BotFactoryFX

USER MANUAL

BotFactoryFX

Version 1.34.0

Everything: search, judge, quarantine — for your strategies and for a robot you were sold —, portfolio, your MetaTrader 5, JForex and web3 accounts on one screen, Dr. Quant and export to five platforms.

The same Help that lives inside the program, chapter by chapter. It is generated from the program's own source, so it can't say anything the program doesn't do.

Inside the program, this same Help lives under Help, with a search box (Ctrl+K), and Help → Take the tour walks through every screen in the order of the work.

BotFactoryFX 1.34.0 · 6 November 2026

Contents

GETTING STARTED AND BUILDING

What is BotFactoryFX 3

Home 3

Autopilot 3

Build › Manual 5

Build › Evolve — the genetic search 9

Build › Explore 12

JUDGING

Reality Check 14

The method 19

Installing the robot on each platform 21

DATA, LIBRARY AND PORTFOLIO

Data and timeframes 27

Connecting MetaTrader 5 29

Connecting JForex 32

Working with TradingView 34

Library 35

Portfolio 36

AFTER THE BACKTEST

Quarantine 37

Live accounts and wallets 38

Dr. Quant 40

Dr. Quant on the web — and what it costs 41

AT THE END

Frequently asked questions 43

Metrics glossary 46

Shortcuts and tips 46

GETTING STARTED AND BUILDING

What is BotFactoryFX

BotFactoryFX is an **algorithmic trading lab**: it builds trading strategies, and then spends most of its effort trying to prove the ones it found are worthless. Import your own price history, build a strategy by hand or let the genetic search find one, put it through the Reality Check, combine what survives into a portfolio, and export the result as MQL4, MQL5, JForex, a cTrader cBot, Pine Script or a Python bot that trades Hyperliquid. Your work is saved on this machine between sessions.

The menu on the left is the order the work happens in:

1. **Home** — the curve you are watching: a simulated portfolio or the last backtest, a quarantine, or a live account.
2. **Data** — the price history, its timeframe, what your broker charges you to trade it, and the instrument card.
3. **Autopilot** — say in one sentence what you want, and it searches, judges, saves and combines on its own.
4. **Build** — write a strategy by hand (*Manual*), let the genetic search find one (*Evolve*), or screen ideas before committing to one (*Explore*).
5. **Reality Check** — whether the result is real, told how many variants were tried to get there.
6. **Library** — the strategies you kept, with their reports, ready to download for any of the six platforms.
7. **Quarantine** — a strategy runs on candles that are still arriving, before it sees money.
8. **Portfolio** — combines what passed both, and simulates them together.
9. **Live** — the accounts that trade, with their real trades on the chart.

Home

Landing screen. Shows the KPIs of your hypothetical portfolio (or of the last backtest as fallback), an equity curve, and a list of recent saved strategies.

- **Equity curve**: a live account's if you have one, else a quarantine's, else the simulated one — the last portfolio simulation, or the last backtest. The buttons above it switch between them; see *Live accounts and wallets*.
- **Quick access**: shortcuts to Data, Build, Reality Check and Portfolio, in the order of the menu.
- **Recent strategies**: your last 5 saved strategies.
- **Quarantine at a glance**: what you have in quarantine, together — robots, closed trades and how many won, the result of all of them as a percentage of each one's account (what the same capital split between them would have done), each one's verdict, their combined curve and the whole list, best to worst. *Show* picks all of them, a folder, an asset, a timeframe or some of them one by one, and remembers it. Robots that already trade on an account are not added in: theirs is money an account made, not a simulation.

Autopilot

Its own section, before Build. Say in one sentence what you want — the instrument, the timeframe, the style, how much drawdown you can live with — and it does the rest: it searches, measures, picks, judges, saves and combines. The steps are the real operations, not a narration of them: the same genetic search as Build › Evolve, the same Reality Check, the same portfolio simulation.

The sentence is read in code, not by a model, so building needs no AI provider and cannot invent a timeframe you never mentioned. Two more orders are read in code too: “judge” plus a Library strategy's name runs the Reality Check on its own history, and “take” plus its name “to the portfolio” marks it in the Portfolio selection — each behind a card that says exactly what will run, and nothing runs without your click. The Autopilot builds new strategies and nothing else: an order about one you already have — “optimise the RSI period”, “change the stop loss” — is none of those, and it does not reach the model either. Dr. Quant answers it himself, saying that he runs nothing from this conversation and where it is done for real: an indicator period is changed in Build › Manual, the stop and the target in Money management, and then Backtest. Everything else is taken as a question, and questions go to Dr. Quant's model, which does need a provider.

A short order with no verb counts too — “S&P 500 daily, reversion and trend” — as long as it names an instrument and a style or a timeframe and is not a question; anything you are asking about still goes to Dr. Quant.

Before it starts

It shows what it understood and, under “You did not say, so I chose”, everything it had to decide for you, each with its reason. Change any of it by writing another sentence; Start runs it, Not yet leaves it. If something makes the request impossible — no history loaded, an instrument that is not loaded, a timeframe finer than the loaded candles — it says what, and offers no Start.

Pick the history and the candle in the strip at the top: a request that does not name another instrument runs on that one, even if another history of the same symbol has more bars or exactly the timeframe asked for. A request that names another instrument runs on that instrument's series, not on the one you have open: it loads its history and, if the timeframe you asked for is coarser than its bars, regroups them. Everything the Autopilot itself does uses it too — the Reality Check, that instrument's spread and commission, and the portfolio it combines. The summary before it starts names the file it will run on. The Portfolio screen does the same: each strategy there is simulated on its own history, with its own spread and commission, and one whose history is not loaded is left out and named rather than run on another instrument's candles.

“Already tried here”, over the series it will work on, is the verdict journal speaking: how many judgements that market has had, the best word and when the last one was. Every judge writes it by itself — the judge screen, the batch, the Autopilot, the Wheel — and Dr. Quant reads it too, so what was already rejected does not come back as a new idea. If nothing was judged there yet, no line shows.

The summary also says what the Reality Check will be able to compare the result against: which other instruments are loaded, and that the other clocks are built by regrouping this same series. If no second instrument is loaded, it says so before you start and says what it costs — with only one, nothing found in the run can score above 64, and “solid” starts at 75. Importing one takes seconds; finding out afterwards costs the run.

The style chips

Under the box sits a row of Style chips: any kind, and the seven families. A chip writes its word into your sentence, where you can see it, move it or delete it — the request is still one sentence. And a chip is lit because the sentence names that style, not because it was pressed, so deleting the word turns it off.

Tap several and they are joined in the sentence, and the search mixes those families, as ticking several chips does in Build › Evolve. Typed by hand it works the same way: the styles you list — “trend and breakouts”, “price action with divergence” — are all searched, and one you rule out stays out: “trend, not breakouts” searches trend alone.

Any kind removes every style word. A sentence that names no style runs the search in random mode: no family and no templates, over a short core pool — close, SMA, EMA, RSI and ATR.

What it does once it starts

1. Search — eight genetic searches, one after another, unless the sentence asks for another number ("10 searches"), each with its own random seed, over the families the sentence names. Each search settles on one to three ideas by chance, and eight different ones find what one alone misses. Half of them also train on the instruments the Reality Check will compare against, and half train on this one alone: on the S&P 500, training with the Dow is what found the strategies that hold on both; on bitcoin, training with ether held the search back.
2. Backtests — every strategy in the final generation of each search, measured over the whole series.
3. Select — the strongest of each search, five unless the sentence asks for another number, leaving out any that take the same trades as one already kept, from that search or another.
4. Reality Check — each of them, told how many rule sets all the searches tried between them: eight searches are a tournament eight times as large, and the judge asks for more evidence to match. It cannot know what you picked under How it judges, so the other-instruments check uses the loaded instruments with the most history.
5. Its sibling — the solid ones on the stretch the search never saw are judged once more with their sibling instrument as the subject, from the same date. The sibling is the loaded instrument whose daily moves were most like this one's before that date, with a correlation of at least 0.5, and the stop and the target travel scaled to its typical bar. Only those that stay solid there are saved on their own and combined; the list shows both scores. Without a sibling, nothing changes. On pure noise, 11 of 576 searches left some solid strategy, and none of them passed this.
6. Library — only the solid ones are saved, with their verdict — the one from the stretch the search never saw, when there is one. At the end it asks whether to keep any of the others: the buttons under the list save the fragile ones, all of them, or one by one, with their verdict and the same count of what the searches tried.
7. Portfolio — only the solid ones are combined. Since the stretch the search never saw is what chose them, the portfolio's curve over that stretch is not fully unseen, and the step says so. If none came out solid, no portfolio is built, and that is a result, not a failure.

Close to the bar — the ones that do not pass it but come close: 45 or more on the stretch the search never saw, or solid there and not with their sibling. They are neither saved nor combined on their own: the list marks them, and Send to quarantine saves one with its verdict and opens Quarantine with it already picked — the solid ones that pass have the button too, because quarantine is their next step. Quarantine costs nothing, and it is where they show whether they were worth it. The 45 is provisional: each record notes how its strategy came in, so the band can be moved with what each group does live.

On the brief, Also test the procedure adds a seventh step at the end: the same search, repeated on windows of the history, each traded on the stretch it did not see — the test that Build › Evolve calls Test the procedure. It takes longer than the search itself, and the brief says how much, so it stays off unless you tick it; its result is shown under the run. It judges how the strategies were found, not any one of them, and it changes no verdict.

It stays on this screen: the six steps play out here, and Stop ends the run at any point. When it finishes, See the portfolio — or See what was found, when no portfolio was built — takes you to the result.

Build › Manual

Visual strategy editor with AND/OR rule blocks — the *Manual* tab of **Build**, next to *Evolve* and *Explore*. It works on the history you pick in the strip at the top — the history and the candle — and the strategy takes its symbol and

timeframe from there. To bring in another, the strip's **Download another in Data** button takes you to Data. There is no synthetic sample: every verdict is worth exactly what the data behind it is worth.

The editor has two modes: **Guided** (pick a family + template, rules pre-filled) or **Custom / Alchemist** (free-form rule mixing). Includes the live redundancy checker. Under the name, the symbol and the timeframe, the **Reference series** card adds up to three markets the rules read without trading them — "long EURUSD only if the DXY is falling" —: they are picked from the library, and each condition then has a *Series* box to read it on one of them. **Money management** sits folded at the bottom: core risk (SL, TP, %risk, max positions, lot) — the stop and the target in **pips**, in **multiples of the ATR** or in **percent of the price**: in ATR, "twice the ATR" is the same idea on any instrument, which pips are not; the last two are measured when the entry is decided and stay fixed for the trade —, position sizing (fixed lot / % risk / Kelly) and advanced MM (trailing stop, break-even, max daily loss, max trades per day). Below it, **Trading hours and time exits**, all in UTC hours: open only when the signal bar starts inside the hours — 8 to 17, or 22 to 6 across midnight —, with four buttons for the sessions — Asia, Europe, Europe and America, America — set on the history's clock when it is known; the **Friday close**, which from that hour closes what is open and opens nothing until Monday — nor on Saturday or Sunday, which have bars on a market that never closes —; and close at a set hour, or after so many bars or minutes. And the **Account cut-off**: if the balance falls some percent from its peak, if so many losing trades come in a row, or if some percent is lost in one UTC week, nothing more is opened — in the backtest, to the end; in the exported robot, until someone resets it by hand: MetaTrader with `KillSwitchReset`, JForex with `killSwitchReset`, the Python bot with `KILL_SWITCH_RESET`, and the robot remembers it across restarts. Unlike the daily limits, it does not lift by itself. It is measured on the closed balance, and the report says when it tripped and why. Advanced MM and the clock are simulated by the backtest *and* written into the exported robot, so the verdict describes the robot you actually download. A bar that stays at the bottom of the screen holds **Backtest**, **Export…** and **Save to Library**. Both of the last two save the strategy and take you to the Library, where it is exported: **Export…** opens its download menu straight away, with six destinations — **MQL4**, **MQL5**, **Java (JForex)**, **C# (cTrader)**, **Pine Script** and a **Python** bot for Hyperliquid. See *Installing the robot on each platform* for what to do with the file.

The entry order

Above money management, **Entry order** says how it enters when the rule says yes: **at market**, on the close of the signal bar; with a **stop order**, with the move — above to buy —; or with a **limit order**, waiting for the pullback. The order rests at the signal bar's close or at **a level**: the highest or lowest of N bars, a moving average, bands, a channel, the previous day's high or low. It is the textbook breakout: *buy with a stop order at the highest high of the last 20 bars, and sell with another at the lowest low*. The **distance** goes beyond the level, and the **expiry** is how many more bars it waits after the next one. If the level lies on the wrong side of the price — a buy stop below the close — no order is placed, as a broker would refuse it. The genetic search writes them too — with the Breakout family, on a level as well —, and under *Fitness & constraints* you choose which it may use: at market, stop, limit, or any mix as long as one is left. The limit is the one the backtest treats most generously: it counts it filled as soon as the price touches it, and in a real market a touch is not always a fill. All six exported robots place them as the backtest does.

Strategy families — seven. Each one is a pool of indicators, a short list of periods and a set of templates, and the genetic search draws from those when you tick it. MACD, Bollinger and Keltner are the exception to the periods: each packs two or three numbers into one — fast and slow EMA (and the signal); period and deviation; EMA and ATR — so the search draws each number within its own range instead: MACD fast 5–20 with the slow at least twice the fast and at most 60, signal 5–15; Bollinger 10–50 bars and 1.0–3.0 deviations; Keltner EMA 10–50 and ATR 5–20. Rules show them unpacked: `Bollinger Upper(20, 2.0)` is 20 bars and 2.0 deviations, and it is what the Bollinger templates use. There is a second exception, for the opposite reason: the Parabolic SAR, the Awesome and Accelerator oscillators and the four Ichimoku lines **have no period at all**. Theirs are part of what they are — the SAR has an acceleration step of 0.02 and a ceiling of 0.2, the Awesome oscillator is the 5-bar average minus the 34-bar one, Ichimoku is 9, 26 and 52 — so the search draws none, the builder shows no period box, and rules

write them without a number: `Awesome Osc`, not `Awesome Osc(14)`. A saved strategy carrying that number goes on trading exactly as before — the engine and all six robots already ignored it; what stops is showing it.

- **Trend** — enters in the direction of an established move: moving-average crossovers and alignment, with or without an ADX filter. Pool: SMA, EMA, SMMA, LWMA, ADX, MACD with its signal and histogram, Parabolic SAR, the four Ichimoku lines.
- **Breakout** — price escaping a recent range or level. Pool: the bar's high, low and close, the previous close, the highest and lowest of N bars (the Donchian channel), the previous day's high, low, open and close, Bollinger and Keltner bands, Bollinger width, ATR, standard deviation, inside and outside bars.
- **Reversion** — bets that a stretched price snaps back. Pool: RSI, Stochastic, CCI, Williams %R, DeMarker, MFI, RVI, Bollinger bands and their middle line, SMA and EMA, engulfing and pin-bar patterns. Two of its templates are for sideways markets and say so: *Range Fade (Bollinger + ADX gate)* buys the lower band and sells the upper one, and *Range Fade (Stochastic + ADX gate)* fades Stochastic coming back from 20 and 80 — both only while ADX(14) is below 20. The gate is half of each: fading a level pays in a sideways market and loses on every bar of a trend, and ADX under 20 is what sideways looks like as a number.
- **Price Action** — candle structure and no oscillators. Pool: open, high, low, close, body, range, the previous close, the previous day's high, low and close, and the candle patterns — engulfing, pin bar, inside and outside bar, doji. The patterns are proportions of the candle's own range, so the same rule works on a currency pair, gold or a perpetual.
- **Momentum** — the speed of price change. Pool: ROC, MACD with its signal and histogram, the Awesome and Accelerator oscillators, TRIX, RSI, EMA.
- **Volatility** — expansion and contraction. Pool: ATR, standard deviation, Bollinger width, Keltner bands, Chaikin volatility, range and body.
- **Divergence** — price and its oscillator disagree. Pool: close, high and low, and three oscillators — RSI, MACD and CCI. Pick *one* of them: two oscillators disagreeing with each other say nothing about price. Templates: RSI (14, 5 bars back), CCI (14, 13 bars back), MACD (12/26, 8 bars back), and a *hidden* divergence on RSI (14, 8 bars back), which is the mirror of the classic and means the opposite — a continuation, not a reversal.

> **A divergence is always whole.** One condition is half of one. A whole divergence needs, in the same entry block and joined with AND: a condition that compares price — close, open, high or low — with itself some bars back; a condition that compares the oscillator with itself *the same number of bars back*; and the two pointing opposite ways — price higher and the oscillator lower, or the mirror. With OR the block fires on either half alone; with two different distances there is no disagreement, only two sentences about two stretches of market; with the same direction it is momentum confirmed, the opposite of what the family promises. Anything else in the block — a level filter such as "and RSI came from oversold" — is fine. The genetic search never hands over half a divergence: whatever it breeds with half of one, it completes.

> **Why there is no eighth family.** The old taxonomy had one called Range / Carry / Grid. Range trading is what Reversion already does. A grid needs several positions open at once, and the engine holds one — it opens only when flat — so offering it would promise something that does not run. Carry is a cost, the funding rate, not a signal: there is no rule to write about it. What was worth keeping is the regime filter, and it is in the two range templates above.

Strategies somebody else wrote down

Under the families and templates of **Guided** mode, a shelf of sixteen more starting points. Sixteen entries taken from real sources — books (Murphy, Elder, Schwager, Connors), web articles, videos and papers — each with its author, its citation, its year, the instrument and timeframe it was written for, what its edge is supposed to be, its

own list of warnings, and what BotFactoryFX's own Reality Check said of it on the S&P 500 daily and on EURUSD daily and four-hourly. They are sorted by that verdict: of the fifteen the shelf started with, none came out Solid — the best, Murphy's ADX filter on the S&P 500, 71 Fragile —, and the 2-period RSI pullback of Connors and Alvarez, added after, did: 88 on the S&P 500 daily. It is what the judge said then on that history, not a promise: judge it again on yours. Search them by title, author, citation or keyword, filter by source or by family, and sort also by structure — a checklist of stop, target and risk, not a backtest —, title or source. Only families that actually have entries get a filter chip, so no chip can empty the list.

Use these rules puts the entry's rules, exit and risk into the strategy you are editing, keeping its symbol, its timeframe and its starting balance, and the screen stays where it is: the rules land in the editor below. They are new rules, so any count of variants the previous ones carried is dropped rather than inherited — see *Reality Check*.

Describe it in words

Above the reference series, a box to write the idea the way you would say it — “buy when the 14-period RSI crosses above 30 and the price is above the 200-period average; stop at 2 ATR and target at 4” — and **Write it with the AI**. Dr. Quant, with your key, builds it with the program's indicators and operators and nothing else, and the rules land in the editor below, on the same strategy — it can be undone — keeping its symbol, timeframe and balance. Whatever the idea asks for that the program does not have — another indicator, a news filter, a second instrument — it builds as close as it can and tells you underneath, one sentence per thing; never silently. It is read like an imported JSON: anything outside the program's model does not get in. It is not tested on its own: you read it and press **Backtest**. And **every version you test counts as an attempt for the judge** — with a backtest or in the Reality Check, once per version; tweaking a number by hand does not add one — because asking an AI for version after version until one comes out well is searching, even by hand. A single tested version carries no count, like a strategy you wrote yourself. It is not in the browser edition.

> **The 0–100 score says nothing about money.** It checks that the risk is not reckless: that there is a stop, that there is a target, that the reward-to-risk is sane, that the risk per trade is conservative, that nothing is a martingale or a grid without a stop. It is a structural sanity check, which is why the curated entries nearly all score high — they were written by people who put stops on things. Whether any of them has an edge on your history is what *Explore* and then the Reality Check are for.

> **Nothing on this shelf is ordered by what it earned, on purpose.** Sorting proposals by their result on the history you happen to have loaded is a machine for overfitting: it would hand you whichever one fits your data best, which is the definition of the problem the Reality Check exists to catch. And nothing on it was invented from nothing — combining indicators at random and scoring them is what the genetic search does, far better and with its samples kept apart.

Each condition has the form `indicator OP operand`. Operators: >, <, ≥, ≤, =, crosses above, crosses below — and is rising, is falling, turns up, turns down, which read only the left side.

Besides the indicators, a condition can read the clock: the **Hour** the bar opens at, 0 to 23, and the **Day of the week**, 1 Monday to 7 Sunday, on the history's clock. “The hour is at or above 8” and “the hour is below 17” make a window; “the day is at or below 4”, Monday to Thursday. A window that crosses midnight does not fit in an “AND” block —it is two conditions joined by “OR”—: that one is what the money-management hours are for. The six exported robots read them as the backtest does.

Live redundancy check. The Build tab scans for overlapping conditions: duplicate indicators, near-duplicate periods (e.g. SMA(20) and SMA(22)), self-comparisons (SMA(20) > SMA(20), trivially false), or mixing RSI + ROC (both momentum oscillators). Flagged conditions are highlighted in amber.

Moving a number by hand

This is where the numbers of a strategy are written, and the editor covers all of them: an **indicator period** goes in the rule that uses it, right beside the indicator, and the **stop and the target** go in *Money management*, folded away at the foot of Build. Change one and press Backtest.

> **Three indicators keep several numbers in one.** MACD, Bollinger and Keltner are written packed — 1226 is a MACD of 12 and 26; 2020 is Bollinger over 20 bars at 2.0 deviations — and the editor has a single box for that packed number, with its parts spelled out underneath. The Reality Check names the parts separately («MACD(12, 26) · fast EMA»), so to move the one it points at you compose the packed number yourself: fast × 100 + slow, period × 100 + deviation × 10.

Nothing will tell you which value to write, and that is deliberate. Until 25-09-2026 there was an *Optimize* screen that swept a number and offered the middle of the band it held up over; it was withdrawn because that band is measured on the same slice the strategy was built on, and a value chosen there is a value chosen with hindsight. What you can still find out is how much a number you chose holds up on history it never saw: judge the strategy, and the **parameter sensitivity** row breaks that down number by number.

Build › Evolve — the genetic search

Genetic evolution of strategies. Each **individual** is a complete strategy. The population competes for a **fitness** and evolves by tournament selection, crossover, mutation and elitism. It works on the history you pick under *Instrument* when setting up the search; a strip at the top names it, with **Download another in Data** beside it. Three tabs:

- **Search** — the genetic search itself, with **Money management** at the bottom.
- **Wheel** — searches, judges and goes round again on its own. See below.
- **Population** — every strategy the searches produced, with their results.

Workflow:

1. Load at least 300 bars under **Data**. Any reference series its rules may read is added here, in **Search** — and in the **Wheel** —: the *Reference series* card, under the history, picks them from the library. Under 1,500 it warns you that the history is short.
2. On **Search**, choose how hard to look — **Quick look**, **Balanced**, **Thorough** or **Deep** — the stretch of history the search may see, the direction, if you only trade one way, and the **Families** it may write (below).
3. Check the **Worked out for you** box: the stop and target ranges, the round-turn cost and how far the smallest target clears it, the minimum number of trades, how many backtests the search will run and roughly how long that takes.
4. Click **Run the search**.
5. Import the best result with **Import to Builder**. It arrives in Build carrying the number of rule sets the search tried, which is what the Reality Check is told when it judges it. Then set risk in the **Money management** section at the bottom of *Search*. The code comes out of the **Library**, once it is saved.

Avoid my portfolio, next to *Seed from current*: when on, the strategies of your latest saved portfolio are run before the search starts, each on its own history, and every candidate is compared with each of them the way the *Portfolio* screen compares them. One that correlates 0.7 or more with any of them loses: it drops below every other, and the more alike, the lower. It is not thrown out unseen: it is measured and counts as tried, so it raises the judge's bar like the rest. When the search ends it says how many dropped for that. With no saved portfolio, or no history for any of its strategies, it does not start and says so.

Hour and day filters, in *Fitness & constraints*, next to *Stop and target in ATR*: when on, about one in eight of the conditions the search draws is a clock one —"the bar opens at 8 or later", "Monday to Thursday"—, on the hour and the day of the week of the bar in the history's clock (see *Build › Manual*). Never "at 13:00 sharp", which an H4 chart never meets. It comes off, like StrategyQuant X's hour and day blocks: a clock filter is the cheapest way to fit the noise of a history —"on Tuesdays at 14:00 it won"—, and each one it tries is one more variant the judge counts.

> **Overfitting.** Win rates above 85% on few trades are usually noise. The search already demands a minimum number of trades worked out from your history — between 20 and 120 — and holds the last stretch back unseen. Judge what it finds in the Reality Check rather than trusting its fitness: fitness is what the search optimised, so the top of the list is the result most likely to be overfitted.

What the judge's stretch can tell apart. Under the stretches, in *What the search may see*, the program says how many trades the judge's stretch needs to tell a profit factor of 1.2 from one of 1.0 — about 750, with winners and losers the same size — and one of 1.5 — about 150 — and how often a strategy would have to trade to get there with the stretch you placed. Below that, a modest edge and a coincidence look too alike for any judge to separate: better to know before spending the search.

Test the procedure

At the foot of *Search*, before money management. The Reality Check's walk-forward re-tunes the stop and the target of *one* strategy; this one repeats the **whole search**, with the settings above, once per window: it searches on one stretch of the history and trades what it found on the next stretch, which the search never saw. It is what you would do trading for real — search, trade for a while, search again — done on the past. **Rolling**: the search stretch always has the same length, four times the one outside, and moves forward. **Anchored**: it always starts at the beginning and grows. Three to eight windows; five by default.

For each window it says what the strategy found made in sample and out of sample, per year, and the **efficiency**: out of sample over in sample. The procedure **holds** if the median efficiency reaches 50 %, if at least six windows in ten make money out of sample, and if all the out-of-sample trades together give a PF above 1. With fewer than three windows trading out of sample it says nothing. It judges the settings, not a strategy, and saves nothing; it costs as many searches as windows.

The same test is offered at the end of the Autopilot, ticked on its brief, and under the Reality Check's verdict for a strategy that came out of a search, which repeats the search that found it with the settings it ran with.

The families, and the cocktail

The seven families — Trend, Breakout, Reversion, Price Action, Momentum, Volatility and Divergence, described under *Build › Manual* — are a row of chips on the *Search* tab, always in view. They are the strongest thing you say there: the search takes its indicators, its periods and the templates it seeds about half of its first generation from out of the families you tick. Tick one and the line below it describes the family. Untick all of them and the search cannot start: with no family there is nothing to draw from.

Tick several and they are mixed — anything from one to all seven. The indicator pools, the period pools and the templates are joined, and the seeding draws a family first and then one of its templates, so each ticked family gets the same share of the seeded half whatever the number of templates written for it. That is how a hybrid comes

out: a breakout with a trend filter is a strategy neither family can write alone, because the previous day's levels are not in the trend pool and ADX is not in the breakout one. Mixing makes hybrids possible, not compulsory — the search keeps whatever scores best, and a strategy that uses only one family's indicators is as welcome as one that mixes them. The line under the chips says how much the pool grew, and what that costs: a wider pool tries more distinct variants, and the Reality Check asks for more out-of-sample evidence the more candidates were tried.

A divergence is always whole, alone or mixed. Whenever a strategy starts to write one, the search completes it (see *Divergence* under *Build › Manual*) instead of letting half of one through: half a divergence trades more often than a whole one, and without this the search kept the half. In a mix, a strategy that carries nothing of the divergence family is left exactly as it is. Ticking Divergence also raises *Max conditions / block* to two if it was set to one, because a whole divergence needs two.

Deep effort widens the indicator pool beyond the ticked families' own; the seeding still comes from their templates. In *Advanced settings* the same chips sit next to a **Guided** / **Random** switch: Random is not restricted to any family and seeds from no templates.

> **Before the chips, the family did not reach the search.** In earlier versions the family could only be chosen inside *Advanced settings*, and a search started from the setup panel always ran the default, Trend, whatever had been picked there. A strategy found that way came out of the Trend pool.

The Wheel

The third tab. It does the loop you would otherwise do by hand — search, read the verdict, change something, search again. Each round takes the next dataset you marked and the strategy families you ticked — one at a time or all together, see below; tick none and it uses all seven — runs a genetic search, and judges the best result with the thirteen Reality Check checks.

Reference series come along too. The same *Reference series* card from *Search* sits in the wheel's tab, and every round searches and judges with whatever series are registered when it starts: the rules it finds may read them, and the judge treats them as the Reality Check does — when a strategy truly depends on another series, the other-instruments check becomes unavailable, its weight spread over the rest.

One family per round, or all of them together. The switch next to the family chips decides. Rotating — the default — makes each round a narrow search, and you know which family whatever it found came out of. Mixing runs the whole cocktail every round: each round is wider and can write rules no single family can, because the indicator pools are joined. What changes from round to round is then the dataset and the seed, not the shape of the strategy. Neither is better; the wider search tries more distinct variants, and the Reality Check asks for more out-of-sample evidence the more were tried.

Rotating, dataset and family move on together, one step each per round, so the pair repeats after the lowest common multiple of the two counts — datasets × families only when they share no divisor. With all seven ticked they share none unless you mark a multiple of seven datasets — seven is prime — so every dataset eventually meets every family; it just takes datasets × 7 rounds to get there, which with three or more datasets is past the default limit of 20. Mark seven datasets, or fourteen, and each one only ever sees one family. Tick fewer families and the cycle is however many you ticked, and that number can share a divisor with the number of datasets: four families with two datasets repeats after four rounds, and each dataset only ever sees two of them. Every family you ticked is still tried; what is not guaranteed is trying each one on every instrument. Mark fewer datasets if you want that inside one run.

It stops at the first strategy judged *solid* — or *fragile* as well, if you untick **Only stop for "solid"**, which puts the finish line where noise can reach it. Otherwise it stops at the round limit (20 by default) or once an hour has passed. The hour is only checked before a new round starts, so the round in progress always finishes: on a large dataset or at *Deep* effort, that round alone can take far longer than the hour.

Every round's rule sets are added to one running total, and each round's verdict is given the total so far. The more rule sets have been tried, the more likely the best of them is best by luck, so the higher a strategy must score on the checks that look outside its own history before its score is allowed to stand. The total travels with the strategy: imported to Build from *Population*, or sent from there to the Reality Check, a Wheel candidate carries what the *whole* Wheel tried — more than its own round's verdict was told, if rounds came after it, because picking it out of the whole harvest is picking it out of everything the Wheel tried. So judging it again does not loosen that demand; if anything it tightens it. Saved to the Library straight from Population, it takes its count along too, kept with it.

Everything it judges goes to *Population*, marked *From the Wheel*, whatever its verdict; strategies that take exactly the same trades share one tab. They are kept only while the app is open, and nothing is saved to the Library by itself. **Clear** on the Wheel tab, or starting a new run, removes them.

> **Stopping without a find is a result — when a find was possible.** It means no round produced a strategy the Reality Check calls *solid* (or *fragile*, if you unticked the box). The summary gives the best score reached, and anything rated *fragile* is still in Population. If not a single rule set was tried — every marked dataset too short, too flat, too large, or with costs larger than its typical move — it says so, and that says nothing about the market.

> **Mark at least two different symbols.** It rotates over the datasets you mark, not over timeframes, and the check on other instruments needs a different symbol: two timeframes of the same instrument count as one market. With a single symbol marked, that check scores zero, nothing can be judged *solid*, and with the box ticked the Wheel cannot find anything — so it will not tell you anything about the market either.

Build › Explore

The third tab of **Build**, and the only one of the three that does not end in a strategy. It is where an idea gets looked at before anyone commits to it: a backtest that re-runs as you move a number.

Lab — what changes when a number changes

The strategy open in Build, drawn over your loaded history: the candles with every entry and exit marked, the equity curve under them, the whole trade list, and the metrics beside it. Moving any control on the right re-runs the backtest over the whole series — at the instrument's own spread, commission and slippage, and with its instrument card — so the figures here are the same figures the rest of the program would give for the same strategy.

The rules stay where they are. What the controls move is everything around them: the candle the strategy is read on, the stop and the target with the reward-to-risk worked out beside them, how much to bet (fixed lot, percent of the account, or Kelly), the risk per trade, the lot, the maximum open positions, the starting balance, the trailing stop in pips or in ATRs, break-even, and the two daily caps. To change a condition you need *Manual*, and **Send to Manual** takes the strategy there with whatever you settled on.

It re-runs about a tenth of a second after you stop moving a control. **Auto** turns that off and **Run** does it on demand. The backtest runs off the main thread, so dragging a slider does not freeze the screen, and a slow pass from three changes ago cannot land on top of the numbers you are looking at now.

> **The candle is a control too, and it is the one with the most effect of all.** A 20-bar EMA is twenty hours on H1 and twenty minutes on M1 — not the same strategy measured more finely, a different strategy.

What the Lab does not do

It does not sweep a number, it does not say whether one stands on a plateau, and nothing it shows is ever told to the judge. What is left here is the question, next to the dials where you move those same numbers by hand. A sweep used to live here, then moved to a screen of its own, and that screen was withdrawn on 25-09-2026: measuring the shape of a number over the training slice is not evidence of anything. The **Reality Check** asks the same question where it counts, over the held-back slice, and its parameter sensitivity row breaks the answer down number by number.

It saves nothing either. Nothing reaches the Library from here, nothing is judged, and a backtest is not a verdict — it is precisely the answer the Reality Check exists to doubt.

Moving around the chart

The Lab's chart is the same one that appears in **Live** and in **Quarantine**, and it works the same way in all three. It holds the whole series, which can be tens of thousands of bars. To travel through it:

- **Drag** the chart left or right.
- **Shift + wheel** scrolls through time. The wheel on its own zooms.
- The **buttons** above the chart: a screen back, a screen forward, jump to the oldest bar, jump to the latest, zoom in and out, and fit the whole series on screen.
- **Arrow keys** once you have clicked the chart — ← and → scroll, Page Up and Page Down move a full screen, Home and End go to the ends, + and − zoom.

The dates on screen are printed next to the buttons, alongside the timeframe, so you always know where you are looking.

Candles, bars, line or area

Japanese candlesticks are the default and the recommended view — they are the only one of the four that shows the open, the close and both extremes of every bar at once. The other three are one click away above the chart.

JUDGING

Reality Check

The screen this product is named after. It does not ask whether a strategy made money — the backtest already answered that. It asks **whether that answer is real**. Four tabs at the top: one strategy, several at once, one somebody sent you, and how the judge weighs things.

A strategy of mine

Thirteen checks against the strategy picked in the bar above — the one open in Build, or any one from the Library, so a search that left forty strategies saved can be judged without opening each one first. They run off the main thread because thirteen checks over a real dataset are well over a hundred backtests — around 150 at the default settings.

What they ask: how much the result degrades out of sample; whether it beats that same unseen history with the direction of each bar drawn at random — the same size and the same place, going up or down at random around its underlying trend: the runs of volatility stay and the only thing removed is the direction —; whether re-tuning and walking forward survives; whether the same rules work on **other instruments** — the ones you pick under *How it judges*, or else the ones loaded with the most history once regrouped onto its timeframe, with its stops in pips scaled to each one's typical bar, because a pip is a different size on each; whether they work on **other timeframes**, slower and faster; how sensitive it is to its own parameters; whether it holds up with that stretch begun elsewhere and its prices nudged — forty times, counting its worst tenth —; Monte Carlo resampling — three ways: one trade at a time, in blocks of consecutive trades, which keeps the streaks, and skipping one in ten; the worst counts —, and whether it would still win without its best 5 % of trades, because if not it lives off a handful and its score is halved; whether there are enough trades to say anything at all; consistency across periods; what the drawdown looks like counting open positions and not just closed ones; whether the bar resolution is hiding anything; and how much broker cost the edge can absorb before it disappears.

> **Nine of the thirteen are meant to measure what a search could not have fitted**, because a check run on the same candles a search fitted to cannot tell a real edge from a fitted one, however clever the test. Seven run on the judge's stretch: the end of the history that neither the search nor the selection saw. The other two leave that history altogether: other instruments run on other files — counting only the trades that open in the judge's stretch, because two markets that move together share the years a strategy was fitted on — and other timeframes cut the prices into different bars. Period consistency, cost tolerance and bar resolution look at the whole file on its own bars — between the three, nine points of the hundred. Drawdown looks at it too, but since 28-09-2026 it does not count toward the score: it measures how risk is handled, not whether there is an edge, and it is shown as a warning.

> **The search's window has three stretches, and the judge only looks at the last one**. By default the search learns on the first 60%, Population and the wheel choose on the next 20% — the selection — and nobody sees the last 20% until the judge does. The autopilot no longer chooses on the selection: it judges all its finalists on the last stretch, and the selection is left unlooked-at, as a gap between what the search learned and what is judged. That changed on 17-10-2026, after measuring in the duel with StrategyQuant that this gate chose worse than average and threw out exactly the ones the judge rated solid. A strategy that comes out of a search carries where it must be judged from, and the judge cuts there, whether or not you move its slider. What the judge cannot know is what you did outside: if you found the strategy on another file or tuned its rules by hand against the whole chart, it has already seen that stretch, and those eight checks are grading work it could copy. Judge a strategy on history it was not built on.

How far the sensitivity check moves a number has a ceiling, and for the stop and the target it comes from the instrument, not from a fixed number of pips: 4 % of the series' own price, said in that instrument's pips, or the old 1,000 and 2,000 pips when those are larger. On currency pairs it never bites, because a pair's whole price is ten to twenty thousand pips; on gold or a perpetual it is the difference between trying neighbours around the stop the strategy has and trying them around a clipped one. A strategy whose stop already passes both ceilings brings its own. The same ceilings bound the walk-forward grid, so both checks move a number the same distance.

Which other timeframes. By default, the usual step down and the usual step up on the usual ladder — M1, M5, M15, H1, H4, D1, W1: from H1, M15 and H4; from H4, H1 and D1; from M15, M5 and H1. In the *How it judges* tab, next to the contrast instruments, you can pick others, M30 and H2 included. Pick them before judging: whatever you pick is written into the report and the saved verdict, and the card says which clocks it was tried on — dropping clocks until the strategy passes is fitting the verdict, and this way it shows. If the rules read another series, a clock that would change how that filter lines up is not tried, and it says so.

Faster timeframes need finer data. Slower ones are rebuilt from the bars you are looking at: from H1 comes H4. Faster ones cannot be invented — an hour of bars does not contain its quarter-hours — so they are only tried when the file you imported is finer and the timeframe asked for comes out of it in whole bars: import M5 and view H1, and the check looks at M15. In the settings, the ones that cannot be built are greyed out; and when one you picked could not be built, the check says so and says what to import

The `verdict`

A score out of 100 and one word: **Solid** from 75, **Fragile** from 45, **Weak** from 35, **Not credible** below that — or **Not enough evidence**, with no score at all, when there are too few trades to judge. Two things come with it and matter as much as the number: the checks that could not run, marked grey in the list, each saying why; and a box of reasons to read the figures as indicative rather than exact. A passing score with three checks unavailable is not the same thing as a passing score.

The score is the weighted average of the checks that ran, and then it can be **capped**. When it is, the report says so and gives the score it would otherwise have had.

- **Nothing tested outside this history.** Only two checks ask whether the edge exists outside the series it was built on: other instruments and other timeframes. Everything else re-measures that one series, and a series can be fitted. With neither of the two, the verdict cannot go above *weak*; with one, not above *fragile*. *Solid* needs both.
- **Too little of the score rests on unseen history.** When less than 70% of the weight that counted comes from the nine checks that measure on history the strategy never saw, the score cannot go above 70; below 55%, not above 50.
- **The search that found it was large.** The judge is told how many variants were tried to get to the strategy, and the more there were, the more likely the best is best by luck, so the higher it must score on the checks that look outside its own history. The Wheel and the Autopilot tell it for their own verdicts. On this screen it is told for the strategy open in Build when that came out of a genetic search — imported from Evolve or from *Population*, where a Wheel candidate brings what the whole Wheel tried; for a strategy picked from the Library when its entry and exit rules, its risk settings and its management are the same as those of the strategy open in Build that carries a count — saving it does not wash the count off; and, on *Several at once*, for every strategy sent from Population. A line above the verdict says how many variants the judge was told about. The count is also **kept with the strategy**: when it is saved to the Library — from Build or from Population — and when a verdict that carried it is saved. It only goes up, so judging it again another day, or opening it in Build, does not lose it: the judge uses the larger of the session's and the kept one. And picking in the Library with a **filter on figures** set is charged too: the one picked carries the sum of the counts of every strategy the filter compared. A strategy written by hand, with none of that, carries none, and this cap does not apply to it. One the AI wrote

from *Describe it in words*, in Manual, carries one for every version you tested — with a backtest or here — and the line above the verdict says so: searching by hand with an AI is still searching.

The count always comes from a genetic search, and the report says so: that many *different rule sets* were tried, and this was the best of them. Until 25-09-2026 it could also come from a sweep in an *Optimize* screen, which measured that many variants of the same numbers; that screen was withdrawn, so there is one source and one sentence. Whenever there is a count, the report also says what a search of that size scores on prices built to have no edge in them, so the number can be read against luck.

Two statistical figures come with the count; they inform and do not score. The **deflated Sharpe** is the probability that the strategy's Sharpe, on the stretch where it was chosen, beats what luck would give the best of that many trials. It tells noise from an edge well, but it only looks at that stretch: an edge that stopped working afterwards, or a flaw in the data, scores as high as a real one. The **probability of backtest overfitting** (PBO) is measured by the search: it splits its history into two halves every possible way and counts how often the best of one half finished below the median in the other. Near 50 %, picking the best was picking at random; but a low figure is not an edge: on prices with none it comes out near 17 %, because with costs the rule sets that trade less lose less in every half, and the best stays among the best without earning anything.

The count lives in this session, not on disk. A strategy saved to the Library is still judged with it as long as its rules are the ones the count belongs to — saving does not wash it off. Edit it after saving and the saved copy stops matching, so that copy is judged without the count; the one open in Build keeps it either way. Closing the app loses it altogether.

Not loading a second instrument costs points. Until a second symbol is loaded on bars at least as fine as the strategy's — they are regrouped onto its timeframe, and 200 of them have to come out — the check on other instruments scores zero: it is a test that was not run, and leaving it out must not make a strategy look better than testing it would. Import one on the Data screen. Coarser bars are no use: finer ones regroup into coarser ones, and the other way round there is nothing to regroup.

Rules that read another series. A condition can read a second series — "only buy on H1 when the daily is rising". Such a strategy cannot be moved to another instrument as it is: doing so means deciding what that daily series becomes there, and that choice would change the strategy being judged. So for it that check is *unavailable* rather than zero, and loading more instruments does not change that. The other-timeframes check still runs, but skips any clock that would change how the filter lines up with what it filters; if the filter is on the same timeframe as the strategy, no clock qualifies and that check is unavailable too. The ceiling above still applies — without evidence from outside this instrument, *solid* is not available, whatever the reason. A filter that never changes a single trade on this history is not a filter: the other-instruments check judges the strategy as the rules without it, and the other-timeframes check skips no clock because of it — it still runs the strategy as written, because on a faster clock that same filter may start to decide.

> Nothing to divide by. Eight of the thirteen compare profit factors — gross profit over gross loss — so a stretch in which the strategy never lost a single trade leaves them with no denominator. That is not a perfect result: it almost always means a stop that stretch never reached, and a sample that has not yet met the risk it is taking. Such a check says not measurable instead of a figure, counts as no evidence either way, and its weight is spread over the ones that could be measured. It can still fail — when the nudged or re-tuned versions make no money, that answer is measured and stands. What it can never do is pass. The ceilings above still apply, so a strategy that has never lost cannot reach solid: without a measured comparison outside its own history, that word is not available. One row uses those same two words and does count: sample size. What it asks is whether the average result can be told apart from zero, so when every trade on the unseen slice returned the same amount, the spread that is missing is itself the answer — and that row scores zero. The list says so under it.

Below the verdict

Is this a strategy? The thirteen checks all assume that what was measured is a strategy. This asks whether it is: whether one trade holds the whole result up, whether one direction pays for the other, whether it is in the market almost all the time, whether nearly every trade leaves the same way, and whether it almost never loses — which over enough trades is far more often a mistake than an edge. It runs by itself, on the same bars and costs as the verdict, and says so when there is nothing to report.

Does it fit what you already have? Runs your most recently saved portfolio twice — without this strategy and with it, on the same money — and says whether the combination is better. Adding a strategy takes capital from the others, so it has to earn the share it takes. It compares return per unit of drawdown, refuses a strategy that loses money on its own (such a strategy can smooth a portfolio's curve and still cost the account), and calls a near-copy of something already in there a copy. It is two full portfolio runs, so it waits for you to press it.

Test the procedure that found it. A button, for strategies that came out of a search: it repeats that search —with the settings it ran with, which every strategy found by a search keeps from this version on— on windows of this history, each traded on the stretch it did not see. It is the same test as Test the procedure in Build › Evolve and the Autopilot's optional last step. It asks whether the way the strategy was found finds things that hold up, not whether this one does, so it informs and does not change the score. It costs several searches, which is why it waits to be asked. A strategy written by hand, or found before the program kept how, has no procedure to repeat, and the panel says so.

When it works and when it doesn't. Profit factor and result per trade in each market regime — low, medium or high volatility by the ATR(14) against the price; in a trend or not by the ADX(14), cut at 25; the session; the year — in the judge's two stretches: where the strategy was chosen, and the one the search never saw. Each trade's regime is read on the bar before the entry, the last one a robot can look at, and the volatility thirds come from the first stretch only, so the held-back stretch tunes nothing. If a regime lost in both stretches, with at least twelve trades in each and a loss that matters, it says so as a rule — "switch it off when the ADX(14) is below 25" — and what the out-of-sample result would have been without those trades. If every regime loses, it proposes none: then it is not the regime, it is the strategy. Sessions are in UTC when the history's clock is known — MetaTrader stamps bars in its server's time — and otherwise in six-hour blocks of the file's own hours. It does not move the score.

Out of sample, with its margin. Under the verdict's rows, the profit factor and the average result of the stretch the search never saw, each with the margin it falls in 90 % of the time when its trades are drawn again: a 1.34 over thirty trades fits anywhere between something that loses and something that makes a lot, and one over three thousand does not. And the sample's power: how often it would tell a profit factor of 1.2 from one of 1.0, and how many trades it would take to do it eight times in ten — with winners and losers the size of its own. With thirty

trades it is rarely above 15 %: that is why the verdict asks for so much evidence before it says "solid". It does not move the score.

> **Costs come from the instrument.** Spread, commission and slippage are recorded once on the Data screen, and the search, the backtest and this verdict all use those same numbers. A strategy that only survives at zero cost is not a strategy — change the costs there and run this again.

Several at once

A search returns strategies sorted by the fitness they were bred for, which puts the most overfitted one on top. This tab judges a whole shortlist and re-orders it by verdict instead. Send strategies here from *Build › Population* — the ones that pass your bar, or one at a time; each send replaces the list.

Judge all of them loads the contrast instruments once and runs the thirteen checks on every strategy, several at a time, with the same costs, instruments, weights and measurement settings as the single check. The result comes ordered best to worst: scored strategies by score, a tie going to the one with fewer checks that could not run; then the ones with not enough evidence; then any that failed outright. Open a row to read its full report. **Stop** keeps and ranks whatever had already finished.

Each one is judged with its own count. The winners of a genetic search are usually judged here, so this is where the count matters most: a strategy from Evolve carries the number of rule sets its search tried, and a Wheel candidate the number the whole Wheel tried. No strategy borrows its neighbour's figure. The header of the list says it — one number when they all share it; when they do not, the range, and how many carry no count at all.

> **Every strategy is judged on the data loaded now.** A strategy built on another symbol or timeframe is judged on these bars anyway; the tab says so above, with how many and which ones.

One from outside

Somebody sent you a robot, or a screenshot of a backtest that looks too good. Attach their file here: the **source** (.mq4, .mq5, .java, .pine) and/or the **report** their platform exported. You get what can be read off the code — symbol, timeframe, lot, stop and target where they are written as plain assignments, MQL inputs and `#property` lines, JForex `@Configurable` fields — and what the report claims, recomputed from its own trade list rather than taken from its summary page. The code is read by pattern, so "not declared" can mean "not found": Pine inputs, for one, are not read.

If you attached their report and trades could be read from it, press **Run mine on this history**: the strategy chosen in the bar above is measured over your loaded history at your broker's costs, and the two land in one table. With only the source code there is nothing to compare against, and no table appears.

> **The code is read, never executed.** BotFactoryFX cannot run MQL, Java or Pine. The two columns are two different questions — their platform on their history, yours on yours — and they only line up if both covered the same period and the same instrument. When they do not, read the columns and ignore the difference.

What their trades say. With their trade list, and without running their rules, what can be judged is judged: whether their figures add up to their own trades, how they come out redone with your money and your costs (or, if their file carries them, the costs of their test) and, when the loaded history is the same asset, whether their trades fit your candles: prices that never existed, hours with no market, entries too good to be true. On another asset's history they are not placed on your candles, because on another instrument's candles an honest report would look like a fraud. What a list alone cannot answer is said: out of sample, better than chance, the neighbours, other instruments… everything that runs the rules again.

Their rules, translated. A StrategyQuant X `.sqx` also carries the rules, and they are translated into an engine strategy *only when they match exactly*: the same indicator with the same formula, the same comparison, the same way in and out. A single piece that does not match and there is no translation, with every piece, where it is and why: judging a similar one would be judging another under this one's name. Few match today, and the reasons are given: SQX's ATR is Wilder's and the engine's is the plain average; SQX moves the trailing stop with each bar's open; its waiting orders are placed again with every signal; its Friday exit closes at the exact hour and minute. When they match, you see their rules in words and what runs differently in some cases, you can **run it on your history and compare** it trade by trade with theirs — with the clock offset between their data and yours, if there is one — and **judge it with the thirteen checks**. SQX does not write in the file how many strategies it generated to find this one: if you know, type the number, and the judge will take it into account and only use the candles after its test; without it, it judges it as the only one tried, which is the most lenient reading.

How it judges

The judge, opened for inspection: the thirteen checks, what each is worth, and what each one measures.

The shares. Once a report exists, move any of them and a second figure appears here, beside the factory score: what the report scores with *your* weighting. The next run also prints it on the report itself. It never changes the official score or its word, and that is deliberate — a *solid* bought by switching off the checks that fail you could be exported, or shown to someone, as the verdict. Audit the judge; do not buy it. Your figure is never capped and the official score may be, so a gap between the two can be a ceiling rather than your weighting.

What gets measured — the train/test split (0.7), the period windows (6), the draws against chance (50, and more when the judge's stretch is short) and the Monte Carlo resamples (1,000). These change the experiment itself, so they *do* change the verdict, and they take effect when you press **Measure again**.

Contrast against. Pick the instruments the other-instruments check uses. You know which ones belong together — gold with silver, EURUSD with GBPUSD — and the program cannot work that out from a list of files. Only instruments that can be put on the strategy's timeframe are offered — its own bars or finer ones, regrouped, with at least 200 left afterwards. Pick none and it uses the ones with the most history.

A set of shares and settings can be saved under a name and applied later. Saved sets are kept on this machine only.

A name of its own

When the judge rates a strategy **Solid**, a card under the report offers it a name of its own, taken from a theme that matches its family: rivers for trend, mountains for breakout, myths of return for mean reversion, physicists for momentum, winds for volatility, poets for price action, composers for divergence. The first one offered is the one whose birthday — or first ascent — is closest to the day it passed, and if it is that very day it says so. *Other three* brings more, you can write your own, or *No, thanks*. The name goes in front of the technical one, which stays below because it is the one that says what the strategy does: in the Library and its search, in Quarantine and on Home. No two strategies in your library share a name; when a theme runs out, the name carries the day it passed («Danube of 7 November»). A Fragile strategy keeps only its technical name.

The method

Everything in this program follows one method: **elimination**. A backtest almost always looks good, and a search that measures thousands of variants guarantees that some look wonderful by luck alone. So the job is never to collect winners — it is to destroy pretenders as early and as cheaply as possible, and to trust only what survived

everything that tried to kill it. When a search throws away 99 % of what it bred, that is not waste: that is the method working.

Eliminate, do not collect

Each stage exists to kill what the previous one let through. The search breeds and measures in bulk; the selection keeps a handful; the Reality Check tries to prove each survivor worthless; the Library records the verdict next to the strategy, so it cannot be quietly forgotten; the Quarantine watches, live, the distance between the promise and the account; and the Portfolio only combines what held up. The expensive mistake is rescuing a pretty equity curve from the stage that just condemned it: the curve was the exhibit, the verdict is the trial.

Evidence has a price, and the judge charges it: the more variants a search tried, the more its best one has to prove on history nobody saw — being the best of three thousand is, above all, a property of the number three thousand. That is why the attempt count travels with the strategy, does not reset by changing files, and is spent, never laundered.

The funnel: cheap doubts first

Order the doubts by what they cost to resolve. A backtest costs seconds. The judge's thirteen checks cost minutes. Your own attention costs hours. The Quarantine costs weeks of calendar — but no money. A live account is the only step that can cost money, so it goes last, and small. The funnel is the menu read top to bottom: measure in bulk, judge the shortlist, quarantine what passed, and only then risk anything. Never spend an expensive step on something a cheaper step could have killed — and never skip a cheap step because the curve looks obvious: obvious is exactly what luck looks like.

The signs of overfitting

Overfitting is reading the noise and calling it a rule. Its signs:

- **Too good for the game being played**: returns far beyond what that market pays the strategies that survive in it.
- **It collapses out of sample**: brilliant on the bars it was bred on, ordinary on the stretch nobody saw.
- **A sharp peak instead of a plateau**: move any parameter one step and the result falls apart — the number was tuned to the noise, not to a behaviour.
- **It only works in one place**: one instrument, one timeframe; the same rules anywhere else earn nothing.
- **It lives off a handful of trades**: remove its best 5 % and nothing is left.
- **It was the best of thousands**, and is asked to prove no more than the only one ever tried.
- **It was judged until it passed**: tweaking and re-judging on the same bars is fitting the verdict, not the strategy — the verdict journal exists so that this shows.

> **The judge can only guard what it saw.** Its reserved stretch protects against what the search did inside this program. If you tuned the rules by hand against the whole chart, or brought the strategy already fitted from somewhere else, that stretch is burned evidence: the checks are grading work the strategy could copy. Judge on history it was not built on, and never repeat a test on burned data.

None of this ever says a strategy is good. The method only says one thing with confidence: *this one failed*. What it cannot kill, you are allowed to risk — small, watched, and knowing that surviving the funnel is the beginning of trust, not the end of doubt.

Installing the robot on each platform

The download menu of a strategy in the **Library** produces a finished file for six destinations — **Export...** in Build saves the strategy and opens that menu for you. Getting it to run is different on each one, and most of the ways it fails are silent — the file compiles, nothing happens, and no error says why. Every trap below is one that actually stops people.

> **Before any of this: the robot is not the backtest.** What the backtest simulated, the file carries: the rules, read on closed bars only; the entry — at the market, or a STOP or LIMIT order left waiting for as many bars as the backtest waited; the stop and the target; break-even and the trailing stop; the exits by time; both daily caps, counted on the UTC day as the backtest counts them; the size — fixed lot, risk percent or Kelly; and one position at a time. Each file says at the top where its platform cannot do exactly what the backtest did. What no file can carry is everything the simulation had to assume: your broker's real spread at the moment of the fill, the slippage on a fast bar, partial fills, requotes, a widened spread over news, and a gap that jumps past a stop or a waiting order, which a broker fills beyond the level while the backtest filled it at the level. The code is also minimal on purpose — no news filter, no session filter. Run it on a demo account first, on every platform, without exception.

Export with the strategy's own history open. The file is written with the instrument card of the series open on the Data screen — the size of a pip, the units in a lot, the leverage the backtest capped each trade at — so every distance and every size in it means what it meant when it was measured. With another instrument open, the MetaTrader, JForex and Python files are written with that one's card; the Pine script notices, leaves it out and works its own out from the chart.

Stops in pips from a search. When the strategy came out of the Autopilot or a search and its stop and target are in pips, the download menu starts with **Its stops**: **Adapt them to the ATR**, recommended and picked to begin with, or **Keep them fixed**. Adapted, the file carries the stop as so many times the ATR(14) — the number that makes it exactly as wide as the judged one on the last candle the Reality Check judged — and from there it grows and shrinks with volatility instead of staying fixed while the market changes scale. Measured on eight markets, the strategies the Reality Check did not reject beat chance afterwards 75 % of the time instead of 63 %. The target and a trailing stop in pips are adapted the same way; break-even and the offset of a waiting order stay in pips. A strategy you set up by hand is not asked: the file keeps its pips.

MetaTrader 4 — the .mq4

1. In MetaTrader, **File → Open Data Folder**. This is the step people get wrong: the folder you want is not where the program is installed, and MetaTrader is the only thing that knows where it is.
2. Copy the file into `MQL4/Experts/`.
3. Press `F4` to open MetaEditor, open the file, and press `F7` to compile. It has to say *0 errors*; a warning is fine.
4. Back in the terminal, right-click in the Navigator (`Ctrl+N`) and choose **Refresh**. The robot appears under **Expert Advisors**.
5. Drag it onto a chart of the same symbol and timeframe it was built for. In the dialog, the **Common** tab has **Allow live trading** — tick it.
6. Turn on **AutoTrading** in the toolbar. It has to be green.

A **smiling face** in the top-right corner of the chart means it is running. A sad face means AutoTrading is off, or live trading was not allowed for that chart. MT4 runs **one robot per chart**: to trade two, open two charts.

MetaTrader 5 — the .mq5

Same six steps, with `MQL5/Experts/` and the toolbar button called **Algo Trading**. Three differences worth knowing:

- **If it compiles and never trades, read the Experts tab** in the Toolbox (`Ctrl+T`). MQL5 creates its indicator handles in `OnInit`, and if one fails the robot returns `INIT_FAILED` and stops before its first tick. That failure is written there and nowhere else.
- **Netting or hedging.** The robot holds one position at a time, as the backtest did, and knows it by its `MagicNumber`. On a netting account every trade on a symbol is one position, so another robot or a manual trade on the same symbol merges into this one's. Keep the symbol to this robot there, or use a hedging account. Your account type is in the terminal, under the account name.
- **Use the Strategy Tester** (`Ctrl+R`) before the live chart. It is the fastest way to see whether the robot trades at all, and it costs nothing.

MetaTrader 4 and 5 — the settings the file adds

- **`BacktestPipSize`** — when the file was written with an instrument card, every distance in pips (stop, target, entry order, break-even, trailing stop) is counted in the pip of the instrument that was measured, not in the digits your broker quotes the symbol with. Check that your broker quotes it at the same scale as the history you measured.
- **`BacktestLeverage`** — the size is never more than the account could margin at the leverage the backtest capped every trade at, or at the account's own if that is lower. The value of a pip comes from your broker, and the size is put in your broker's lot steps; below its minimum lot the trade is not taken, and the Experts log says why.
- **`ServerUtcOffsetHours`** — appears when the strategy has daily caps, exits by time, or reads the previous day's levels. The backtest counted those days and hours in UTC; MetaTrader stamps its bars with the server's clock. Leave it at 0 if the history you measured came from this broker's MetaTrader; set it to the server's offset from UTC — 2 or 3 at most brokers, changing with daylight saving — if that history was cut in UTC.
- **The server's clock** — with a closing hour, daily caps or the previous day's levels, the robot needs `ServerUtcOffsetHours`: how far its server runs ahead of UTC, minus how far the clock of the history you measured it on ran (zero if that history came from the same broker's MetaTrader). Outside the Strategy Tester the robot measures its server's offset on each new bar, writes it to the Experts log and raises an alert when the setting does not match the history's clock, which the file names at the top when it is known.
- **Restarts** — the last bar the robot decided on, and with daily caps the day it is counting, are kept in the terminal's global variables (`F3`), so a restart does not decide a bar twice or forget the day. Two copies of the robot on the same symbol and chart period need different `MagicNumber`s.

Dukascopy JForex — the .java

1. In the JForex platform, open the **Strategies** panel, choose **Open** and find the file. The desktop app can put it in your JForex's strategies folder first, on Linux and Windows: **Send it to the platform's folder → JForex**, in the Library's download menu.
2. Compile inside the platform. The class implements `IStrategy`, with the logic in `onBar`.
3. Right-click the strategy and **Start**. The `@Configurable` fields appear in the launch dialog: the instrument and the chart period, the stop and the target, the lot or the risk, and whichever of the entry order, the time exits and the management the strategy uses. The file says of the ones that belong to the measurement that changing them trades a strategy nobody measured. The pip, the lot size and the leverage of the measured instrument are written into the code, not offered as settings.

The JForex export refuses what JForex cannot trade as it was measured: sizing by risk or by Kelly outside currency pairs, and any market other than currency pairs, gold, silver, bitcoin and ether, which JForex counts in contracts of a size the file cannot know. Outside currency pairs it also needs the instrument card, because the pip and the lot cannot be told from the symbol: exported with no history open, the file carries only the refusal.

> **Java requires the file name and the class name to match.** Renaming `MyRobot.java` to `test.java` makes it stop compiling, with an error that talks about a public class rather than about the name. Rename the class too, or leave the file alone.

Run it on the **Historical Tester** first — it is built into the platform and uses Dukascopy's own tick data, which is a genuinely different opinion from the one you got here.

Run live or on a demo account, the strategy keeps a journal and a heartbeat in JForex's files folder, under `BotFactoryFX`, for *Live* to set it beside its backtest decision by decision; nothing is written in the Historical Tester. Run it on your own computer: a strategy run on Dukascopy's remote servers writes them there.

cTrader — the C# cBot

1. In cTrader, open **Algo** (**Automate** in older versions) and create a new cBot. The name you give it is the one you will see in the list.
2. Delete everything the template brings and paste the whole file.
3. Press **Build**. It has to finish without errors.
4. Add an instance on the symbol and the timeframe the strategy was measured on, check its settings and start it.

It computes the indicators itself. It does not use cTrader's indicator library: each one is ported from the BotFactoryFX engine with the same arithmetic, so every rule reads the number the backtest read. It reads only closed bars and runs in UTC: the hour, the weekday and the previous day's levels are UTC ones, as in the backtest.

The settings come in groups — size, stops, entry, day, clock, account cut-off and other series —, only the ones the strategy uses. The pip and the lot of the measured instrument are written in: if your broker's lot is a different number of units, it does not trade and says so in the log (`Check the instrument` turns that off knowingly). On a timeframe other than the measured one it stops at start. It knows its trades by their **Label**, so two strategies on the same account leave each other alone.

What it counts survives a restart. Its equity —the balance when it started plus what its own trades make, as the backtest counts it—, the day's limits and the account cut-off are kept in the instance's local storage, and when it starts again it counts the trades that closed while it was off. **Start over** forgets it and starts from the balance. In cTrader's backtester it always starts clean.

> **The account cut-off does not lift by itself.** Once it trips the cBot opens nothing more —what is open runs to its stop, its target or its exit— until you set **Start over** and start it again. That is on purpose: it is what the backtest does.

Run it on **cTrader's backtesting** with tick data first: those are your broker's prices, a genuine second opinion. BotFactoryFX cannot read cTrader accounts yet —there is no connector—, so *Quarantine* and *Live* do not follow it.

TradingView — the Pine script

1. Open a chart of the right symbol, and set the timeframe to the one the strategy was built on.
2. Open the **Pine Editor** in the bottom panel, then **Open → New blank strategy**.

3. Select everything in the editor and paste the script over it. Save it with a name.
4. **Add to chart**. The **Strategy Tester** tab at the bottom then runs it.

> **The script trades whatever chart it sits on.** On the wrong pair or the wrong timeframe it still runs, still reports, and describes a strategy nobody built. This is the single most common way a Pine result turns out to be meaningless.

Two more things that make the numbers disagree with the ones here. **The script sets no commission, slippage, spread or funding**, and says why at the top: the export does not carry the costs the backtest was measured with, and any figure written there would be made up. So its results differ from the backtest's by those costs. Setting *Commission* and *Slippage* in the script's *Properties* to what your broker charges brings the totals closer, not the trades: TradingView quotes one price, so no setting moves the levels by a spread, and it has no setting for funding. And TradingView's feed for a symbol is not your broker's — different sessions, different closes, different gaps.

What the script does carry: the entry at the close of the signal bar, as the backtest entered — its market orders fill at that close; a STOP or LIMIT entry left waiting and cancelled after the same number of bars; both daily caps, counted on the UTC day; and the pip, the lot and the leverage of the instrument you measured, when the history open at export is the strategy's own — otherwise it works them out from the chart, and the script says how.

TradingView does not place orders. Nothing you add to a chart trades an account. To act on it you need an alert, a webhook and a bridge that turns the alert into an order at a broker — three more pieces, none of them supplied here.

Hyperliquid — the Python bot

The only destination that connects to a venue by itself, and the only one where a mistake costs money without a platform in between. It needs **Python 3.8 or newer** and nothing else to run dry.

1. **Run it dry.** `python3 yourbot.py`. It fetches real bars, prints the order it would have sent, and sends nothing. No key, no install, no account.
2. **Read it.** The indicators are inside the file, ported from this engine — that is the whole point of this destination. What it computes is what the backtest computed.
3. **Leave it dry for a few days** and compare its signals with the backtest. This is the step everyone skips and the only one that catches a mismatch before it is expensive.
4. **Create an API wallet** in Hyperliquid, under **API**. It can trade but **cannot withdraw funds**. That property is the difference between an expensive mistake and a total one — do not use your account key.
5. **Export the credentials** and install the SDK:
   `export HL_ACCOUNT_ADDRESS=0x…`
   `export HL_SECRET_KEY=0x…`
   `pip install hyperliquid-python-sdk eth-account`
6. `BFX_LIVE=1 python3 yourbot.py` trades on the **test network**, where the money is not real. Adding `BFX_MAINNET=1` is what puts real money behind it.

It manages the trade the way the backtest did: the stop and the target go to the exchange **with the entry**, as reduce-only trigger orders, so they fill on the tick that reaches them and they still fill if this program dies. A STOP or LIMIT entry rests at the exchange with its stop and target attached, and is cancelled after as many bars as the backtest waited. Break-even, the trailing stop, the exits by time, the daily loss limit and the trades-per-day cap all run. It holds **one position at a time**, because that is what the backtest measured, and if you restart it while a trade is open it picks that trade up instead of opening a second one on top.

Read this before BFX_MAINNET=1.

- **Check `PIP_SIZE` at the top of the file.** Every distance in the strategy is counted in pips. When the history you measured carried its instrument card, `PIP_SIZE` is that card's pip, and the bot keeps it even if the price has crossed a power of ten since, because a 30-pip stop must stay where it was measured. Only when there was no card, and the symbol is neither a currency pair nor a known contract, is it a guess; then the bot compares it against the live price on its first bar and **refuses to trade live** if they disagree, telling you the number to put there.
- **It sizes each trade as the backtest did** — the fixed lot, the risk percent or Kelly — with the units per lot of the measured instrument, capped at what the backtest's leverage could margin, or the venue's own if lower. Hyperliquid then takes the size in the coin's own step, and an order whose size that changes says so.

It charges you nothing for spread, commission or funding, because it cannot know them — so its running total is gross, and the daily-loss guard and Kelly read that gross number.

It wakes two seconds after each bar closes, reads it and acts, and never decides on the bar in progress — keep the computer's clock synchronised. If it was not looking when bars closed (asleep, offline, stopped), it goes through each of them in order when it is back, managing the trade as it would have; but it opens a trade only on the bar that has just closed, and only within two minutes of its close, never on a signal whose moment has passed. On a server, run it under `screen`, `tmux` or a systemd unit. An open trade survives the bot stopping: its stop and target are resting at the exchange, not in the process — with one exception the file names: an entry order Hyperliquid would not take with its stop and target attached rests alone, and a trade it opens while the bot is stopped has no stop until the bot runs again. It says so when you stop it.

The key never belongs in the file. It is read from the environment on purpose. Do not paste it into the script, and do not commit a file that has it — a strategy is meant to be shared, and a key inside one gets shared with it.

When the strategy reads other series

A condition can read a series other than the chart it trades — "only buy on H1 when the daily GBPUSD is above its average". The file asks the platform for that series itself, and reads it the way the backtest did: on each bar of the chart, the last bar of the other series that had *closed* by then, never the one still forming. When you download it, the Library warns that the file reads other series, and the top of the file lists what it reads and what to check.

- **Where they are picked.** In **Build** — in *Manual*, under the name and the symbol; in *Evolve*, in *Search*, under the history, and in the *Wheel* — and in the **Autopilot**: the same *Reference series* card → **Add reference series**, and all of them show the same series. The Reality Check judges with whichever are loaded. They come from a history in the library — the same SP500 in M1 gives its H4 and its M30 —; whatever is not there is downloaded or imported in **Data** first, a CSV too. The candles are grouped into the timeframe you pick, which can only be the history's own or a coarser one that contains it whole.
- **The series has to be loaded when you export.** Reference series live in memory: after a restart the strategy still names them, but nothing says which market they were. Export refuses until you add them again under *Reference series*, in **Build** or in the **Autopilot**, rather than guess.
- **The two series have to be on the same clock.** The backtest joins the other series to the chart's by the time of each bar. A MetaTrader file is in the broker's server time —two or three hours ahead of UTC— and a Dukascopy or Binance one in UTC: together, a condition would read the other series hours early, which is reading the future. When you load a reference series, BotFactoryFX measures the offset from how the two move and warns if there is one. Load them from the same source.

- **The symbol is a setting; the timeframe is not.** Brokers name markets differently — `EURUSD.a`, `GER40` against `DE40` — so each series has a symbol you can change: `Ref1Symbol` in MetaTrader, **Series ref1: instrument** in JForex, **Series ref1: symbol** in cTrader, the *Reference series* inputs in Pine, the `coin` in `REFERENCES` in Python. Its timeframe is part of the strategy and is fixed.
- **Until there is enough history of the other series, those conditions count as false**, exactly as they did in the backtest. It never decides on a value it could not read.
- **Daily and weekly bars depend on the clock.** The backtest used the bars in your file. MetaTrader cuts days at the broker's server time, TradingView at the feed's session, JForex and Hyperliquid in UTC, cTrader at the time its broker sets. If they differ from your file, the robot reads other bars.
- **Testers.** MetaTrader 4's tester does not move other symbols bar by bar; MetaTrader 5's needs real ticks or 1-minute OHLC; JForex's needs every instrument added to the test; cTrader's loads each symbol from your broker's history. Try it forward on a demo account.
- **MetaTrader 4 has no H2, H3, H6, H8 or H12 history.** A series on one of those waits until you build it on an offline chart. MetaTrader 5 has them.

What all six share

- **The indicators are the engine's.** Where a platform computes an indicator differently from the engine — its ADX, its RVI, its MACD signal line, and several more — the file computes that one itself, with the engine's formula, so the robot reads the line the backtest read. The cTrader cBot and the Python bot go further: they compute all of them themselves.
- **Nothing in a strategy reaches the file raw.** Free text is made inert, and every symbol, series name and number is checked, so a strategy somebody sends you cannot slip code into what you export.
- **Numbers the engine does not measure are refused** — a negative number of bars back, a period below 1 — by all six, in the same words, instead of being written out and traded.
- **When a destination cannot carry a strategy honestly, it offers no file.** The Library says why: a reference series that is not loaded, a timeframe the platform does not have, JForex's sizing and market refusals, a chart on a timeframe cTrader does not have, a Pine script past TradingView's limits, an indicator the Python bot has no port of.
- **They compile.** The MQL4 and MQL5 files are compiled with MetaEditor, the Java file with JForex's own compiler and the C# file with Spotware's cTrader package, as part of this program's tests. And the cBot is also run bar by bar against the engine: it decides and trades the same.

DATA, LIBRARY AND PORTFOLIO

Data and timeframes

Everything about the instrument you are testing lives on the **Data** screen: the price history, the timeframe, what your broker charges you to trade it, and what the instrument is. Asked once, used everywhere.

Where the history comes from

The sources sit in a row, in two groups. First the ones that hand you history: your own **CSV** — from your broker, or TradingView's chart export — then **Dukascopy** for currencies and **Binance** for crypto. Then, past a thin line, the platforms you connect to: **MetaTrader 5**, **MetaTrader 4** and **JForex**, which read a platform open on this machine through a connector, and **web3**, where you paste your wallet's public address and BotFactoryFX tells you which venue it has traded at. You do not have to know which one: the address says so. What is asked for there is public data — the kind anyone can read off the chain — not a private key, and nothing is signed.

Above the row, **What do you want to download?** looks for an asset in all of them at once: type “dow”, “gold”, “bitcoin” or a symbol, and it tells you which sources have it and how far back their history goes; one click opens its tab with the asset already chosen. JForex shows up according to what your account offers, which the program learns the first time you connect it: not every account offers the same. MetaTrader does not show up, because what it has depends on your broker: look in its tab.

The library

Imported files are kept on disk between sessions — up to **50 series**. Loading a second instrument adds it rather than replacing the first. Large exports are fine: they are read as a stream and folded into a workable timeframe as they load, so a half-gigabyte M1 file does not have to fit in memory.

Each history is kept in the program's own format, which is small and quick to read but which no other program opens. To take one elsewhere, the **Export to CSV** button on its row saves it as CSV — times in UTC and the same header that imports — and an exported CSV comes back in exactly as it was. And if a strategy has to read another market — a reference series — you download or import it here like any history and pick it in **Build** or in the **Autopilot**.

A history that carries the future

Some broker histories carry, in one hour of the day — nearly always the first — the high and the low of the whole day that starts with it; it is usually history rebuilt from daily bars. A rule that reads that bar's high or low — breakouts, channels, Ichimoku, stochastics, ATR — sees the day ahead, and the backtest makes what no one could make trading. The Reality Check cannot see it with its tests, which shuffle and cut whole bars, so it is looked for in the data: if the loaded history carries it, this screen says so at the top with the hour and the stretch, the check of what you import counts it as serious, and the Reality Check says it before anything else. Cut the history to start after that stretch, or get it from another source. A history without it shows nothing.

Stocks and their splits

When a company splits its stock — turns each share into several — the price drops by that same ratio from one day to the next and nobody loses anything. Dukascopy serves stocks without correcting for it: its Apple carries the 4-for-1 of August 2020 as a 74 % fall in one hour, and a backtest on that series wins or loses on a fall that never happened. So, for the **stocks and ETFs** you download from Dukascopy, and for a CSV that carries Dukascopy's name for the asset (AAPLUSUSD), the program finds the splits and corrects them on its own before saving: a gap between one close and the next open that is exactly 2, 3, 4, 5… up to 50 to 1 (or the reverse, when it merges them), after which the price stays at the new level. Everything before it is brought to the new scale, as exchange charts do, and the data-quality panel says so, with the date, the ratio and how many bars changed. A jump like

that in an index, a currency, a metal or a crypto is left alone — they have no shares to split, so it is something else — and so is one in a CSV whose name is in no catalogue, such as "AAPL": there the data review only warns, with the date and the ratio.

How much history to keep, and what it costs

The desktop app has no bar limit. Load what your idea needs. What it does have is a consequence, and the import preview puts it on screen before you commit: it names the resolution, the bar count and the date range together, so you can see what you are trading away.

That trade is the thing to understand. A file is stored at whatever resolution you import it at, and afterwards the timeframe selector can only offer that one and coarser — four M15 bars make an H1 bar, and nothing recovers the four from the one. So the choice is made once, at import. A 23-year M1 export is 8.7 million rows; the same file gives you every minute of the last few months, or every four hours of the whole two decades, and both are one dropdown apart.

Measured on this machine, one backtest of a simple crossover, so you know what you are buying:

- **50,000 bars** → 41 ms, 122 MB of memory
- **100,000 bars** → 78 ms, 156 MB
- **500,000 bars** → 345 ms, 318 MB
- **1,000,000 bars** → 783 ms, 586 MB

It is linear — about 0.8 microseconds a bar, no cliff. A single backtest stays fast at any of these. What grows is a *genetic search*, because it runs 840 of them: a million bars is a few minutes across your cores rather than seconds. And memory is real — several million bars will exhaust it. Above 100,000 the import preview says so.

Pick the window your strategy actually trades. Nobody backtests 23 years of one-minute bars; an intraday idea needs weeks of M1, and a swing idea needs years of H4. A large file takes about half a minute to read whichever resolution you choose — it is streamed, so the app stays responsive.

The instrument card

Every series carries an instrument card: the size of a pip, the units in one lot, and which side of the pair the account's dollar is on. The import works it out — from the name for a currency pair, a metal, oil or a known index; from the price for anything else, where one lot is one unit; and from the venue itself when the source states it, as Hyperliquid does. Every stop distance, every position size and every figure in a report is counted with it, and the exported robot is written with it. When the card has something worth knowing — a cross with no dollar leg, whose pip value is estimated; an index priced in euros, pounds or yen, which this engine does not convert — the library row says so.

The card follows the symbol. Each history's symbol is the one of the source it was downloaded from, and under *What you already have loaded* it is shown, not typed: it follows the highlighted history. If it is ever wrong — a CSV whose name does not say it — you correct it by clicking it in its row of the library and confirming with Enter, and only that history changes. Changing it replaces the card: for a currency pair, a metal or a known index the new name decides it at once; for anything else the old card is cleared and worked out from the prices again, right there if it is the open history and when it is opened otherwise. A card that arrives with the change — from a source that knows the real contract — wins over both.

What trading it costs, and where the figures come from

Every history is born with its costs, and the costs card says where they come from. From **MetaTrader**, your terminal's: the spread from the symbol's card, the commission you were actually charged, and the swap. From **Binance** and **Hyperliquid**, the venue's: the spread from the book and its published fee. From **Dukascopy** and

JForex —its platform—, the spread *measured on its own feed*: at the end of the download the bid and ask candles of the last two complete months are fetched too, and their difference, hour by hour of the week, is what each entry is charged; and the commission from its published table for the smallest accounts (35 $ per million traded per side on currency pairs, 52.5 on the rest; 0.10 % on stocks and ETFs, plus a minimum per trade the backtest does not charge). A larger account pays less: adjust it down if that is your case. If the measurement cannot be made —the feed loses a file—, the history is saved with the preset, and it says so.

A **CSV** does not say which broker it comes from, and gets a preset: the figure of a standard account on currency pairs, carried over as a relative cost. Checked against Dukascopy's raw spread, on silver, copper, crude, gas, crypto, stocks or bonds it fell short by five to thirty times, so for the measured instruments it no longer goes below what Dukascopy charges raw. It is still not your broker: your own figures are always better.

Changing the timeframe rebuilds the bars

The timeframe selector is not a label. Picking a coarser one **rebuilds every bar** — open from the first, close from the last, the true high and low of the whole period, volume summed — and the chart, the backtest, the genetic search and the exported code all move with it together.

Once stored, only the file's own resolution and coarser ones are offered. Four M15 bars make an H1 bar; nothing recovers the four from the one, so a finer timeframe is not on the list rather than being faked. Weekly bars start on Monday and monthly bars on the 1st, in UTC, so two overlapping files always produce identical bars.

Connecting MetaTrader 5

BotFactoryFX can pull price history straight out of a running MetaTrader 5 terminal — every instrument your broker offers, any timeframe, without exporting a single file by hand. It reads the terminal and **never places an order**.

MetaTrader 4 connects the same way, through a bridge of its own: the last section of this chapter, *Connecting MetaTrader 4*, tells how.

How it connects

Through a small add-in called the **connector**: an MQL5 *service* that runs inside MetaTrader with no chart. It talks to this app by writing files in MetaTrader's own common folder.

That is a deliberate choice. A network connection — MetaTrader's MCP server, a socket, or `WebRequest` — needs a setting inside the terminal that has to be switched on and remembered, and MetaTrader is known to forget it between restarts. Files need no port, no allow-list, no firewall rule and no dialog. And because a *service* is restarted by the terminal every time it opens, you install it once and never think about it again.

Installing it, once

1. Open MetaTrader 5 and log in to your account.
2. Here, in **Settings** → **Trading platforms**, press **Install the connector** in the MetaTrader 5 panel. It is also on the **Data** screen: pick **MetaTrader 5** among the sources, open **Import from MetaTrader 5** and press **Install the connector**.
3. If MetaTrader was open, close it and open it again; if it was closed, just open it. The connector starts with it.

The button does all of it: it finds MetaTrader's folder —inside the Wine prefix too—, puts the connector in its services folder and sets it to start with the terminal, the same entry MetaTrader writes when you add a service by hand. From then on the terminal starts it every time it opens, and the panel here turns green on its own. If you would rather not close MetaTrader right now, you can start it from the Navigator instead (`Ctrl+N`): **Services** →

BotFactoryFX → right-click → **Add service**, then **Start** — once: adding it twice makes two copies, and the panel says so.

The bridge folder

The connector is a small program of its own — **BotFactoryFX Bridge** — and it travels inside this app as a folder: the compiled service (`BotFactoryFX.ex5`), its source (`BotFactoryFX.mq5`, so anyone can read what it does), the icon MetaTrader shows in the Navigator, a `LEEME.txt` and the manual as a PDF. Under the install cards there are three buttons: **Save the bridge folder (connector, manual, icon)** copies the whole folder to `~/.botfactoryfx/conectores/metatrader5` and shows the path with a copy button; **Open the manual** saves it and opens the PDF; **Open the folder** saves it and opens that folder in your file manager. It is for installing by hand, reading the source, or carrying the connector to another computer — the install button above does the everyday job on its own.

On Linux: which Wine

MetaTrader 5 runs on Linux through Wine, and **MetaQuotes supports Wine 10.0 and above**. On anything older the terminal prints *"unstable and unsupported"* in its own log at every launch. This app reads that line and tells you which Wine your terminal is on, because a problem in the host underneath looks exactly like a problem in the app on top.

On Debian, Ubuntu and Mint the distribution's own Wine is usually too old. WineHQ's is not:

```
sudo apt install --install-recommends winehq-stable
```

Two traps when upgrading Wine. Both are silent — neither gives an error you could search for.

- **Close everything Wine is running first**, including processes you cannot see: `wineserver -k`. The old and new versions keep their sockets in different folders, so the new Wine cannot tell that the old one still holds your MetaTrader folder. It simply will not open, and says nothing.
- **If you remove the old Wine packages**, the menu shortcut for MetaTrader may stop working: Debian and Ubuntu provide a `wine-stable` command that WineHQ does not, and the shortcut calls it by name. Clicking the icon then does nothing at all. Edit the `.desktop` file and change `wine-stable` to `wine`.

Installing MetaTrader 5 on Linux, or handing it to an AI

A smooth connection depends mostly on MetaTrader being set up right on Wine: Wine from WineHQ, a prefix of its own (`~/.wine-mt5`, which this app finds by itself) and that prefix in **Windows 10** mode — on Windows 7 it fails in ways that do not look like Wine. The verified procedure, step by step, with each step's check and its four traps, travels in the bridge folder as `INSTALL-MT5-ON-LINUX.txt`.

If you would rather hand it to an AI assistant with access to your terminal, give it `PROMPT-FOR-YOUR-AI.txt` from the same folder, or press **Copy the prompt for your AI** in the MetaTrader 5 panel when it finds no terminal. It carries the same steps and the limits it must keep: it asks before every `sudo`, removes nothing without asking, does not download the installer — you get it from your broker —, never touches your broker login or password, stops and asks at any unexpected output, and shows you the five checks at the end.

If an instrument comes back empty

MetaTrader only keeps bars for charts it has been asked to draw. The first request for an instrument it has never opened makes it download the history first — for several years that can take a minute or two, and it is instant after that. If it still comes back empty, open that chart in MetaTrader once and scroll back.

The instrument's trading hours

Since connector 1.5.0, every MetaTrader history arrives with the hours your broker publishes for that symbol: when it takes orders and when it only quotes, day by day and in its server time, which is the same clock as the bars. You see them in the costs card. And the audit of what arrives uses them: a session day with no bars at all is a holiday, not a gap —up to three in a row; a whole week is a hole in the data—, and what is missing is counted against the hours when the broker quotes, not against the ones the file usually has. What it does not know: these are today's hours. If the broker changed them —an index that used to quote only in exchange hours—, the old part can show gaps that are not; and a holiday eve that closes early counts as missing bars, because MetaTrader publishes the standard week, not the holiday calendar. With an older connector they do not arrive, and everything stays as before: the hours are worked out from the bars themselves.

The backtest does not trade when your broker would not let it. What is decided at a bar's close —entering, placing a pending order, exiting by a rule or by the clock— a robot sends with the first price of the next bar, and it only goes through if trading is open then; what rests on the server —the pending order, the stop, the target, the trailing— only fills in bars with trading hours. An index that quotes from one in the morning and opens at eight no longer fills orders overnight; and if at eight the price opens beyond the stop, the stop fills at the open, as after a weekend.

After an update

The program keeps the connector up to date by itself: when a new version brings a new connector, it replaces the old file in every MetaTrader where it finds one, as soon as it starts, and tells you. A running service keeps the code it started with, so MetaTrader goes on with the old one until it is closed and opened again; the panel says so, and how to switch without closing it: in MetaTrader, Navigator → Services → BotFactoryFX → Stop, then Start.

And when the connector does not answer, the panel reads MetaTrader's own files to tell you why: whether MetaTrader is closed or open, and whether the connector is missing, old, not set to start, or there twice. Whatever it is, the button next to it fixes it.

Connecting MetaTrader 4

MetaTrader 4 has its own bridge, this one's twin: it asks the same questions and answers the same things — candles, the contract specification, the account, what is open and what was closed—, and it speaks the same protocol, which is why it carries the same version number. What changes is what it is. MQL4 has no services, and the bridge is an **indicator** that sits on a chart, any chart: it draws nothing, and the symbol and the timeframe do not matter to it. An indicator and not an expert because, measured in a real MetaTrader 4, with the **AutoTrading** button off an expert never receives a single timer event while an indicator keeps running; and because MetaTrader 4 lets only experts and scripts trade: as an indicator, the bridge could not place an order even if someone wrote one into it. Nor does it take the chart's expert slot, so it can sit on the same chart as a robot.

The same button installs it, **Install the connector**, in the MetaTrader 4 panel of **Settings → Trading platforms**; it is also on the **Data** screen: pick **MetaTrader 4** among the sources and open **Import from MetaTrader 4**. It copies the indicator into the terminal's `MQL4/Indicators` folder and, **with MetaTrader 4 closed**, adds to its profile an H1 chart that carries it, which MetaTrader 4 opens every time it starts. So close MetaTrader 4, press the button and open it again: on that chart, bottom left, a label starting with `BotFactoryFX Bridge` says the bridge is running and read-only, and the panel here turns green on its own.

With MetaTrader 4 open, the button only copies the file: when it closes, MetaTrader 4 rewrites its profile from what it has open, and a chart added from outside would be lost. The panel says so, and offers the way that does not wait: in MetaTrader 4's **Navigator**, drag **Indicators → BotFactoryFX** onto any chart. It starts at once, and MetaTrader 4 keeps it in the profile.

It talks in a folder of its own inside the common folder, not in the MetaTrader 5 bridge's: both MetaTraders share that folder on Windows, and also in a Wine prefix that has both, and each bridge would pick up the other one's

questions. MetaTrader 4 keeps *orders*, not *deals*: each closed order arrives as its entry and its exit, and only closed ones. And it only hands over what its **Account History** tab shows: if only the last few months come through, right-click there and choose **All History**.

If a history is not in MetaTrader 4 yet, the bridge does not sit waiting: it goes on answering everything else, tries again for 20 seconds and, if it still has not arrived, tells you to open that chart in MetaTrader and scroll back so it downloads.

On an update, the program replaces the old indicator with the new one in every MetaTrader 4 where it finds one, without touching its profile; the one already running keeps its code until MetaTrader 4 is closed and opened again. To remove it, close the bridge's chart and delete the indicator from `MQL4/Indicators`. Its panel has the same three bridge-folder buttons, and the folder is saved in `~/.botfactoryfx/conectores/metatrader4` with its manual.

And your robots: in the **Library**, a strategy's **Download** menu has, under **Send it to the platform's folder**, **MetaTrader 4**, which leaves the `.mq4` in `MQL4/Experts/BotFactoryFX`; then open it in MetaEditor, compile it with `F7` and drag it onto a chart. The MQL4 robots this version exports write their heartbeat and their journal next to the MetaTrader 4 bridge, and the earlier ones next to the MetaTrader 5 one: export them again to see them with their terminal.

Connecting JForex

BotFactoryFX can pull price history straight out of a JForex platform running on this machine — everything your account offers, subscribed or not, any timeframe, without exporting a file by hand. It reads the platform and **never places an order**.

How it connects

Through the same idea as MetaTrader 5: a small add-in called the **bridge** — *BotFactoryFX Bridge* — that runs inside the platform and talks to this app by writing files, here in `~/.botfactoryfx/jforex`. No port, no allow-list, no firewall rule. The difference is what the add-in is: JForex has no services, it runs *strategies* written in Java. The bridge travels in two forms: compiled, in a `.jfx`, which is what JForex starts, and as its source, `BotFactoryFX.java`, so you can read every line of it before you run it. The compiled one is built from that source with JForex's own compiler and its very options, and it is, byte for byte, the one JForex writes when it compiles it; it carries nothing of your account or your machine. It answers one request at a time, which is why the panel queues a second instrument instead of asking for both at once. JForex only serves candles for what you are subscribed to, so for a download the bridge itself subscribes the instrument and removes it once a minute goes by without a request: your JForex list stays as you left it.

Installing it, once

On Linux and Windows this app installs it, whole and in one go:

1. Here, in **Settings** → **Trading platforms**, press **Install the bridge in JForex** in the JForex panel (the same one the **Data** screen shows when you pick **JForex** among the sources). The app finds your JForex — on Linux, through the shortcut its installer leaves or in `~/JForex4`; on Windows, through what its installer notes in the registry or in `C:\Program Files\JForex4` — puts the bridge in its strategies folder, compiled and with its source, and adds two lines to JForex's options (`JForex4.vmoptions`, with a copy of the previous ones beside it) so that it starts the bridge when it opens.
2. If JForex was open, close it. Open it and log in to your account: the bridge starts by itself, on your computer, a few seconds later (measured: 13 on Linux, about 20 on Windows), and the panel here turns green on its own.

From then on it starts every time you open JForex, without you opening or starting anything inside it.

If your JForex already starts other strategies when it opens, that setting is left alone — yours would start too, and those may trade — and the panel tells you to start the bridge by hand.

On Windows, JForex installs itself in `C:\Program Files\JForex4`, which is also its data folder: it keeps its strategies there, and that is where this app writes. This was measured with JForex for Windows on Wine; whether its installer lets a user who is not an administrator write in that folder is still to be measured on a real Windows. If it does not, the panel says so, with the path: give your user permission to write in that folder and press the button again.

On a Mac (where the app does not look for JForex yet, because it has not been measured there), or if it does not find yours, it is done by hand:

1. Open JForex and log in to your account.
2. Here, press **Copy the connector**: it leaves the bridge, compiled and with its source, in `~/.botfactoryfx/jforex`, and shows the path of the compiled one, with a button to copy it.
3. In JForex, on the **STRATEGIES** tab, press **Open** (the folder icon) and paste that path into the file name box: the folder's name starts with a dot, and on Linux and on a Mac the dialog does not show it.
4. Start it on your computer: **Start** → **Local Run** (JForex refuses a **Remote Run**, because the strategy asks for full access). It comes compiled, so it simply starts. Leave it running: the panel here turns green on its own.

The permission JForex asks for

The first time the bridge starts on an account, JForex shows a dialog — the **Full Access Disclaimer** — asking whether the strategy may have **full access**. Accept it with **I Agree**. Dukascopy keeps the answer on its server, with the account: on an account that has already accepted it, JForex asks nothing, not even on another computer. The strategy carries the annotation `@RequiresFullAccess` because JForex's policy only lets a strategy write inside its own files folder, and the bridge has to write its answers in `~/.botfactoryfx/jforex`, where this app reads them. The permission is about *where it may write*, not about trading: the bridge has no code that places, changes or cancels an order, and you can check that in its source. The manual shows the dialog.

The bridge folder

The panel also has three buttons: **Save the bridge folder (connector, manual, icon)** copies the whole folder — the compiled bridge and its source, the icon, a `LEEME.txt` and the manual as a PDF — to `~/.botfactoryfx/conectores/jforex` and shows the path; **Open the manual** saves it and opens the PDF; **Open the folder** saves it and opens that folder in your file manager.

After an update

A running strategy keeps the code it started with. So a new version of BotFactoryFX brings a new bridge, but the one JForex is running is still the old one. If this app installed it, the panel, as soon as you open it, puts the new one in JForex's strategies folder by itself and tells you so: it takes over the next time you open JForex. By hand: copy it again, stop the strategy and start it again. The panel compares the version that is running with the one it ships, and tells you.

If something does not answer

- **No JForex platform is answering**: is the strategy running? Installed by this app, it starts when you open JForex and log in, not before: if JForex was already open when you installed it, close it and open it again. Its console line reads `BotFactoryFX Bridge … escribiendo en …`, and it writes a status file every few seconds; if the platform has been closed, the panel notices on its own.

- **Only your subscribed ones show up**: a bridge from an earlier version is running, which offered only those. Installed by this app, close JForex and open it again; by hand, copy it again and restart the strategy.
- **Entries in STRATEGIES you did not put there**, with paths from another computer: that list lives in your account's cloud, and anyone who uses the same account on two computers sees the other one's entries there. To load the bridge by hand, choose the file with **Open**, not an entry in the list: opened from the list, JForex compiles it in its cloud and leaves nothing on the disk. The automatic start does not use the list.
- **A request seems stuck**: JForex answers one request at a time, and fetching years of history for an instrument it has never drawn takes a while. The download strip at the top of the Data screen shows what is on its way, and a big download moves forward in chunks of 20,000 candles.

And your robots: in the **Library**, a strategy's **Download** menu has, under **Send it to the platform's folder**, **JForex**, which leaves the `.java` in your JForex's strategies folder, the same one as the bridge. It works on Linux and Windows; not on a Mac yet: download it and open it in JForex. Then, in JForex, open it with **Open** on the **STRATEGIES** tab and start it when you want it to trade: **Start** → **Local Run**.

Working with TradingView

TradingView and MetaTrader are connected in opposite directions, and it is worth knowing why before looking for a button that does not exist.

> **There is no automatic link, and there cannot be one.** MetaTrader is a program on your disk with a plugin folder, which is what the BotFactoryFX connector uses. TradingView is a website: Pine Script runs sandboxed on their servers and cannot read or write anything on your machine. TradingView also publishes *no market-data API* for individual users — their prices are licensed from exchanges under terms that do not allow redistribution. Tools that claim otherwise are reverse-engineering an undocumented private channel, and they break without warning.

Bringing price history in

Use TradingView's own export: on the chart's top toolbar, **Download chart data...** (PRO+ and Premium). Then drop the CSV on the Data screen here. It is read as it comes — both the ISO timestamp variant and the Unix-seconds one — and the indicator columns TradingView appends are ignored rather than mistaken for price.

Taking a strategy out

Build the strategy here, then **Export...** and choose **Pine Script** in the Library's menu. Open the Pine Editor on a TradingView chart, paste, save, and add it to the chart. The Strategy Tester tab then runs it on their data.

Checking that the two agree — the part that matters

Two backtests of the same rules on the same instrument should produce the same equity curve. When they do not, one of them is wrong, and knowing *which* is the entire point of a Reality Check.

So bring TradingView's answer back: in the Strategy Tester, open the **List of Trades** tab and export it. Then go to **Reality Check** → **One from outside**, pick TradingView as the platform, attach that file, and press **Run mine on this history**. The two land side by side in one table.

> **Export the List of Trades, not the Performance Summary.** The summary is a page of totals, and a total nobody can recompute is a total nobody can check. Every figure shown here is recalculated from the individual trades — including the starting capital, which TradingView does not state in that export but which its profit and profit-per-cent columns give away between them.

What a difference usually means

- **Costs.** The exported script sets none — no spread, commission, slippage or funding — because the export does not carry the ones the backtest charged, and says so at the top. TradingView's totals are therefore better than the backtest's by what the backtest paid, or worse by any funding it collected. Setting *Commission* and *Slippage* in the script's *Properties* brings the totals closer, not the individual trades.
- **Data.** TradingView's feed for a symbol is not your broker's. Different sessions, different closes, different gaps.
- **The bar the order fills on.** The backtest enters at the market at the close of the signal bar, and the exported script does the same — it runs with `process_orders_on_close` on. A script of your own left on TradingView's default fills at the open of the next bar instead, which after a gap is another price.

Library

The strategies you kept, each with an **exhaustive report**. Master-detail layout: list on the left — search it, sort it by date, name or symbol — detail on the right.

- KPIs from a backtest over the loaded history (P&L, WR, PF, Max DD, Sharpe).
- Interactive equity curve.
- Rules in plain language.
- Risk management and full trades table.
- Expandable JSON for inspection.
- **Download** — the strategy as a file for any of the six platforms, or as JSON. It carries the instrument card of the series open at the time. **Import** reads such a JSON back.
- **Edit** opens it in Build › Manual; **Full report** opens the whole report.
- **Duplicate** — a copy under another name to change without touching the original: it remembers which one it came from and takes its verdict and its count of variants along.
- **Notes and tags** — each strategy's own; the list can be filtered by tag, and the search looks at them too.
- **Compare with…** — two strategies side by side: how their rules differ, their figures on the loaded history and both curves.
- **Explain it to me** — your AI provider (the one under Settings) tells in plain language what the strategy does and what its verdict means. Only when you press the button, and what travels are its rules in words and its verdict, nothing else. It is kept; if you later change the rules, the card says it went stale.

In the **full report**, the profit factor and the average result come with their 90 % margin, from drawing the trades again — with few of them it comes out wide, and that is what has to show; and among what it says under the charts is the stop that would have kept nine winners in ten, next to the one the strategy has.

> **Judged from here, a strategy carries a count only if it matches.** The Reality Check is told how many variants were tried to find a strategy picked from the Library only when its rules, risk and management are the same as those of the strategy open in Build that carries a count. See *The search that found it was large*, under Reality Check.

Portfolio

Combines multiple strategies with an **allocation** (weight) each. Balance and lot size scale proportionally.

Each strategy is simulated on its own history — its symbol, its timeframe, its spread and its commission — and the curves are combined by date. A portfolio of several instruments is the normal case, and this is what makes it measurable: a leg is worth what it was worth where it was judged. One whose history is not loaded is left out and named; it is never run on another instrument's candles to make a number appear.

- **See their curves before combining** — each candidate alone, on its own history, with the whole starting balance and no allocation, on one chart: to see whether they fall at the same time before putting them together.
- Aggregated KPIs and equity curve with per-strategy breakdown, each row saying which history it ran on.
- Correlation matrix between pairs (on equity-curve returns).

> **Correlation.** +0.7 to +1: parallel (little diversification). -0.3 to +0.3: independent. -1 to -0.7: opposite (hedges). It is measured only where both legs have history, and if they share too little the screen says so instead of giving a number.

> **Legs that do not cover the same dates.** Outside its own history a strategy counts as idle capital: its share of the balance, not trading. Nothing is invented and no history is thrown away, but the combined drawdown covers stretches where part of the portfolio could not trade. The screen lists the legs whose history starts late or ends early, and over what dates; a gap inside a history counts the same way and is not listed, because that is a fault of the file and shows up under Data.

Portfolios can be saved under a name, loaded again and updated. A portfolio saved before 23 September 2026 carries figures measured the old way — every leg on whichever history was open that day — and says so when reopened, with a button to simulate it again. Arriving from an Autopilot run with nothing selected, the strategies its own portfolio combined are already selected — the ones that came out solid, which are the only ones it combines. Others you chose to save at the end of the run are in the Library, but they are not selected here. If it combined nothing, nothing is selected, and the screen says so instead of leaving you an empty page — what the run found is in the Library, with its verdict. Quarantine and Live used to be tabs of this screen and are sections of their own now, in the order of the work: Quarantine before Portfolio, Live after it. This screen does one thing — combine strategies and simulate them together.

AFTER THE BACKTEST

Quarantine

Its own section, between Library and Portfolio, because that is where it falls in the work: the Reality Check decides what deserves keeping, Quarantine decides what deserves money, and only what passed both goes into a portfolio. A backtest measures the past; trading measures the present with money on it. Quarantine is what goes in between: the strategy runs on the candles your broker is publishing, without a single order being sent, and a record builds up that you can hold against the backtest that approved it.

It does not expire, a strategy of yours needs no account anywhere, and there is no limit on how many you keep.

Putting one in

- **Strategy** — one from the library. Its name and its last backtest's figures are copied, so the record still reads if you later delete the strategy. If its verdict is current, the record also notes its score and how it came in — whether it passed the bar or came in *close to the bar*: 45 or more, or solid without its sibling — so you can compare later how each group does live. Its card shows the score: green if it passed the bar, amber if not. The Autopilot's *Send to quarantine* button brings it here already picked. If its stop and target are set in pips, the record also warns when the market now moves 1.5 times as much per candle as on the stretch the Reality Check judged, or 1.5 times less: in proportion its stops are no longer the ones that were judged, and in a market that has doubled, stops in pips are too tight and the usual noise reaches them. If it came out of a search with stops in pips, a box under the form asks what to do with them: **adapt them to the ATR**, recommended and picked to begin with, or **keep them fixed**. Adapted, they measure exactly the same on the last candle the Reality Check judged, and from there they grow and shrink with volatility; measured on eight markets, the strategies the Reality Check did not reject beat chance afterwards 75 % of the time instead of 63 %. The record then says *stops in ATR*. A strategy you set up by hand is not asked: its pips are yours.
- **Broker or venue** — Hyperliquid, any MetaTrader terminal you have open, or JForex when its connector is running. This is half the question, not a detail: the same strategy on the same instrument behaves differently in each venue, because the spread, the funding and the liquid hours all change.
- **Without an account anywhere** — Binance (spot crypto) and Dukascopy (currencies, indices, metals and shares) say *no account needed*: their candles are public, so you can put a strategy in quarantine before opening an account with any broker. They are that venue's prices: if you will trade with another broker, its spread and its hours will differ a little. Dukascopy publishes the recent days one file at a time, so the first update takes a few minutes; after that it asks only for the new days. Each record says which costs it charges — spread, slippage and commission — and where they come from: those of the strategy's history, or default ones when that history is not loaded. If your broker charges more, your result will be worse than the record's.
- **Instrument** and **candle** — whatever that venue offers, M1 to D1.

Each time you open the section, the new candles are fetched and the strategy is re-run over everything accumulated: each quarantine's candles are kept on this computer and the new ones are added to them. Nothing runs in the background: the app does not need to be open for the market to carry on, only to pick up what happened. The one thing it cannot pick up is what the broker no longer keeps: each visit reaches back a few thousand candles — a few days of one-minute candles, months of one-hour ones — and if it stays closed longer than that, that stretch is missing and the record says so.

A robot that is already trading

At the top of the card you choose what goes in: **One of your strategies** or **A robot that is already trading**. The second is a robot you were sold, held against the backtest it was sold with. Its code is never run — it is read for the numbers it declares — and what gets compared is its promise against the trades the account where it runs really

made, so here the sources are accounts: MetaTrader 5, MetaTrader 4 or JForex through their connector, or a wallet from Settings. It is what BFQuarantine does. In a MetaTrader account you also choose which robot, by its magic number, and whether to count from its first trade there or from today.

The verdict looks at the money first

What matters is whether it makes the money its backtest promised, so the verdict compares the **profit factor** — what the winners made divided by what the losers lost— live against the one promised. With few trades a profit factor says little, so a band is drawn around it by resampling its own trades two thousand times: it only says *doing worse than its backtest promised* when the promise sits above that band with 90% confidence, and never before 20 trades. The hit rate is shown too, as information: a robot can win less often than promised and still make its money, or win as often and lose it. Five answers: no trades yet, too early to tell, losing money but not yet clearly worse than promised, making money as promised (or better), and doing worse than promised. Measured on 32 real robots built with StrategyQuant X against what they did in months they had never seen, this caught 12 of the 21 that lost money without one false alarm among the 11 that did not; the hit rate alone caught 10 and raised 4 false alarms. If the backtest gives no profit factor, the hit rate is compared as before, with a 95% Wilson interval.

A second chance for what came close

A strategy that came in *close to the bar* did not pass the Reality Check's bar, and quarantine is its second chance. When its verdict says it is *making money as its backtest promised* —which takes at least 20 live trades—, the record marks it **Approved in quarantine**, and the Library shows it next to its verdict. Until then the card says how many trades it has and roughly how many it needs. The bar has not been lowered: what approves it is a market it had never seen, not a score. And it is not for life: if it drifts, the mark goes away.

> **The chart draws what the strategy would have done** — every entry and every exit on the candles that have arrived — not orders left waiting.

> **This is not a demo account.** It places no orders anywhere, so it measures no slippage, no partial fills, no rejections and no latency — things a broker demo does touch to some degree. What it measures is whether the strategy still resembles its own backtest on a market that is the real one. For the other thing there is **Live**. And it does not depend on one either: it does not expire after a few weeks the way a broker's demo does, and it needs no server running day and night.

Live accounts and wallets

The Live section, the last in the menu, reads real accounts and draws their trades on the chart. A bot running loose against an exchange trades blind: the fills come out on a console and the chart is in another window, of another program, at another scale.

- **Hyperliquid** — from the wallet address. Equity, open positions, resting orders, closed trades and the account's curve.
- **MetaTrader 5** — through the same connector that brings the history, with no key and no open port. At the top, the account: balance, equity, floating, margin level and what is open, with the robot behind each position (this needs connector 1.4.0). It reads *trades* rather than ledger entries: it pairs each entry with its exit, nets out commission and swap, and separates robots by magic number. It asks only for new trades, not the whole history every time.

- **JForex** — through its connector, like MetaTrader: a strategy inside the platform that only reads. Under MetaTrader's account, its account: balance at yesterday's close, equity, floating —equity minus base equity, because JForex's balance is the previous day's—, margin level and what is open, with the robot behind each position (this needs connector 1.3.0).
- While MetaTrader 5, MetaTrader 4 or JForex is not connected, this section offers to install its connector, as Data does.

> **Reading a Hyperliquid account needs no key.** Its API answers to the address, which is public and on chain; the "API wallet" the exchange offers is for *signing orders*, and nothing is signed here. Storing a secret that is never used is risk without benefit, so it is not asked for.

Each robot, with its strategy and its drift

Every robot exported to MetaTrader or JForex carries a magic number taken from its strategy, which does not change when you rename it —in JForex it travels inside each order's label—. In *Your robots* that is enough to put its Library strategy next to it and measure whether it still does what it promised: the strategy is run on the same platform's bars, the trades it closed *before* going live are taken, and the robot's closed trades since then are placed among them. Below 20 closed trades it says it is too early, and how many are missing. JForex appears in the drop-down next to the terminals. It counts sizes in millions, so outside currency pairs a trade's lots are not said, and the drift compares money without normalising per lot, as it warns. Robots exported with earlier versions all carry the same number, 20250810, and cannot be attributed on their own: pick each one's strategy in the drop-down.

If you chose to **adapt its stops to the ATR** when you exported it from the Library, the robot does not carry the Library's pips, and neither does its backtest: the chart, the drift and the decision-by-decision journal run the strategy with the stops in ATR it went out to that platform with —the last time it was exported to it— and say so underneath. If they were kept fixed, or it was exported before the choice existed, with the usual pips.

What entering cost

Under the chart, next to what the backtest traded, is what the market entries really cost: the price they filled at against the close of the **signal bar**, the last one closed when they filled, which is the one a robot deciding at the close has just seen. Only what filled in the first seconds of a bar counts —10 % of the bar, between 5 seconds and 2 minutes—: a pending order or an entry by hand moves away from that close on purpose, and how many are left out is said. With each trade's own pip —the broker's digits on MetaTrader, what JForex says, the perpetuals rule on Hyperliquid— and buys apart from sells: MetaTrader's and JForex's bars are bid prices, so a buy fills at the ask and its figure carries the spread; a sell pays it on the way out. Compare them with what the backtest charges on entry: the spread, and slippage on stop orders. And the backtest beside it runs with that instrument's contract and, unless it is given costs, with its family's: never at zero cost.

Decision by decision

MetaTrader robots (MQL5 and MQL4) exported from this version keep a journal: in the terminals' common folder, next to the connector, they write down each bar they decide —its close as they saw it, what their three rules said (exit, long and short) and, if they did not enter, why: a trade open, an order waiting or the day's limits— and each order they send, with the price asked, what the broker answered and the price it filled at. A file a month; nothing is sent anywhere, and nothing is written in the Strategy Tester.

JForex robots exported from this version keep the same journal, in JForex's own files folder —the one a strategy writes in without asking for full access—, under `BotFactoryFX`. Each order is written down when JForex answers it, with what it answered; nothing in the robot waits for that answer, so it trades exactly as it would without a journal. Next to it they keep a heartbeat, and BotFactoryFX warns of a JForex robot that stops beating while prices

are coming, stops deciding, or trips its cut-off. Only JForex knows where that folder is: its connector (1.3.0) tells BotFactoryFX. Nothing is written in JForex's Historical Tester, and a strategy run on Dukascopy's remote servers writes it there, where BotFactoryFX cannot read it.

In *Your robots*, under the drift, each robot with a journal is set beside its backtest on the same bars of the same terminal, or of JForex, on the clock the robot counts in and starting flat where it started: signal by signal and decision by decision —entries, entry orders, their fills and expiries, and exits by the rule or on time; the stop and the target are executed by the broker, not the robot, and are compared in the drift. Each difference comes with its cause: the robot was off, the broker refused it (and with which code), the rules read differently, or one of the two had its door shut. What follows the first one is marked as a consequence, because twenty differences that come from a single one are not twenty problems. A different signal on the **same** bar is a fault of the export, and worth telling us about; on **another** bar, the history is not what the robot saw. And for what filled at the market, what was got against what was asked: the real slippage, order by order.

> **Robots exported before this do not write a journal.** Export it again from the Library and put it back on its chart. Python and Pine robots do not keep one yet.

Wallets are added in Settings

An address belongs to whoever holds the key, not to a venue, so the list is not "the Hyperliquid account" but whatever wallets you like, named by you. Pasting an address tests it against the venues that can be read and fills the rest in. Hyperliquid shows two different addresses: your account's, in your account menu at the top right, and the one in its “Deposit” window, which is a mailbox that forwards the money to the account. It is the first one that goes here; with the second, the program warns that there is no account at it. The field also strips the invisible characters a copy-and-paste from a web page drags along, which used to make a good address bounce with nothing visibly wrong.

Home lets you pick which curve you are looking at

Simulated, quarantine or live — and under the title it says which of the three: what *would* have happened over imported history, what *would* have happened over candles that have since arrived, or what *did* happen with money. They are never added together; each starts from its own line.

Dr. Quant

Dr. Quant lives in two places. In the **Autopilot** conversation: whatever you write there that is a request to build is read in code and built. Since 08-10-2026 two more orders are read by the program too, without spending your key: **judging a saved strategy** — "judge Breakout 7", or "judge it" with one open — and **marking it in the Portfolio selection or taking it out** — "take Breakout 7 to the portfolio". Both show a card first with exactly what is about to happen — which history the judgement runs on, with which attempt count — and nothing runs without your click on that card; the judgement is the same Reality Check as ever, on the strategy's own history, and its verdict is saved with it. A question — a doubt about a metric, what to do next — goes to your AI provider as conversation; and an order that is none of those — "optimize the RSI period" — is not run from there: Dr. Quant tells you on which screen it is done. **What travels with a question**: the conversation, the verdict journal summarised — what you already tried, so it is not proposed back as a new idea — and only what the question names of your own work, summarised by the program: ask about your Library and the names travel with their verdict; name a strategy and its judge's card travels; ask about your data or your saved portfolios and theirs does. Your rules, your candles and your trades never travel from here. And on the **Data** screen, where **Ask Dr. Quant** looks up your broker's terms to fill in what it costs you to trade (see *Dr. Quant on the web*, below).

It answers as it would to someone starting out: it says what each term means the first time it uses it, and ends with what to look at next. When the answer is not in what it knows of this program, it says so and sends you to the chapter of this Help that covers it, rather than filling the gap with how other programs work. When you ask about the Reality Check —the verdict, a check, what caps the score, overfitting— it is also told how this program's judge works, from the program itself: the checks with their weights, where each word of the verdict starts and what caps the score. Nothing of your strategy goes with it. And when the question is about the program, it gets the passages of this Help closest to it, copied as they are and with the chapter they come from, so it can send you there. All of that has a ceiling —what it is told before each question stays under about 2,500 tokens, and your key pays for it in every message— and the part that never changes goes first, so that the providers that charge less for text repeated from one message to the next can charge it to you for less.

Examples: "Is a Sharpe of 1.8 with 12 trades reasonable?", "How do I validate this strategy out-of-sample?", "Suggest 3 exit conditions for a moving-average crossover."

> **There is no Dr. Quant button in the menu any more, and no agent mode.** If you come from an earlier version: the side panel, with its two modes, chat and agent, was retired. Two doors with the same name and different powers were worse than one, and agent mode was a model improvising on your strategy with tools and no way to reproduce what it did — exactly what this program teaches you not to believe. What it was asked for — "find me a strategy for EURUSD on H1" — the Autopilot does now: it reads the request in code and shows what it assumed before it starts.

Dr. Quant runs on your own AI account. No model is bundled here and nothing is relayed through anybody else's servers. Add a provider in **Settings** with your own API key. The list starts **empty**, and the key tells the app whose it is: `sk-ant-…` is Anthropic, `gsk_…` is Groq. There are five kinds: **OpenAI or compatible**, **Anthropic**, **Google Gemini**, **OpenRouter** — the one that can search the web with any model — and **On your machine**, which needs no key and whose model button lists whatever you have downloaded. The first and the last take an **Address**: empty is OpenAI or Ollama, and any service that speaks OpenAI's dialect — Groq, DeepSeek, Mistral, LM Studio, your own server — is its address pasted there. A key whose signature belongs to one of those, `gsk_…` for Groq, comes in as *OpenAI or compatible* with the address already filled in. The questions are billed to you, by them, at their prices, and the key stays on this machine.

Under each answer, in small print, is what your provider says it cost: the tokens it read —and how many of them came from its cache, which it charges for less— and the ones it wrote. Each question also carries the conversation before it, and that is paid for too: **New conversation**, at the top of the conversation, starts again from nothing, and **Copy the conversation** takes it all out as text.

With more than one active provider, the first by priority answers, and if it is saturated with a rate-limit error (429) the question passes on its own to the next active one.

The Autopilot needs none of this to build: it reads what you ask in code, not with a model. Only its questions go to one.

Dr. Quant on the web — and what it costs

Dr. Quant can look things up on the internet: broker contract specifications, typical spreads, commission schedules. Two different things happen under that one word, and they are billed differently. It matters enough to say plainly.

Reading a page is free

When you give Dr. Quant a **URL**, BotFactoryFX's own server downloads that page and hands the text to the model. Nobody bills you for it — not your AI provider, not us. If you paste the link to your broker's contract-specifications page into **Data → What this costs you to trade**, the whole lookup costs **nothing**, and it reads the official page rather than whatever a search engine thought was relevant.

Searching the web is not free — not even on a free model

A *search* is a separate service your provider buys on your behalf, and it is charged separately from the model. **A free model does not make the search free.** OpenRouter says so in as many words: using web search incurs extra costs, even with free models.

At OpenRouter's current rates, one search costs roughly **$0.007** with the default engine (Exa, auto mode) or **$0.005** with Parallel's basic mode. Ten results are included in that price; BotFactoryFX asks for five, so a lookup never quietly runs over into the per-extra-result charge.

In practice: looking up your broker's terms a hundred times costs about **70 cents**. Negligible for you — but it is your money on your own API key, so you are told rather than surprised.

> **The cheap habit.** Paste the URL when you have it, type the broker's name when you don't. The first is free and better evidence; the second costs less than a cent.

> **Anything fetched from the web is treated as information, never as instructions.** Downloaded text is fenced before the model sees it, private and local addresses are refused, and every redirect is re-checked — so a page cannot talk Dr. Quant into acting on your library.

Frequently asked questions

Do I need to install anything?

No. The desktop app is the whole program: the genetic search, the Reality Check and the export all run inside it, and your work is kept on this machine. Connecting a trading platform — MetaTrader 5, MetaTrader 4 or JForex — needs its connector installed once, from the Data screen or from Live.

What CSV format is accepted?

Any CSV with columns for time, OHLC and optionally volume. The importer auto-detects the most common headers.

How is P&L calculated?

In pips, times lots, times what a pip of one lot is worth — and all three come from the instrument card of the series (see *Data and timeframes*): the size of a pip, the units in a lot, and which side of the pair the dollar is on. On EURUSD a pip is 0.0001 and a lot 100,000 units, so a pip of one lot is $10. On USDJPY a pip is 0.01 and its value in dollars moves with the price. A lot of gold is 100 ounces; a lot of a crypto perpetual is one coin. On a cross with no dollar leg, EURGBP say, the pip value is estimated and the report says its figures are indicative. The stop and the target fill inside the bar that reaches them, at their own level.

Why does my strategy give 0 trades?

Conditions are too strict or periods too long. Try shorter periods (SMA(10) instead of SMA(50)) or simplify the rules.

Does the genetic search overfit?

Yes, if you let it. The setup panel already asks for a minimum number of trades worked out from your history — between 20 and 120 — and holds the last stretch back unseen; under *Advanced settings* you can change *Min trades* and cap *Max DD%*. Then judge what it found in the Reality Check, which is told how many rule sets the search tried and asks more of the best of a big search.

Can I combine long and short strategies?

Yes, and it's recommended. In Portfolio, combine LONG with SHORT to reduce drawdown. The correlation matrix will tell you if they're actually diversified.

Where are my strategies saved?

In the Library, a SQLite database on this machine: the rules, the risk, the money management, the sizing, the lot and the starting balance. It also keeps the judge's last verdict: the score, the word, which history, which count of variants and when, and the figures from the stretch the strategy never saw — trades, PF, Sharpe and drawdown — which the list can be filtered by; it shows it on the strategy's card, and if you change the rules afterwards, it marks it as from an earlier version and says what changed: each condition added or removed, and each setting the engine uses with its before and after. The PDF report then describes the version that was judged. And it keeps what it took to reach it, the count of variants, which only goes up. Each one can carry notes and tags, and the list can be filtered and searched by tag. **Duplicate** makes a copy under another name to change without touching the original: it remembers which one it came from and takes its verdict and its count of variants along, because a copy does not start from zero. **Compare with...** puts two of your strategies side by side: how their rules differ, their figures on the loaded history with the same costs, and both equity curves on one chart. Those figures also cover the stretch the judge keeps apart, so to choose between the two, read their verdicts. Filtering on figures and picking with the filter set is charged: the one picked adds up the counts of every strategy compared, and the list says so before you pick. It keeps no trades — its figures are worked out when you press **Backtest**, on the history loaded then.

What does Dr. Quant do?

It answers questions, and it has two hands with a card. In the **Autopilot** conversation, “judge *name*” and “take *name* to the portfolio” are read by the program — a card says what will happen and nothing runs without your click, without spending your key. A question goes to your own AI provider with only what it names of your own work, summarised: the Library with its verdicts, a strategy you name, your data or your saved portfolios, and the verdict journal; your rules and your candles do not travel. And on the **Data** screen it looks up your broker's terms to fill in the costs. It builds nothing and runs nothing: building is the Autopilot's job, and it reads your request in code. Not financial advice.

Can I choose which AI provider Dr. Quant uses?

By their order in **Settings**: the active provider with the lowest priority number answers first. If it is saturated with a rate-limit error (429), the question passes on its own to the next active one.

Do free AI models charge me for internet searches?

The model is free; the **search is not**. A web search is a separate service your provider buys on your behalf and bills separately — OpenRouter states it plainly: web search incurs extra costs even with free models. Roughly **$0.007** per search at the default engine, about 70 cents for a hundred lookups. It comes off your own API key, not ours. **Reading a page you give a URL for is free** — BotFactoryFX's own server downloads it — so pasting your broker's contract-specs link costs nothing at all.

Why can't I pick a smaller timeframe than my file?

Because it cannot be built. Four M15 bars combine into one H1 bar, but no arithmetic splits one H1 bar back into four M15 bars — the detail was never in the file. Rather than invent it, the selector only offers your file's own resolution and coarser ones. Import a finer CSV if you need finer bars.

I changed the timeframe — does the backtest change too?

Yes. Changing the timeframe rebuilds the actual bars, and the chart, the backtest, the genetic search and the exported code all run on the rebuilt series. They can never disagree about which bars are under test. Earlier versions only changed the label, which is precisely the bug this replaced.

How do I scroll the chart back in time?

Drag it, hold **shift** and use the wheel, use the arrow buttons above the chart, or click the chart and use ← → (Page Up / Page Down for a full screen, Home and End for the two ends). The wheel on its own zooms rather than scrolls — that is the standard charting convention, and it is why the chart can look like it only zooms until you know about shift.

Can I export strategies to MetaTrader?

Yes. Open the download menu of a strategy in the **Library** — or press **Export…** in **Build › Manual**, which saves it and opens that menu — and choose **MQL4** (MetaTrader 4) or **MQL5** (MetaTrader 5). The file is a complete Expert Advisor: the entry rules, the entry order, the stop and the target, break-even, the trailing stop, the exits by time, the daily limits and the sizing, as the backtest ran them. Copy it into the `MQL4/Experts/` or `MQL5/Experts/` folder of MetaTrader's data folder, compile it, and attach it to a chart — step by step, with the settings the file adds, under *Installing the robot on each platform*.

Does the exported robot match the BotFactoryFX backtest?

As closely as each platform allows, and every file says at the top where it cannot. The entry rules, read on closed bars; pending STOP and LIMIT entries, expired after the same number of bars; the stop and the target; break-even and the trailing stop; the time-based exits; both daily limits, counted on the UTC day; and the three sizing modes — fixed lot, risk percent and Kelly — with the pip, the lot and the leverage of the instrument you measured (export with its history open): all of it is simulated by the backtest and carried by the exported file, which holds one position at a time, as the backtest does. **Exceptions today:** JForex sizes by risk or Kelly only on currency pairs, and trades only currency pairs, gold, silver, bitcoin and ether — on anything else it refuses to export; and the TradingView script sets no costs, so its totals leave out the spread, commission, slippage and funding the backtest charged. **What no file can carry** is what the simulation had to assume: your broker's real spread and slippage, partial fills, and a pending order that gaps past its level, which a broker fills at the market while the backtest filled it at the level.

Guided or Custom mode in Build › Manual?

Start with **Guided**: pick a family (Trend, Breakout, Reversion, Price Action, Momentum, Volatility, Divergence) and a template — the rules are pre-filled, you just tweak parameters. Switch to **Custom (Alchemist)** when you want to mix indicators across families without any guardrails. The redundancy checker runs in both modes.

Guided or Random in the genetic search?

Guided — the default, and the only mode of the setup panel — constrains the genetic search to the indicator and period pools of the families you tick, and seeds about half of the first generation from their templates, so it explores around rule patterns that mean something instead of testing indicators at random. **Random**, under *Advanced settings*, is restricted to no family and seeds from no templates. It is not the wider of the two: it draws from the pool shown under *Active indicator pool*, a short core — close, SMA, EMA, RSI, ATR, ADX and MACD — widened only at *Deep* effort. With several families ticked, Guided draws from far more indicators than that.

Can I mix strategy families?

Yes — tick as many as you like, in **Build › Evolve** and in the Wheel, or name several styles in the Autopilot's sentence. Some families complement each other: price action has no oscillators to get in anyone's way, and a breakout with a trend filter is a strategy neither family can write on its own, because session highs are not in the trend pool and ADX is not in the breakout pool. Mixing joins the indicator pool, the period pool and the templates the first generation is seeded from, and the seeding draws a family first and then one of its templates, so each ticked family gets an equal share rather than a share proportional to how many templates happen to have been written for it. It makes hybrids possible without forcing them, and a divergence stays whole in a mix as it does alone. **What it costs:** a wider pool means more distinct variants get tried, and the Reality Check demands more out-of-sample evidence for the same score the more candidates were tried — the line under the chips says how much the pool grew before you start. With all seven ticked the pool holds every built-in indicator the engine has; the seeding from templates still separates it from Random mode, whose pool is much shorter.

Why are some indicators flagged as redundant?

The live checker catches: (1) identical conditions in the same block, (2) the same indicator compared to itself (e.g. `SMA(20) > SMA(20)` — trivially false), (3) near-duplicate periods like SMA(20) and SMA(22), (4) mixing RSI + ROC (both momentum oscillators), (5) 3+ simple or exponential moving averages in one block, (6) LONG and SHORT entries that are identical, which fire on the same bar. These are *hints*, not errors — you can ignore them if you know what you're doing.

Metrics glossary

Profit Factor

`gross profit / gross loss`

> 1 = profitable, > 1.5 = good, > 2 = excellent. ∞ if no losses.

Win Rate

`winning trades / total × 100`

% of positive trades. Only useful with context.

Sharpe ratio

`(mean / stddev) × √(trades per year)`

Return adjusted by volatility, trade by trade, annualised by how often the strategy trades — not by a √252 that would assume daily bars. > 1 acceptable, > 2 very good. **Not measurable** when every trade returns the same — no losses, or every one at the same stop: with no spread there is no risk to measure, and a number there would be a division by almost zero. The search scores it as no edge.

Max Drawdown

`(peak - valley) / peak × 100`

Largest drop from a peak. < 20% reasonable, > 40% dangerous.

Return/DD (Calmar)

`Total P&L / Max DD`

Return per unit of risk taken.

Correlation

`cov(r_a, r_b) / (σ_a × σ_b)`

How two strategies move together. -1 opposite, 0 independent, +1 parallel.

Shortcuts and tips

- In Build › Evolve, each family chip toggles on its own: tick several to mix those families in one search.
- In the Library, the download menu saves a strategy for any of the six platforms or as JSON, and Import reads a JSON back.
- Questions go in the Autopilot's box: anything that is not a request to build is answered by Dr. Quant, on your own AI provider.
- In Portfolio, assign different weights to each strategy for custom allocation.
- If the genetic search stops improving, open Advanced settings and try another Mutation rate — but leave Min trades where the panel put it: it already asks for 20 to 120 trades from your history, and lowering it lets noise through.
- Crossovers (crosses above / crosses below) are evaluated against the previous candle.
- New to Build › Manual? Start in Guided mode, pick a family, then a template — you can switch to Custom (Alchemist) afterwards without losing the rules.
- Ctrl+K searches the whole program — sections, Library strategies, price histories and this Help — and takes you to what you pick; accents and capitals do not matter.
- In Build › Manual, Undo and Redo — or Ctrl+Z and Ctrl+Shift+Z — walk through your changes to the rules, Clear included; inside a text field, Ctrl+Z undoes what you are typing in it.

The content of this app is educational. It does not constitute financial advice. Trading carries risk of loss.

If something doesn’t work, reply to your purchase email or write to support@botfactoryfx.com.