of each section, how positions are sized and whether a project corrects the strategies that overlap.

Field	What it limits
Portfolio & allocator	
Initial capital	The capital the calculations start from.
Annual target volatility	How much you accept the portfolio swinging in a year. It is the main risk dial.
Max weight per position	The most a single instrument can weigh. It is a safety limit, not the sizing: if most positions sit at it, they all take the same size and the correlations can change nothing. The page of a stream tells you when that happens (Section 4.2.1).
Max gross per asset class	The most a whole class can weigh — shares, bonds, currencies. It guards against diversification that is only apparent: four bonds all going the same way are a single position.
Max gross leverage	The total exposure. 1.0 means no leverage.
Position sizing	How a strategy's positions are sized. **Correlation-aware**, the default: the measured correlations between the assets set both the total risk and how it is split between correlated positions (Section 4.2.1). The other three are the earlier ways: correlations measured for the total only, assumed from the asset class, or weights declared by the strategy. A strategy can choose its own.
Ensemble	
Target volatility	The volatility every stream is brought to *before* being weighted.
Regime window, thresholds	How the state of the market is assessed for the modulation.
Overlap correction	In a project, scales down the strategies whose positions overlap more, at that moment, than their weights assume. Never scales up (Section 5.4). On by default.

Target volatility also acts on the real account

It is not a display setting. The same factor that normalises the streams in the backtest multiplies the positions the runner opens on a real account. Changing it changes how much you really risk, not just what you see on the chart.

Changing Position sizing changes every backtest

The sizing decides every position, so results made with one way are not comparable with those made with another. After changing it, run the backtests again: until you do, the pages show the results of the old way.

Further down, **Asset categories** lets you add categories of instruments: once created, a strategy can declare them and restrict itself to them.

8.3 Market rules: the regimes

A **regime** is the state of the market, with three values: **bull**, **neutral**, **bear**. Strategies can declare that they weigh things differently according to the state.

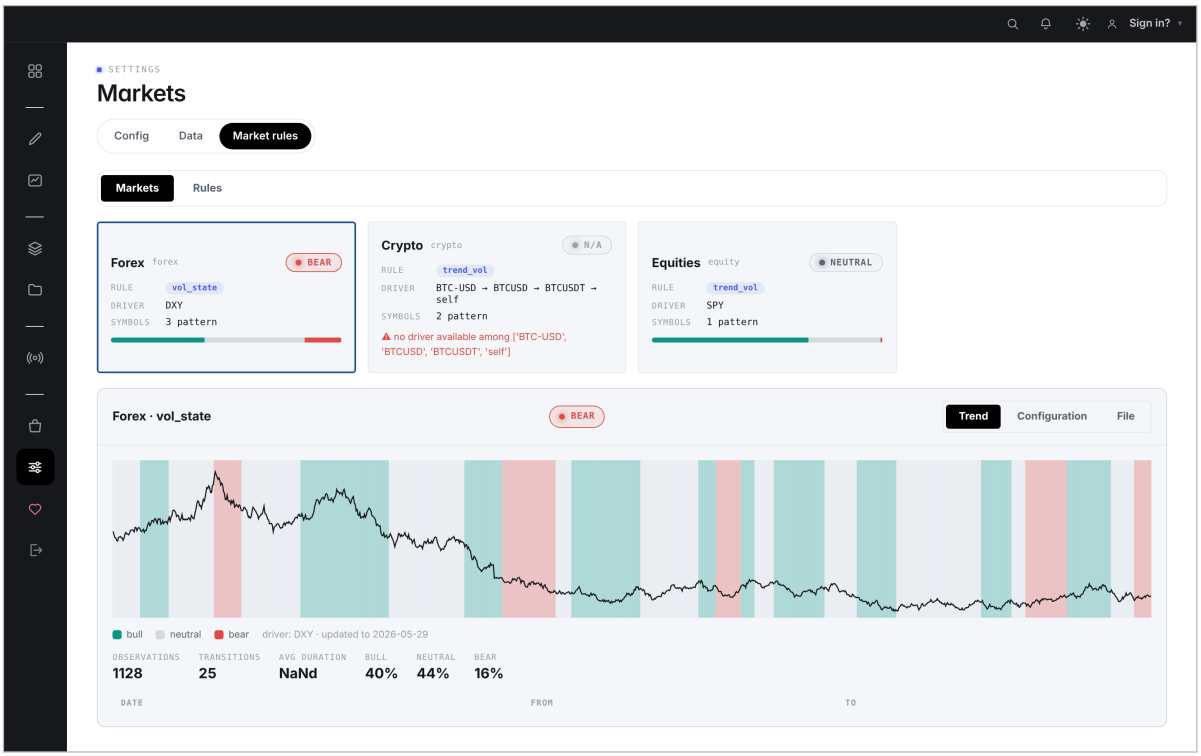


Figure 8.7. The configured markets. Below them, the history of the states: green bull, grey neutral, red bear.

8.3.1 How to read the page

Each panel at the top is a *market* — a set of instruments that share the same way of measuring the regime:

RULE
: The rule used to compute the state.

DRIVER
: The series that drives the decision — for shares, SPY; for forex, the dollar.

SYMBOLS
: How many name patterns fall into this market.

The coloured bar
: How much time was spent in each state.

> **'No driver available'**
>
> The panel of a market that cannot find its driving series says so with a warning, like the **crypto** market in the screenshot. That market **modulates nothing**: it is not an error that blocks anything, but the strategies on those instruments behave as if the regime were always neutral. The fix is to download the missing series from the **Data** page.

8.3.2 The chart

It shows the driving series with the background coloured by state. Below it, the counts: how many observations, how many times the state changed, and the percentage of time spent in each.

> **Few changes of state is a good thing**
>
> A regime must be a *state*, not a reaction. Twenty-five transitions in five years mean the rule tells different periods apart; two hundred would mean it is chasing noise, and a strategy that changes weight every week pays costs without taking a position.

9

Account and subscription

9.1 What the account is for

Four things, and nothing else: entering the marketplace and receiving messages — both in the **Community** — having a public profile, and taking out a subscription. The rest of the application works without one.

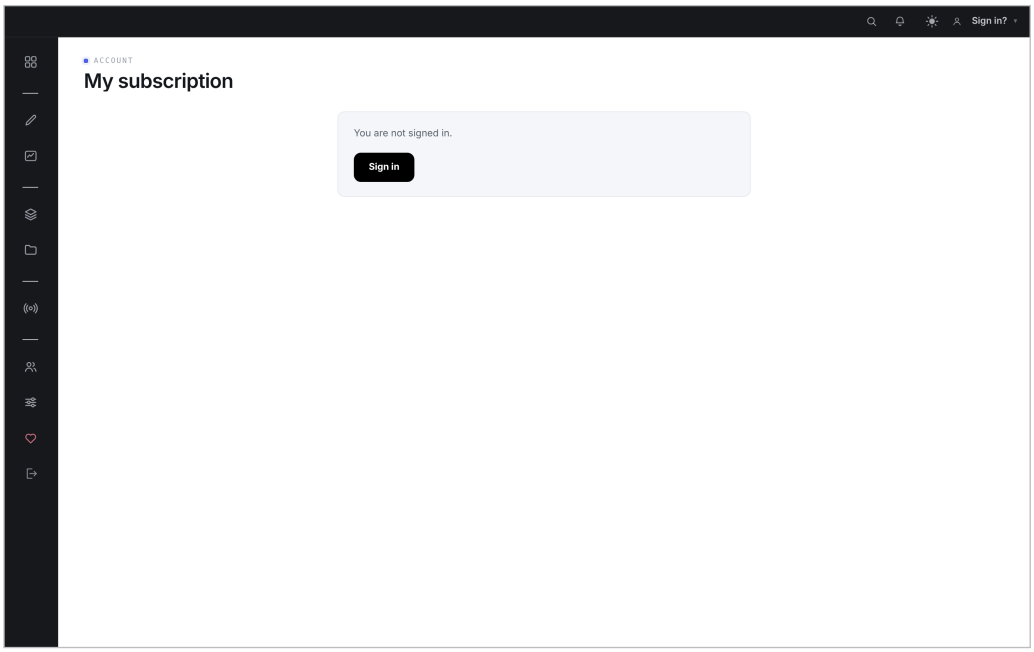


Figure 9.1. Pages that need an account say so and offer to sign you in, instead of disappearing from the menu.

9.2 Signing in for the first time

During the beta

During the beta you do not create an account yourself: Lyra creates it when you ask to join, on the site, with **Entra nella beta**.

1. When the beta starts, an email arrives with the link to download the application, your address and a **first password**.
2. Install the application, press **Sign in** and enter that address and that password.
3. The application asks you to choose your own password, twice. From then on you sign in with it: the one in the email no longer works.

If you already had an account with that address, you keep it and its password: the email only brings the link.

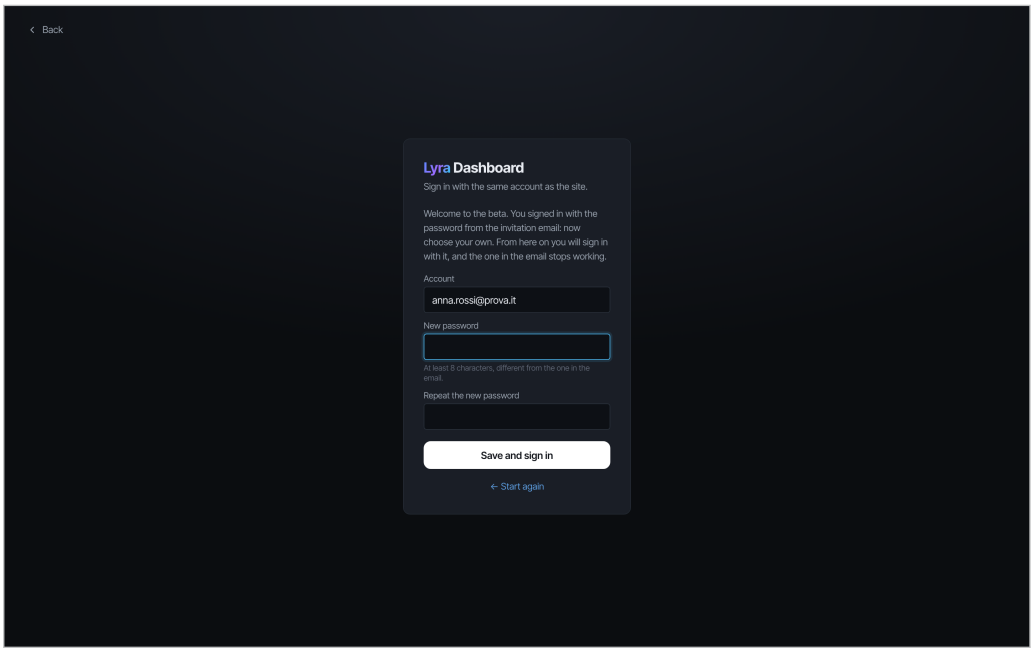


Figure 9.2. The first sign-in with the password from the invitation: the application asks for your own, at least eight characters and different from the one in the email.

The places in the beta are limited. When they are full, a request stays on the waiting list: the site says so, and if a place frees up the team adds you and the email arrives. An address that did not ask to join is refused, and so is an attempt to sign up from the application while the beta lasts; the sign-in page says which of these is the case.
You can also sign in with Google (**Continue with Google**), with the address of an account you already have: the sign-in opens in the system browser and the application picks up the result.

9.3 The profile

Profile, in the account menu at the top right, gathers the account on one page. **Sign out** is at the top right of the page.

Account details

Display name and bio — what others see next to a published artifact — changed with **Edit profile**.

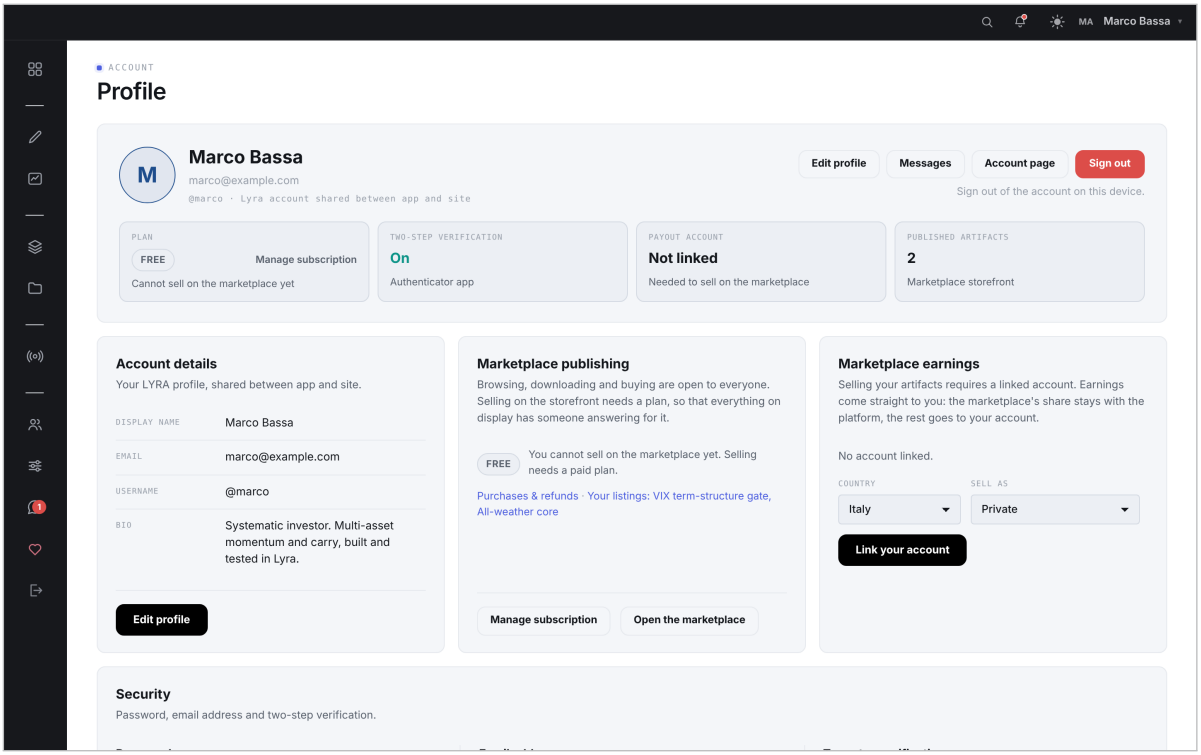


Figure 9.3. The top of the profile: the plan, two-step verification, the payout account and the published artifacts at a glance; below, the account details, marketplace publishing and earnings.

Marketplace publishing
: Whether you can sell, with links to your purchases and your listings, and to **My subscription**.

Marketplace earnings
: The payout account that receives what you sell. Selling for a price needs one, and linking it needs two-step verification first.

Automatic publishing
: Off unless you turn it on (Section 9.3.1).

Security
: The password, the email address and two-step verification. They are not stored by the application: they live on the site, and the application only passes them on.

Preferences, Brokers and data keys
: The theme, how pages open, hardware acceleration and the notifications — the same choices as the **Settings** window (Section 9.4); the broker connections and the data keys, which stay on this computer.

Send us feedback
: A report or a suggestion for the Lyra team.

9.3.1 Automatic publishing

Lyra can upload your best work to the marketplace by itself — only if you ask it to. With the switch on, whenever one of your strategies, approaches or projects reaches an out-of-sample walk-forward Sharpe above 2.0, Lyra uploads it under your account, code and results, **free and not public**: only the Lyra administrators see it. Nothing gets a price, nothing appears in the

storefront, nothing is shown to other users.

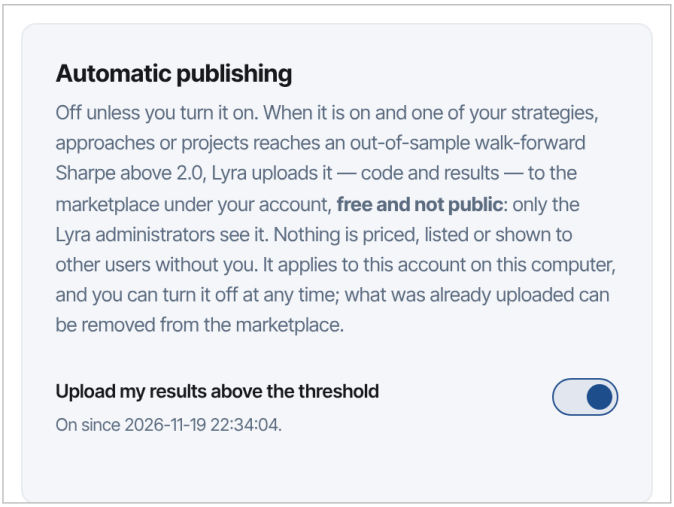


Figure 9.4. Automatic publishing in the profile, switched on. Until you turn it on the line reads 'Off: nothing is uploaded.'

- The choice is for the account you are signed in with, on this computer. It is off for everyone who has not turned it on — including after an update.
- You can turn it off at any time, from the same switch. What was already uploaded is listed under the switch, and can be removed from the marketplace.
- What you bought in the marketplace is never uploaded: it is someone else's work.

How Lyra recognises this installation

With a random number, made the first time Lyra starts and kept with your data. It says nothing about your computer: before, it was derived from the computer's hardware, network card included. It counts towards the limit of installations of a purchase; deleting it makes Lyra a new installation.

9.4 Speed: how pages open, hardware acceleration

Two choices in the **Settings** window (account menu at the top right), and in the preferences of the profile.

Opening pages

Right away, the default: a new page opens at once and fills in as its data arrives, with the thin bar at the top of the window running until it has everything. **When loaded**: the page you are leaving stays on screen until the new one has its data and charts, up to two seconds — no page filling in before your eyes, steadier on slower computers. Quick pages look the same either way. The choice applies from the next page you open.

Hardware acceleration

Only in the desktop application; in a browser it is a setting of the browser. **Automatic**, the default, lets the graphics card draw the pages whenever Chromium supports it: almost always the fastest. **Always** uses the card even when Chromium does not trust it (older integrated graphics, old drivers): smoother pages where the processor was drawing them, but a faulty driver can make the window flicker or go black. **Off** draws with the processor

only, for driver problems; it is slower. Below the buttons, what the card is doing in this session. A new choice applies when the application restarts: Restart now does it.

If the window goes black after choosing Always

Start the application with the environment variable `LYRA_GPU=off` (or `auto`): it overrides the saved choice for that session, and from **Settings** you can set **Automatic** again. If the graphics card stops responding twice with **Always**, the application goes back to **Automatic** by itself at the next start, and the section says so.

9.5 The plans

The plans are the same as on the Pricing page of the site. Free is the whole application; you pay for privacy, data and a higher share of what your strategies earn — not for using the application. They are on the **My subscription** page, in the account menu at the top right.

Plan	Price	What it adds
Free	€0, forever	The whole application: building and backtesting locally with your data, the local AI copilot, out-of-sample validation, live runners and kill-switch. On the marketplace: browse, download, buy. Share of earnings 0%: an allocated strategy enters the track record.
Pro	€49 a month or €490 a year	Full privacy for code and metrics; selling on the marketplace and buying at −10%; the institutional-data sandbox (from 2030); 10% of what your allocated strategies earn.
Professional	€149 a month or €1,490 a year	Raw history on your machine (from 2030) with −10% on data vendors; marketplace −20%; sandbox on more markets; 30% of earnings. It is the recommended plan.
Quant	from €200 a month or €2,000 a year, plus data	Exchange-licensed real-time (from 2032), in per-market packages; extended sandbox; 30% of earnings.
Desk	€8,000 a year	Five seats, installation on your servers, compliance export, white-label and SLA; 30% of earnings, per seat, by contract. Sold by contract, after a demo.

The features with a year

Some features arrive with a later release: the sandbox and the raw history in 2030, real-time in 2032. The cards and the comparison table show the year next to them.

During the beta

The paid plans open after the beta. Until then Free is the whole application, nothing can be bought or sold on the marketplace, nothing is charged, and the plan buttons say **On sale after the beta**. Desk demo requests open later too: until then the Desk button says **Coming soon**.

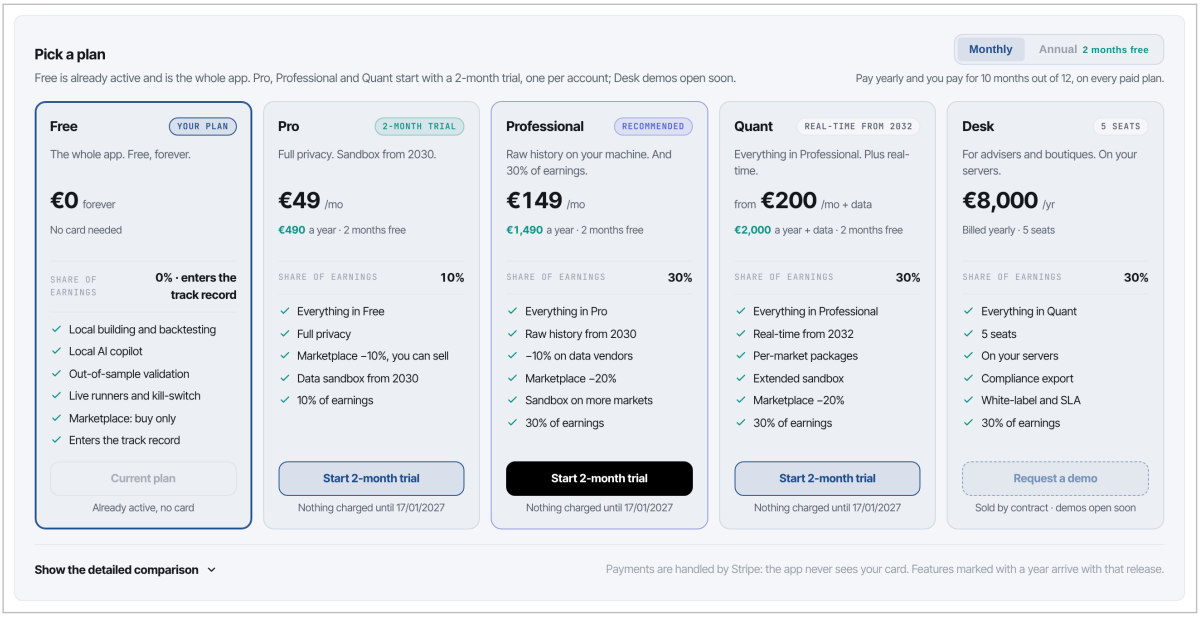


Figure 9.5. The plans on **My subscription**: price, share of earnings and what each plan adds. **Show the detailed comparison** opens the table with every feature, plan by plan.

The share of earnings

It is the part of what a strategy earns, once it is allocated to capital, that goes to its author: 10% with Pro, 30% with Professional, Quant and Desk, set by contract. On Free an allocated strategy enters the track record without a share. Every submission is opt-in, strategy by strategy and under a contract: nothing is submitted without your consent. The site describes the steps, from the protocol to the live track record.

9.5.1 The two-month trial

Pro, Professional and Quant start with a two-month trial, one per account.

Starting the trial

1. On the plan's card, press **Start 2-month trial**; the hint below the button says until when nothing is charged.
2. The window shows **Charged today** at €0, the day the trial ends and the price that applies afterwards. Choose monthly or yearly there, then press **Start the trial**.
3. The Stripe page opens: Stripe asks for your card but charges nothing. The page updates by itself and shows **Trial until** and the date.
4. On that date the first payment is taken. Cancel before, and nothing is ever charged.

Changing plan during the trial costs nothing, and the trial keeps its end date. An account that has already had a trial, or a paid plan, pays from the first day: the button then says **Choose** followed by the plan's name.

9.5.2 Desk

Desk is not bought with a card: it starts with a demo on your firm's own strategies, and a contract. Demo requests for firms are not open yet. Until they are, **Request a demo** on the Desk

card is switched off: point at it, or reach it with **Tab**, and it says **Coming soon**; under it, **Sold by contract · demos open soon**. The Desk buttons on the site's Pricing page say the same. When requests open, the button opens a short form — the firm, how many people would use the platform, a message — and the request reaches the Lyra team, who reply by email and in **Messages**.

9.5.3 Monthly or yearly

Both. Yearly costs ten monthly payments: two months are free. Desk is yearly only. You choose when you subscribe and switch whenever you like, from the button next to your plan: Stripe works out the adjustment for the period already paid.

9.5.4 Changing plan

Upgrade and Downgrade on the cards change the plan at once: Stripe credits what you have already paid for the current period and charges the difference to the card on file. Before you confirm, the window lists what you gain or what you lose.

9.5.5 Cancelling, card, invoices

From Manage in Stripe on **My subscription**. The billing portal hosted by Stripe opens: there you cancel the subscription, change the payment method and download the invoices. The card never reaches the application.

> **Cancelling takes nothing you already have**
>
> The plan stays active until the end of the period you paid for. A purchased artifact stays purchased and installed, and strategies, code and data stay on your computer: you go back to Free, which is still the complete application. What stops is selling on the marketplace and the benefits of the plan.

9.6 When something does not add up

Symptom	What to do
'You cannot sell' but you have a subscription	Close and reopen the application: the link with the site is made at start-up.
The marketplace is empty	Same reason. If it persists, sign out and sign back in.
I paid and I cannot see the artifact	Wait a few seconds and reload. If it stays like that, use the command that explains why it does not show as purchased, on the artifact's page (Chapter 7).
The trial is not offered	The trial is one per account: an account that has already had one, or a paid plan, pays from the first day.
A page says it does not know, instead of showing a number	Not a fault: the application could not ask the site, and says so instead of making up an answer. Try again later.