QML Platform for Alpha Discovery and Strategy Construction (QML V1.1)

The Algebra of Alphas

QML Alpha LLC, April 8th, 2019

Alpha Discovery is a challenging research problem. To be competitive, hedge funds must consider diversification on several critical dimensions: data sources, methods of data processing and human intuition, expertise and creativity.

QML is a **Q**uantitative **M**achine **L**earning-oriented interactive platform for alpha discovery. Researchers could use **QML** to efficiently investigate ideas and discover new signals in datasets by applying time series transformations and Machine Learning algorithms. A special feature of **QML** is the high level of automation of the searching process that leads to a large number of discovered signals. Portfolio managers could use **QML** for the construction of strategies, by applying various signal aggregation techniques and hedging procedures based on risk models and classification systems.

1. QML Architecture

The central concept of **QML** is "Alpha" or "Alpha Factor". An Alpha is defined as a two-dimensional table with two indexes:

- **Dates** – represents the time dimension, typically daily for the analyzed time window
- **Universe** – represents the collection of assets described by their identifying systems, e.g., Cusip, Bloomberg, Barra

A few alphas have special meaning and they **should not be changed** by in-place transformations unless a specific modeling goal is formulated:

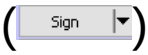
- **ret** – the return information (the reward associated to each period and each asset)
- **close** – the end-of-the-day prices
- **mask** – a filter indicating which assets are active on any given date; e.g., out of more than 6,000 assets with available data, one may choose to work with the most liquid 1,500 names. Customized universes of assets can be investigated by selecting appropriate mask filters
- **volume** – the daily volume in dollars

If the researcher wants to define a customized variant of any of these special Alphas (e.g., **ret**), he/she will have to configure the new vector. For example, **Return** could be customized to **factor neutralized return** or to **3-day return**, using the **Configure Tab**.

Additional fundamental or statistical factors, e.g., **size**, **value**, **growth**, can be uploaded in a similar format from additional in-house or third-party datasets.

Some datasets might contain missing data. QML operates consistently with missing data.

QML can apply two categories of **transformations** to one or more alphas:

- "**in-place**" transformations which change the original input data
- "**standard**" transformations which create a new alpha without changing the input information. Buttons associated to "**standard**" transformations are marked with a *down* triangle (

).

Transformations could be:

- **Cross-sectional** – applied for the **same date** to all or a subset of assets
- **Longitudinal, or Time series** – applied **asset-wise** for some past historical data (panel or rolling window)
- **Mixed** – combining both longitudinal and cross-sectional transformations

A standardized alpha is a $1 long - $1 short portfolio. An alpha can be transformed into its standardized form by applying two in-place operations:

- **DeMean** – cross-sectional, extract the mean of the vector from all its components
- **Normalize** – cross-sectional, divide each positive entry by the sum of all positive ones and apply similar transformations to the negative entries

For a standardized alpha, QML can compute performance statistics:

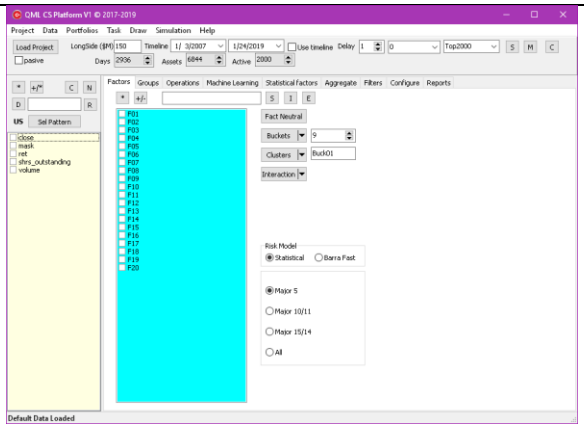
- **PnL** (Annualized return) – Daily returns equal the inner product between the daily vector with the daily vector of ret (returns), lagged by 1 day (for delay 0 mode), 2 day (for delay 1) or more. Average daily return is multiplied by 252 to produce annualized return
- **Annualized Volatility** – Standard deviation of daily returns is multiplied by $\sqrt{252}$ to obtained the annualized volatility (factor adjustment for time is $1/\sqrt{252}$)
- **Annualized Sharpe Ratio** – the ratio between annualized return and annualized volatility
- **Turnover** – the proportion of the alpha that has to be traded to change an existing position into its next period version
- **Max Draw Down** – the cumulative PnL drop from its peak (over the observed period)

QML has three layers of features:

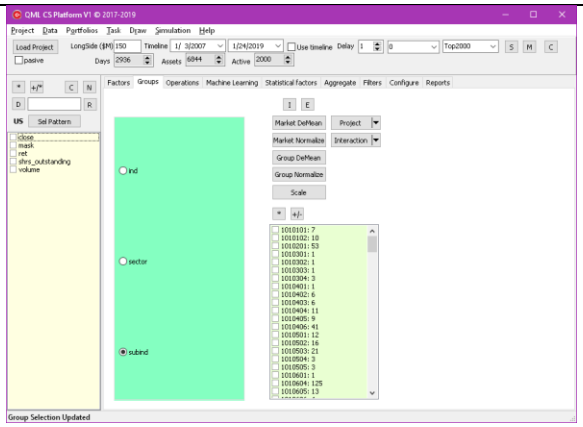
- **Design mode** – researchers/portfolio managers can easily investigate ideas, refine their findings, backtrack and try out different paths for investigation
- **Machine Learning mode** – various Machine Learning algorithms can be used to aggregate weak signals into stronger ones
- **Simulation mode** – the burden of repetitive, boring tasks can be passed to computers. Researchers define a **structures search space** and instruct computers to run extensive simulations with the goal of finding good signals that are hidden in the defined search space

2. QML GUI

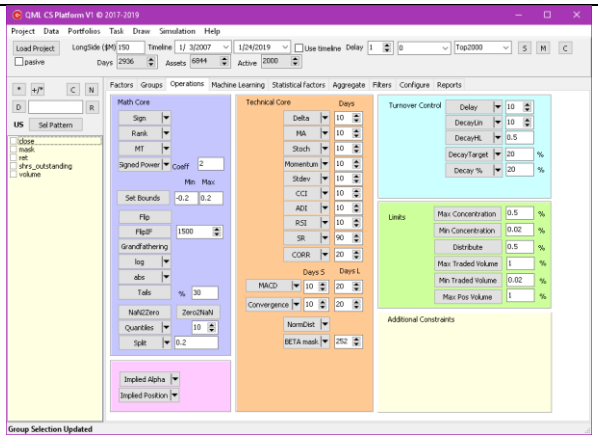
The GUI is organized in "**Tabs**" that correspond to particular type of actions. After loading the standard project, the major available tabs shown below:



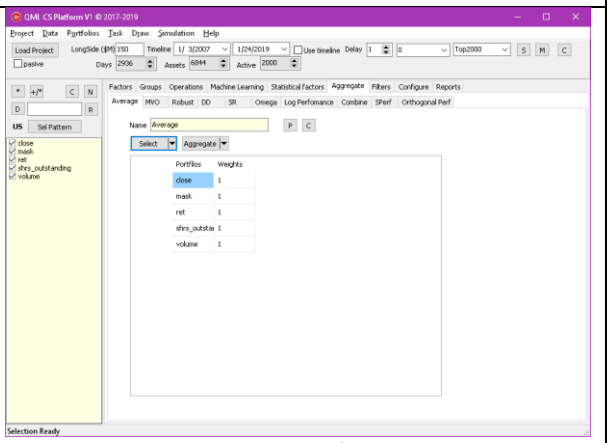
Factors



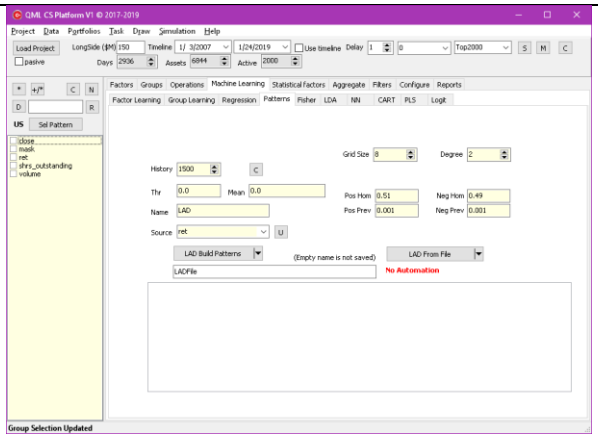
Groups



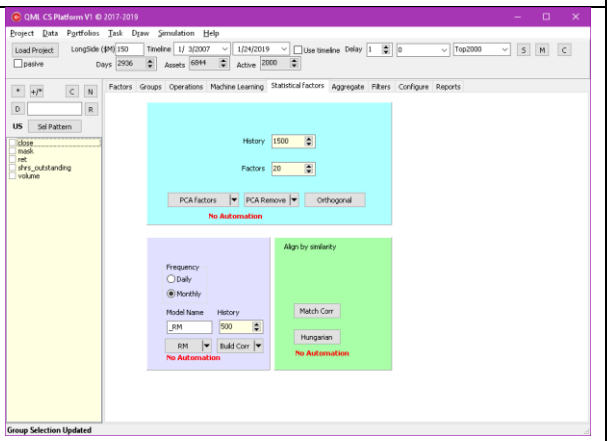
Operations



Aggregation



Machine Learning

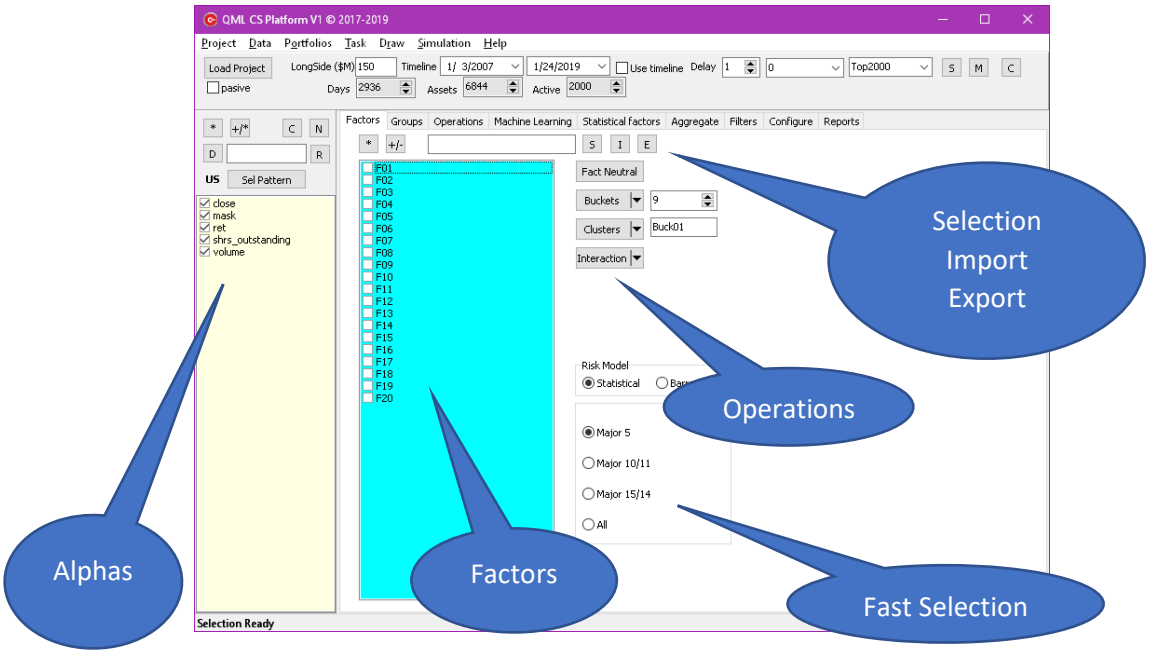


Statistical Factors

There are several additional tabs that are auxiliary, e.g., management, selecting and reporting. These additional tabs will be discussed later.

2.1. Factors Tab

The Factors **Tab** has several areas, as shown below:



Any operation is applied to the set of selected Alphas. Various parameters can be specified. Most of the operations and transformations are executed in parallel.

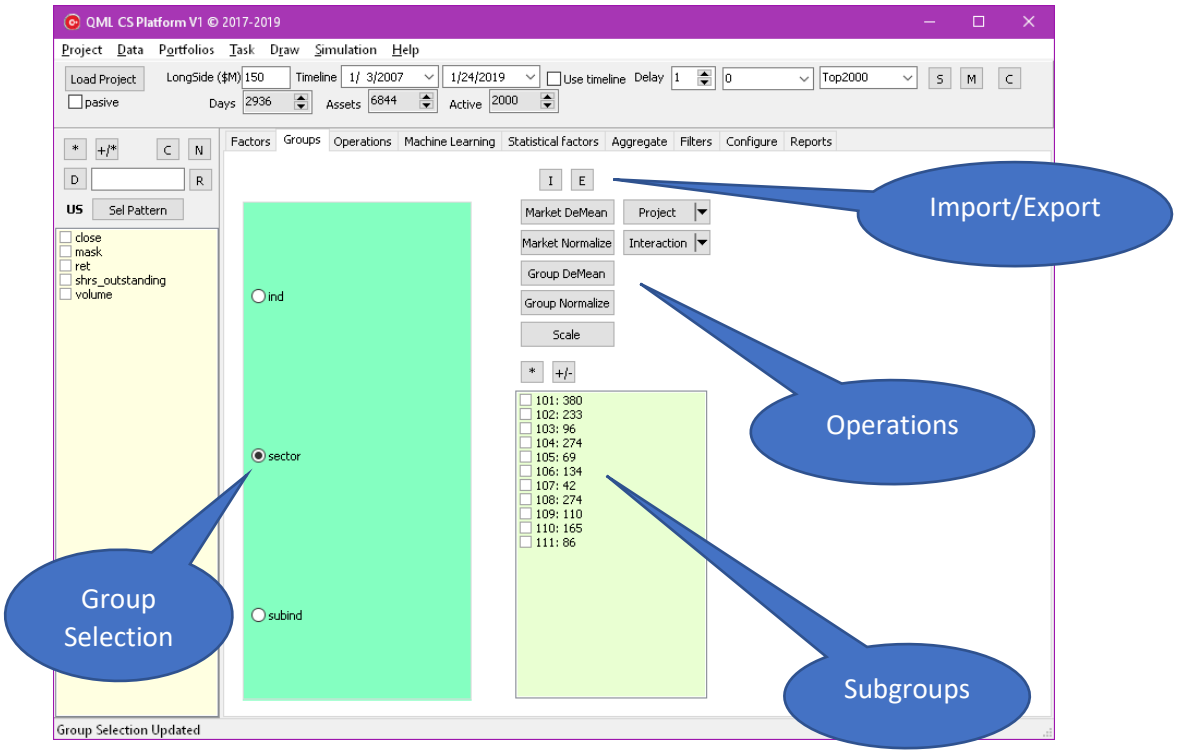
Operations producing new output are marked by a down triangle ( ) displayed next to the corresponding button. Here, "**Factor Neutral**" is an **in-place** operation (input Alphas are changed), and "**Interaction**" is a **standard** operation (it produces new Alphas and does not change the input data).

The **selection** controls (* +/- S I E) are self-explanatory. Hints - text appearing when mouse is hovering over - provide additional explanations.

The available **operations under the Factors tab** are:

- **Factor Neutral** – cross-sectional. Each selected Alpha is decomposed into its projection on the subspace spanned by the selected factors and by its corresponding orthogonal component. Then the alphas are replaced by their orthogonal components
- **Buckets** – cross-sectional. New groups are created based on the selected factors (*n*) and the specified number of buckets per factor (*b*). For instance, if *n*=2 and *b*=9, then a grouping with 81 = 9^2 categories is created. The result appears in the **Groups Tab**
- **Clusters** – cross-sectional. A ***k*-Mean** clustering algorithm is applied to the selected subset of factors to create groups that are coherent over time. The resolution parameter indicates the number of clusters to be discovered
- **Interaction** – creates *n* X *f* new alphas (Here, *n* = the number of selected alphas and *f* = the number of selected factors). The operation is the component-wise product

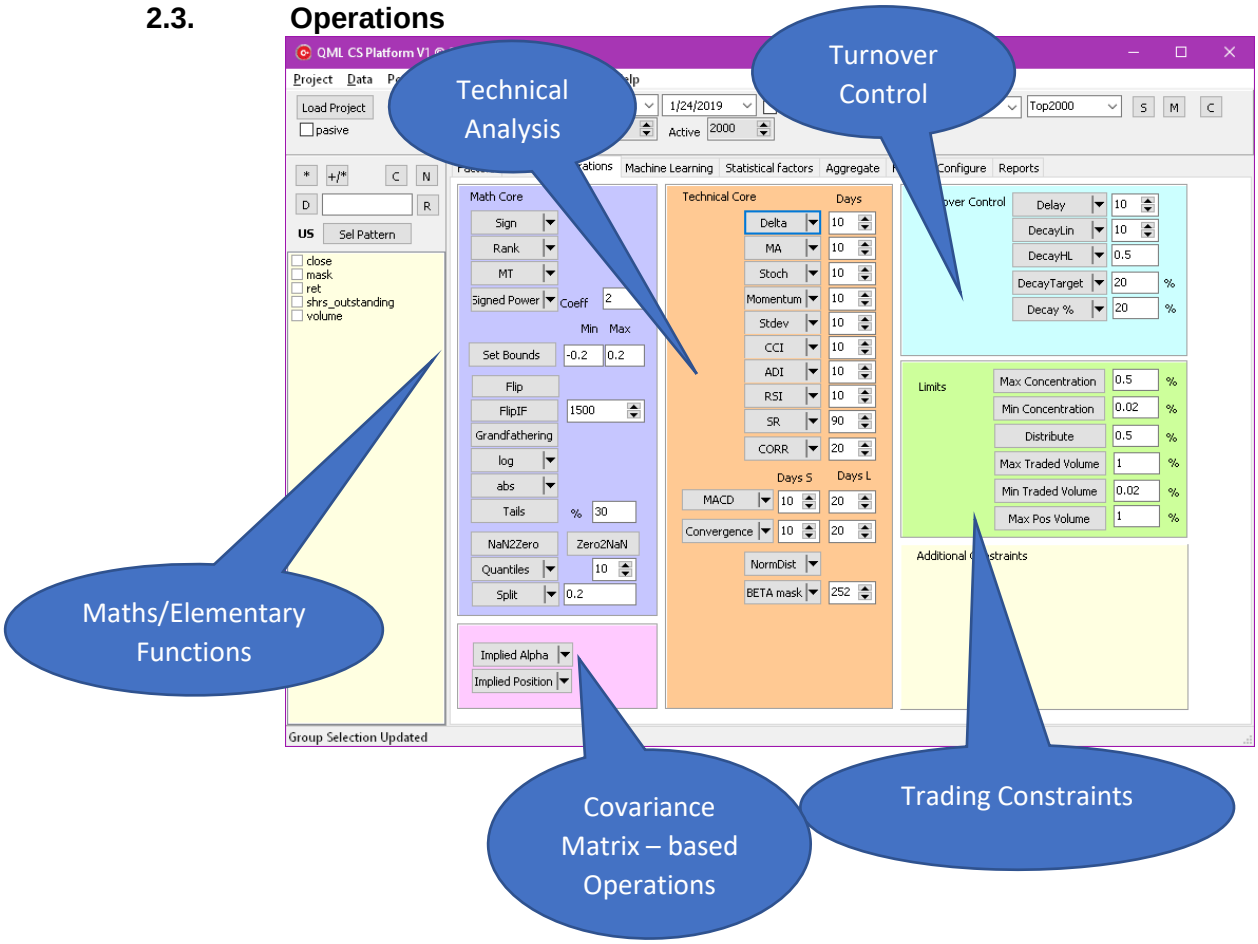
2.2. Groups



The **Groups Tab** operations are useful for hedging and for understanding group performance attribution:

- **Market DeMean** – in-place operation, cross-sectional. Subtract the averages from each component of the selected alphas. It creates dollar neutral, long-short alphas
- **Market Normalize** – in-place operation, cross-sectional. Convert the alpha to $1 long/$1 short portfolio
- **Group DeMean** – in-place operation, cross-sectional, subtract the group average from each component
- **Group Normalize** – cross-sectional. Convert the alpha to $1 long/$1 short portfolio, for each group
- **Scale** – cross-sectional, scale linearly the alphas to max $1 on a side (useful for alphas that are not dollar neutral: e.g., long only),
- **Project** – retain the values for the selected groups only; everything else is replaced by NAN values
- **Interaction** – split the alpha into its group components

2.3. Operations



- **Elementary Mathematical Transformations**
  - **Sign** – component wise sign of the input attribute
  - **Rank** – cross-sectional rank index; range = [0,1]
  - **MT** (market timing, longitudinal) – monthly based trend: if in January (over the entire history) the asset has a positive outcome, the result is 1, otherwise -1
  - **Signed Power** – component wise power of original, preserving the original sign; for instance, *SignPower*(-2,2) = -4
  - **Set Bounds** (in-place) – trim the input to the specified range
  - **Flip** (in-place) – change the sign component wise
  - **FlipIf** (in-place) – change the sign component wise, conditioned on PnL being negative over the specified history
  - **Grandfathering** (in-place, longitudinal) – replace NAN values with the latest known available data
  - **log** – compute a shifted logarithmic function component wise: $log_{10}(x + shift)$; *shift*=1 by default
  - **abs** – compute componentwise absolute value of the input
- **Technical Analysis**
  - **Delta** – Difference between today's value and *d*-day previous one, longitudinal
  - **MA** – Simple arithmetic average of the last *d* days, longitudinal
  - **Stoch** – Signal to noise ratio over last *d* days, longitudinal

- **Momentum** – percent change over last *d* days, longitudinal
- **Stdev** – standard deviation over last *d* days, longitudinal
- **CCI** – Commodity Channel Index, longitudinal
- **ADI** – Accumulation Divergence Index, longitudinal
- **RSI** – Relative Strength Index, longitudinal
- **SR** – Sharpe Ratio on rolling window, longitudinal
- **CORR** – Correlation with the return vector, longitudinal
- **MACD** – moving average convergence divergence index
- **Convergence** = bias measure between maximum and minimum values, longitudinal
- **NormDist** – forced normal distribution, cross-sectional
- **BETAmask** – beta values against with the universe equally weighted basket, cross-sectional

- **Turnover Control**
  - **Delay** – lagged version of the alpha, longitudinal
  - **DecayLin** – linearly decayed version, longitudinal
  - **DecayHL** – exponential (half-life) decay, longitudinal
  - **DecayTarget** – decay with total target, longitudinal
  - **Decay %** – limit asset turnover, longitudinal
- **Trading and Exposure Constraints**
  - **Max Concentration** – cap the weights to admissible threshold
  - **Min Concentration** – do not hold small position
  - **Distribute** – similar to max concentration, with redistribution, cross-sectional
  - **Max Traded Volume** – relative to budget, cross-sectional
  - **Min Traded Volume** – do not trade small quantities, cross-sectional
  - **Max Pos Volume** – do not hold more than you can execute fast, cross-sectional
- **Covariance Matrix-based Operations**
  - **Implied Alpha** – alpha is amplified by the covariance matrix of returns, cross-sectional
  - **Implied Position** – alpha is amplified by the inverse covariance matrix, cross-sectional

  These two operations are using risk model representation of the covariance matrix.

2.4. Aggregate

The **Aggregate Tab** allows the combination of several alphas into a new one. These methods are weighted averages based on the timeseries of PnL of the alphas, not on the weights at the asset level:

- **Average** – simple weighted average
- **MVO** – mean variance optimization, rolling window, maximizing the in-sample Sharpe Ratio, longitudinal
- **Stable Set** – select the alphas with prescribed performance, detect a maximal size subset of uncorrelated ones and construct simple average of the selection alphas, longitudinal
- **DD** – drawdown weighted scheme, longitudinal
- **SR** – Sharpe Ratio weighted scheme, longitudinal
- **Omega** – weighted scheme based on Omega Ratio, longitudinal
- **LogPerformance** – similar to SR, but uses logarithmic scale
- **Combine** – gradually change from one alpha to another one. This is useful to backtest strategies that are upgraded at given dates
- **SPerf** – apply scaling based on performance, longitudinal
- **Orthogonal Perf** – a compression of the alpha set using orthogonalization and residual performance. This transformation changes the input alphas and creates a new one

2.5. Machine Learning

Several **Machine Learning** algorithms implement discriminative and generative learning:

- **Factor Learning** – an alpha is decomposed into components using a collection of factors and then, an MVO method is applied to aggregate these components back into one alpha
- **Group Learning** – similar to Factor Learning, but the decomposition is based on groups rather than factors. The components can be reassembled by using various aggregation methods: (SR, MVO, Fisher, CART). Smart interaction will decompose the signal and retain only the components with prescribed performance
- **Regression** – linear regression can be used in panel or rolling window, with various frequencies (daily, monthly, yearly).
- **Patterns** – this method implements the pattern generation algorithms from Logical Analysis of Data (LAD). They represent rules for assessing the likelihood of return being positive or negative. The proportion of positive (or negative) patterns triggered defines the positive (or negative) score. Alpha is constructed as the difference between the two scores.
- **Fisher** – Linear Discriminant Analysis is used to detect the likelihood of returns being positive or negative
- **LDA** – similar to Fisher, but using a simple discriminant based on the gradient defined by the two centers of the positive and negative clusters
- **NN** – nearest neighborhood applied on a grid
- **CART** – classification and regression trees (node splitting is based on linear discriminants)
- **PLS** – partial least square method applied with limited number of iterations
- **Logit** – logistic regression

2.6. Statistical Factors

The transformations available from this tab are related to feature extraction, noise hedging, construction of statistical risk models and factor alignment:

- **PCA Factors** – construct a specified number of most significant principal components
- **PCA Remove** – remove top principal components of the alpha itself
- **Orthogonal** – transform the collection of selected alphas to become orthogonal (uncorrelated)
- **RM** – construct statistical factor risk models, with specified update frequencies
- **Build Corr** – build the factor-factor covariance matrix
- **Match Corr** – align alphas using their cross-sectional correlation
- **Hungarian** – align alphas using the Hungarian algorithm (maximum weighted perfect matching)

2.7. Administrative Tools

2.7.1. Filter Tab – tools to allow selection of alphas based on their performance statistics

2.7.2. Configuration Tab – allows definition of a new reward vector (**ret** by default) and customization of the universe mask

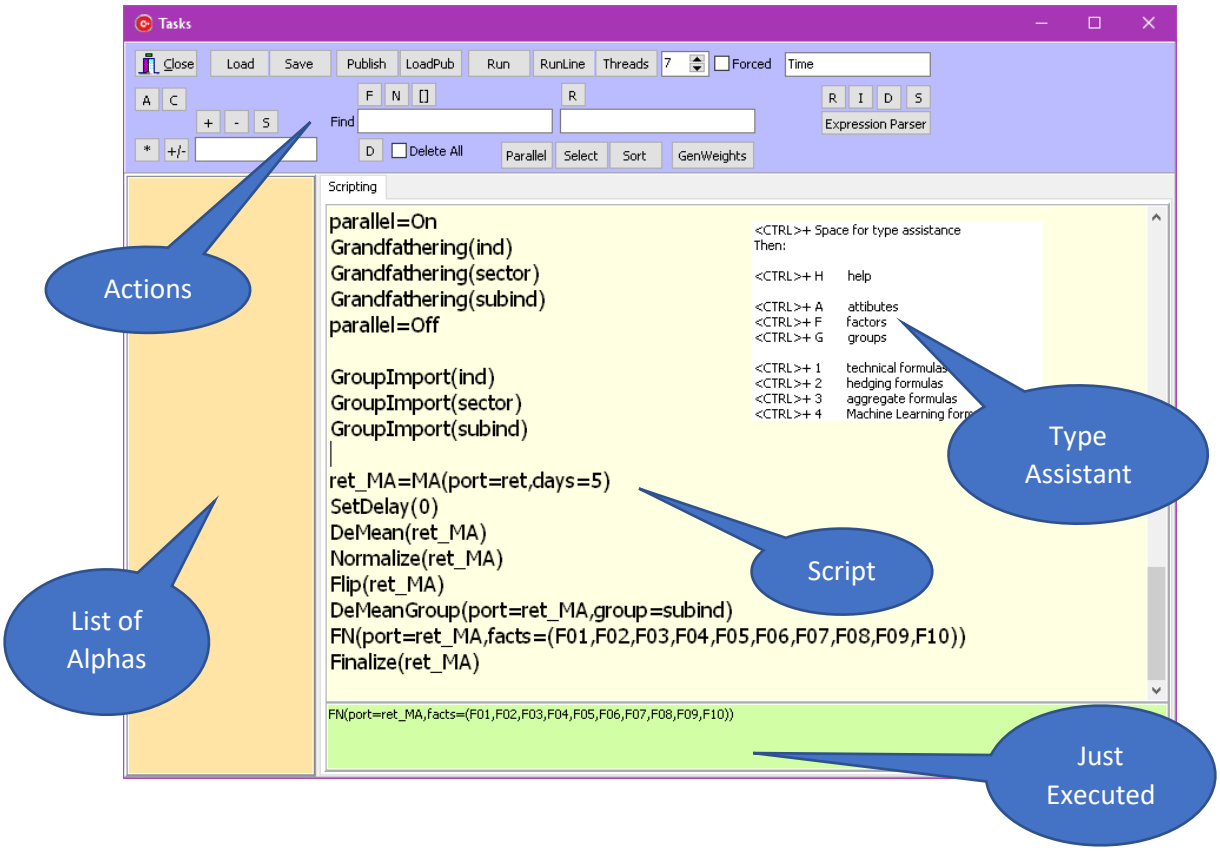
2.7.3. Report Tab – a collection of reporting tools:

a. Inspect alpha
b. PnL
c. Correlation (cross-sectional and longitudinal)
d. Turnover
e. Stats – performance statistics
f. Factor exposure report
g. Crossing
h. Netting

All these reports can be exported to Excel application or CSV files for further analysis.

3. QML Scripting

All actions performed by researchers are recorded in a special scripting language. The scripts can be executed again, rerunning the series of transformations that constructed the alpha model (feature needed for reproducibility and production).



The **Task Form** can be opened from the main *Menu/Task/Script*. The script can be saved and rerun later. It can be executed also on Linux using the server-side application. The script can be modified to change the processing flow.

3.1. Actions

- **Load** – Load script
- **Save** – Save script
- **Publish** – Save user encrypted format
- **LoadPub** – Load user encrypted format
- **Run** – run sequentially the commands in the selected text
- **RunLine** – run the line of the current position of the caret
- **Threads** – run the commands in the selected text in parallel (if parallel section is defined)
- **A** – analyze the script to create a checked list of portfolios that are needed. The list starts with alphas that are “Finalized” and then is expanded to include all dependencies
- **C** – clean the script by deleting commands that are not needed. As a precaution, it is recommended to save the script before this action

- **F** – find first occurrence of the text in the edit line below
- **N** – find next occurrence
- **[ ]** – find non-terminals (will be explained later in the simulation section)
- **R** – replace all occurrences from the corresponding edit line with the new text
- **D** – delete line, search for the next occurrence (if "Delete All" is checked it proceeds automatically till the end)
- **+, +/-** – select all and invert selection
- **+, -, S** – add, remove and select Alphas in the list
- **R, I, D, S** – **R**enumber "Alpha_xxx" to "Alpha_1"… , Import alphas from multiple scripts, **D**elete duplicates and **S**implify expressions
- **Parallel** – create a parallel section around the selected text. The commands in a parallel section will be executed as parallel threads, limiting the number of CPUs as prescribed in the parameter next to the ***Threads*** button
- **Select** – create a selection in the alpha list based on the occurrences of alpha names in the selected text
- **GenWeights** – recovere the weights for *Average* operation from the selected part of the script. This can be used only for the Aggregate/Average method
- **Expression Parser** – launch the expression analyzer

Type Assistant allows fast script editing with the correct syntax, and the following shortcuts are useful:

- **<Crtl>+H** – Help
- **<Crtl>+1** – Technical indicators
- **<Crtl>+2** – Hedging
- **<Crtl>+3** – Aggregation
- **<Crtl>+3** – Machine Learning
- **<Crtl>+A** – Attribute
- **<Crtl>+F** – Factors
- **<Crtl>+G** – Groups
- **<Crtl>+space** – Display the type assistant window

4. QML Simulations

This is a generative approach to alpha discovery. It allows the design of structured search spaces, making use of grammars (as in formal languages and compilers), using terms like: *terminals*, *non-terminals*, *productions* and *starting symbol* (**[StartSymbol]**).

Example. Generate sequences of characters having a number of **a**'s, one **b** and a number of **c**'s. Sequences like: **abc**, **aabccc** and **aaabcc** are valid, while sequences like **abca**, **abba** are not.

The grammar generating such language, using [S] as initial symbol, can be described as follows:

- **[S]** -> [A]b[C]
- [A]->[A]a
- [A]->a
- [C]->[C]c
- [C]->c

Note that non-terminals use square brackets, and terminals do not. A derivation is a sequence of replacements starting for the initial symbol and ending up with an expression containing only terminal symbols.

These "productions" mean that a non-terminal symbol can be randomly replaced by its right hand side in a certain production. Here, [S] (the start symbol) can be replaced by [A]b[C] (by first production). For example, the derivation of the word **aabccc**, could look like:

[S] => [A]b[C] => [A]ab[C] => aab[C] => aab[C]c => aab[C]cc => aabccc

A QML Example

Say that we want to investigate a large number of alphas that can be described as sums of products of two attributes. For convenience, we shall name these attributes A1, A2, …, A11.

A solution could be using the following grammar:

[Attr]=A1|A2|A3|A4|A5|A6|A7|A8|A9|A10|A11
[product]=[Attr]*[Attr]
[alpha]=[product]|[alpha]+[alpha]
[StartSymbol]=[alpha]

Note that terminals should be entities that can be evaluated by the backtest engine. A1, A2 must be either numbers or existing alphas, factors or groups.

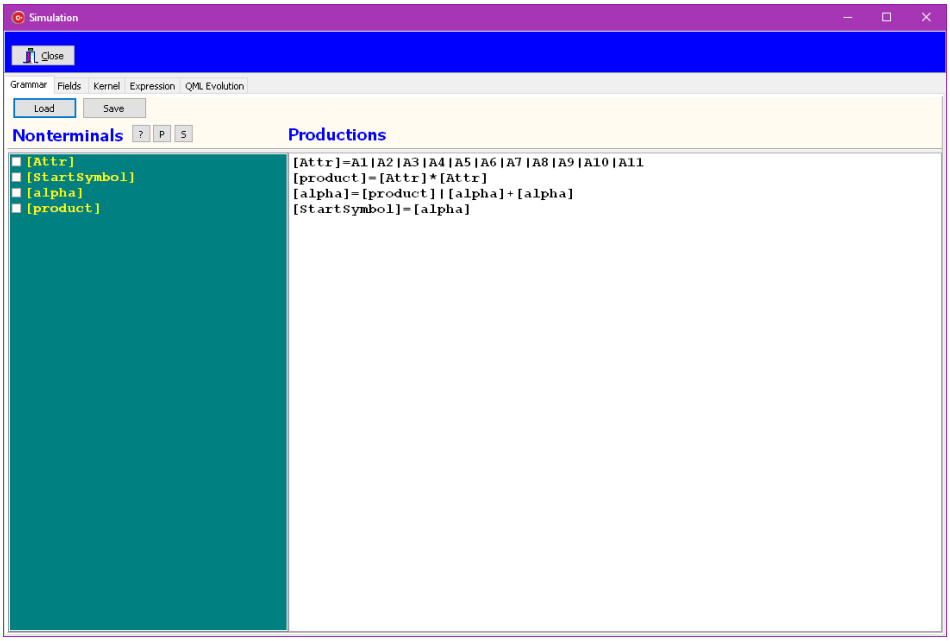
Grammars can be designed in the **Grammar Tab**.

Nonterminals are specified in square brackets, e.g., [Attr], [product], [Alpha] and [StartSymbol]. The [**StartSymbol**] nonterminal is the start symbol, therefore all grammars have to define this nonterminal. This is used to start any derivation.

The button "**?**" validates the productions of the grammar. It checks if any nonterminal needs further description. The button "**P**" adds the productions that need to be completed. The button "**S**" establishes the selected item as **[StartSymbol].**

The grammar is defined in the Grammar Tab and saved under the name "**test**".

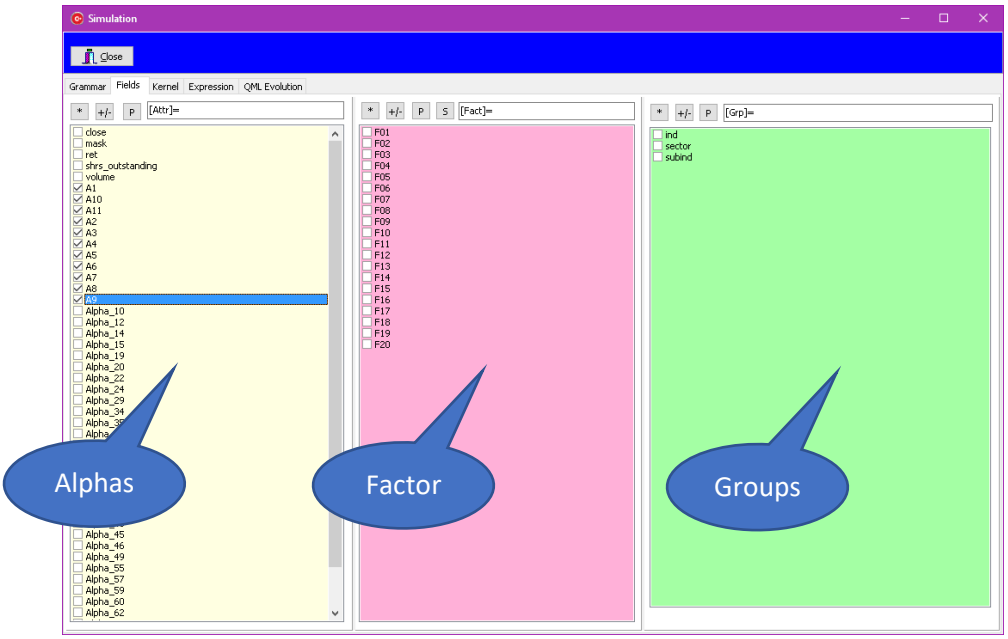
Grammars do not allow the use of spaces.



The **Fields Tab** can be used to generate production easy. For example the first production:

[Attr]=A1|A2|A3|A4|A5|A6|A7|A8|A9|A10|A11

can be generated from this Tab by selecting the fields A1 to A11 and pressing button "**P**".



Productions can be added for Alphas, Factors and Groups by pressing corresponding button “**P**”. Productions with the same left hand-side are generated togetther by specifying the right hand-sides as a pipe- (“**|**”) separated string.

The production

[Attr]=A1|A2|A3|A4|A5|A6|A7|A8|A9|A10|A11

is equivalent to the following 12 productions:

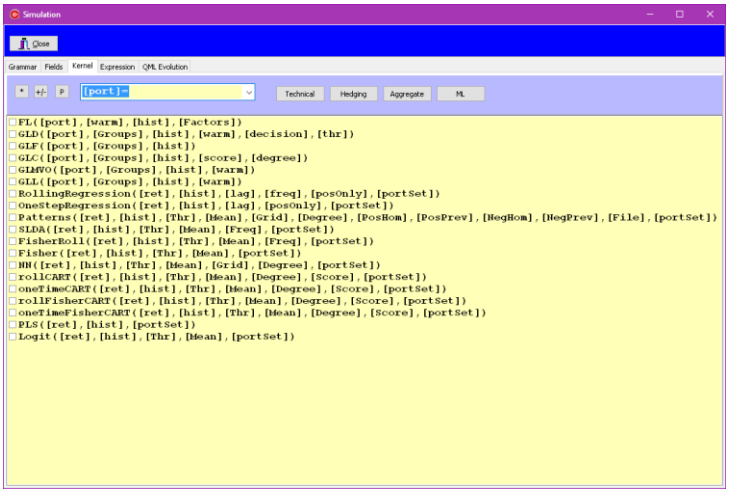
[Attr]=A1

…

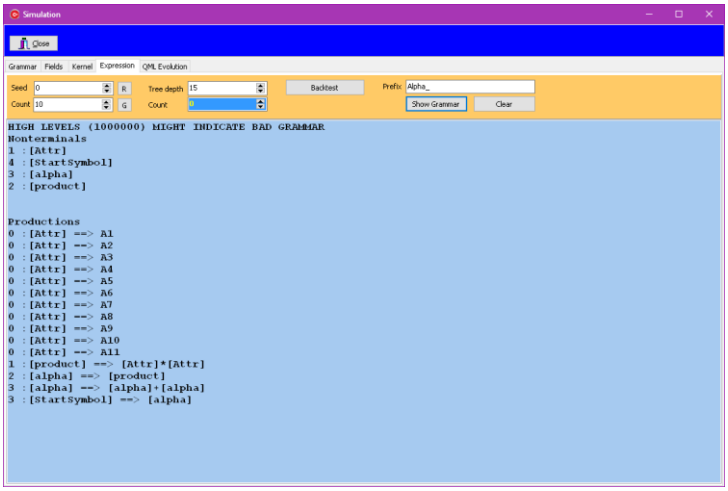
[Attr]=A11

The button “**S**” will generate a set of factors, as a comma separated string.

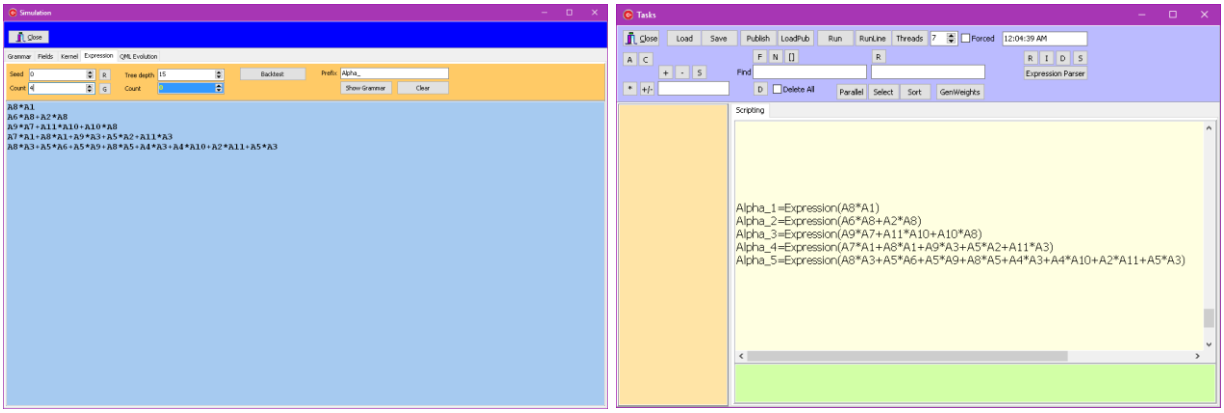
The **Kernel Tab** can be used to add functions from the kernel. For this example, no kernel function is used.



The **Expression Tab** can be used to validate the grammar (press “Show Grammar” button. If a negative number is observed at the beginning of some lines of the report, the grammar is not valid and needs to be fixed.



This tab can be used to generate expressions ("**G**" button) and send them to the script (press "**Backtest**" button).



In the script form the alphas can then be backtested.

Note that the grammar could generate expressions like:

A8*A1
A6*A8+A2*A8
A9*A7+A11*A10+A10*A8
A7*A1+A8*A1+A9*A3+A5*A2+A11*A3
A8*A3+A5*A6+A5*A9+A8*A5+A4*A3+A4*A10+A2*A11+A5*A3

To be evaluated they are wrapped in an **Expression** command and a name is defined automatically, like in the following example:

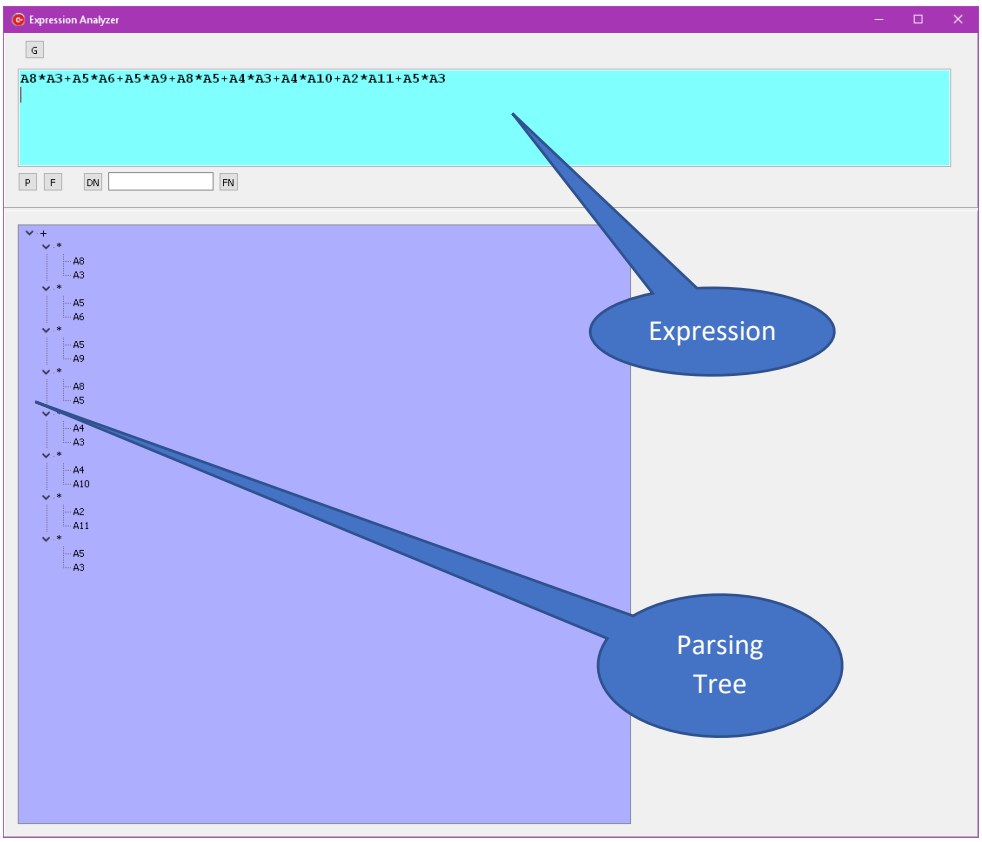
Alpha_1=Expression(A8*A1).

The backtest of these formulas, standardized and flipped is shown below:

An expression can be analyzed by launching the Expression Analyzer, and parsing it.

This feature is illustrated below for the following expression in the list above:

A8*A3+A5*A6+A5*A9+A8*A5+A4*A3+A4*A10+A2*A11+A5*A3

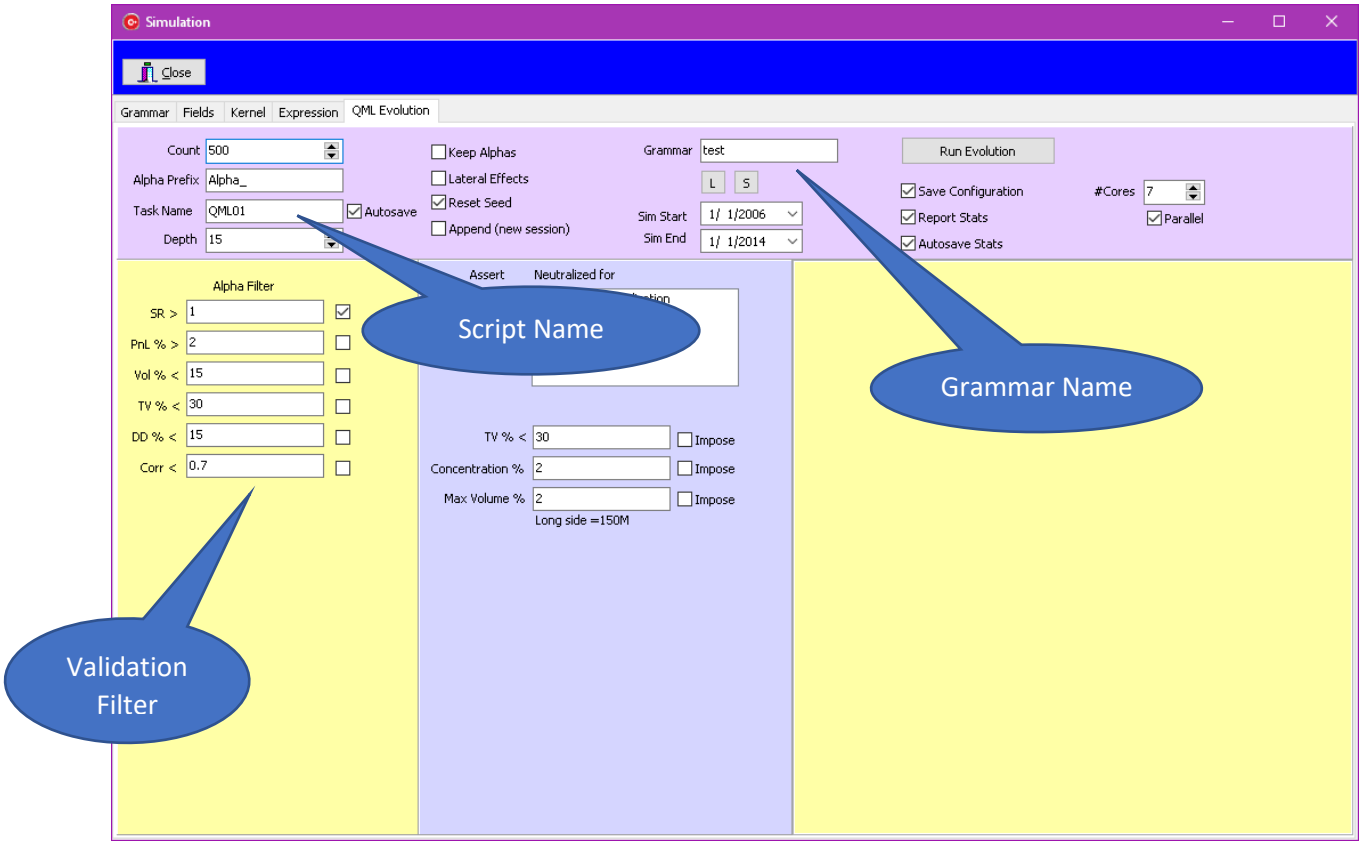


This information can then be used to improve grammar design.

A QML Simulation Experiment

The next step is to design a simulation experiment that generates many expression instances, backtest them, filter them, and report the meaningful results of the search.

We configure the simulation to run 500 experiments of this kind and report those instances that display Sharpe ratio above 1 and have pairwise correlation coefficients below 70%.



The parameters controling the simulation (generative alpha search) are described below:

1. **Generation Parameters**
    1. **Count** – sample size
    2. **Alpha Prefix** – used to create the name of the alpha (= prefix+index)
    3. **Task Name** – name of the task that initializes the simulation and will capture the findings. Check **Autosave** to save the task file after each update
    4. **Depth** – complexity of the samples (the depth of the parsing tree)
    5. **Keep Alphas** – if checked, alphas are kept in memory. This not recommended, as it could run into memory overflow problem if the number of findings is large
    6. **Lateral Effects –** allows the investigation of sub-formulas
    7. **Reset Seed** – reset seed to start a different simulation
    8. **Append (new session)** – assumes data is not loaded; will load and run the task and append new results
    9. **Grammar** – used to define the search space
    10. **L/S** – load/save the configuration file for the simulation
    11. **Sim Start/Sim End** – define learning period (in-sample)
    12. **Save Configuration** – configuration is saved before beginning the simulation
    13. **Report Stats** – the performance statistics report is generated

Autosave Stats – This report is saved automatically

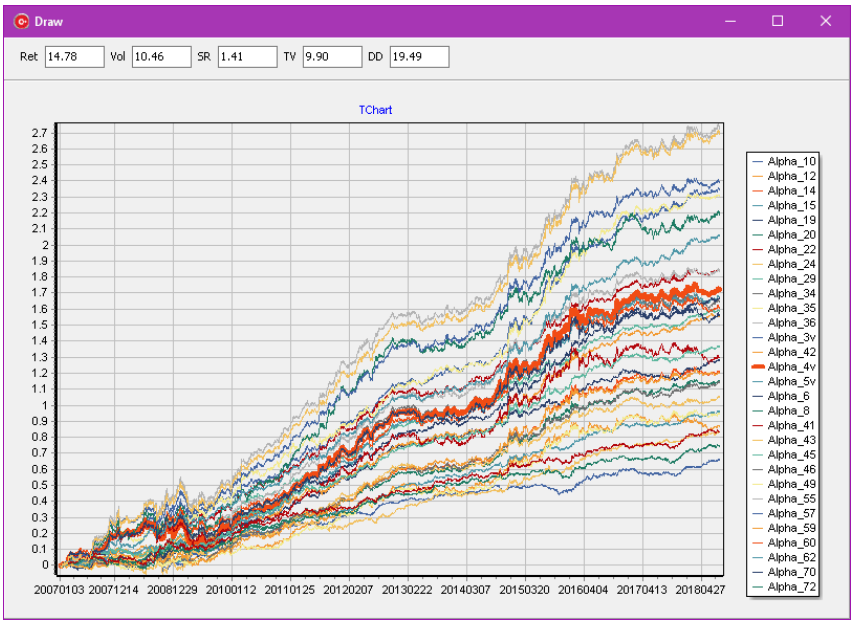
2. **Filter Parameters**
    1. **SR** – Sharpe Ratio
    2. **PnL** – Annualized return
    3. **Vol** – Annualized volatility
    4. **TV** – Daily turnover
    5. **Corr** – pairwise correlation coefficients of PnL timeseries
3. **Post-processing parameters**
    1. **Neutralized for** – neutralization is applied to generated alphas
    2. **TV (impose)** – alphas are decayed to prescribed turnover
    3. **Concentration (impose)** – impose limited concentration
    4. **Max Volume (impose)** -impose limited trading

The run of this simulation produced 32 Alphas (partial descriptions):

parallel=On
Alpha_4v=Expression(Standard(Flip(A1*A1)))
Alpha_5v=Expression(Standard(Flip A7*A6+A1*A11)))
Alpha_3v=Expression(Standard(Flip(A2*A10+A10*A9+A11*A9)))
Alpha_8=Expression(Standard(Flip(A2*A10)))
Alpha_10=Expression(Standard(Flip(A1*A9+A2*A2+A3*A7)))
Alpha_12=Expression(Standard(Flip(A1*A9+A1*A10)))
Alpha_6=Expression(Standard(Flip(A7*A6+A1*A2+A11*A7+A9*A5+…
Alpha_14=Expression(Standard(Flip(A3*A6+A9*A5+A11*A6)))
Alpha_19=Expression(Standard(Flip(A1*A2+A10*A8+A11*A3)))
Alpha_15=Expression(Standard(Flip(A1*A1+A10*A9+A11*A9+A10*A9+…
Alpha_22=Expression(Standard(Flip(A4*A9+A2*A11+A8*A5)))
Alpha_20=Expression(Standard(Flip(A5*A3+A8*A2+A5*A8+A5*A4+A2*A4+…
Alpha_24=Expression(Standard(Flip(A5*A4+A3*A1+A1*A6+A5*A5+…
Alpha_34=Expression(Standard(Flip(A1*A6)))
Alpha_35=Expression(Standard(Flip(A2*A8+A9*A2+A3*A2+A2*A6)))
Alpha_29=Expression(Standard(Flip(A11*A2+A3*A1+A7*A8+A3*A4+…
Alpha_36=Expression(Standard(Flip(A1*A3+A11*A5+A10*A11+A6*A2)))
Alpha_42=Expression(Standard(Flip(A1*A2+A9*A5+A1*A5+A6*A7)))
Alpha_41=Expression(Standard(Flip(A1*A6+A10*A6+A11*A10+A6*A3+…
Alpha_43=Expression(Standard(Flip(A9*A6+A11*A9+A1*A3+A1*A6)))
Alpha_45=Expression(Standard(Flip(A1*A3+A6*A6+A1*A8+A3*A10+…
Alpha_46=Expression(Standard(Flip(A5*A2+A6*A4+A1*A8+A6*A1+…
Alpha_55=Expression(Standard(Flip(A6*A2)))
Alpha_49=Expression(Standard(Flip(A11*A10+A7*A10+A7*A5+A9*A5+…
Alpha_59=Expression(Standard(Flip(A9*A3+A6*A7+A5*A4+A11*A1)))
Alpha_60=Expression(Standard(Flip(A8*A2+A11*A6+A5*A2)))
Alpha_57=Expression(Standard(Flip(A8*A11+A9*A10+A9*A1+A1*A1+…
Alpha_62=Expression(Standard(Flip(A5*A9+A2*A6+A10*A3+A3*A8+…
Alpha_70=Expression(Standard(Flip(A11*A6)))
Alpha_72=Expression(Standard(Flip(A8*A11)))
Alpha_75=Expression(Standard(Flip(A8*A1)))
Alpha_84=Expression(Standard(Flip(A2*A2+A1*A6+A11*A2+A7*A6)))
parallel=Off

The backtest results are shown in the following table and plot.

Alpha_Name	SR	AnnRet	AnnVol	Daily TV	Max DD
Alpha_10	1.99	20.10%	10.11%	16.40%	12.11%
Alpha_12	1.03	7.48%	7.28%	18.42%	10.98%
Alpha_14	1.28	10.38%	8.13%	21.28%	9.01%
Alpha_15	1.54	8.29%	5.38%	20.63%	8.56%
Alpha_19	1.88	10.95%	5.84%	12.27%	8.09%
Alpha_20	1.49	9.85%	6.61%	18.78%	6.37%
Alpha_22	1.87	15.75%	8.42%	19.39%	8.42%
Alpha_24	1.49	9.01%	6.05%	16.11%	7.26%
Alpha_29	2.25	13.65%	6.06%	16.12%	5.42%
Alpha_34	1.32	14.14%	10.73%	9.09%	17.92%
Alpha_35	1.91	19.82%	10.35%	19.53%	10.23%
Alpha_36	1.69	15.74%	9.30%	16.56%	12.25%
Alpha_3v	1.25	5.69%	4.56%	18.58%	10.30%
Alpha_42	1.76	13.38%	7.59%	20.45%	7.83%
Alpha_4v	1.41	14.78%	10.46%	9.90%	19.49%
Alpha_5v	1.6	14.38%	9.00%	10.39%	9.36%
Alpha_6	1.91	14.33%	7.49%	12.87%	7.69%
Alpha_8	1.47	18.78%	12.77%	9.34%	17.45%
Alpha_41	1.07	11.11%	10.36%	9.38%	14.54%
Alpha_43	1.26	7.19%	5.72%	22.80%	10.42%
Alpha_45	1.83	11.71%	6.39%	15.41%	7.24%
Alpha_46	1.49	9.84%	6.59%	14.99%	6.01%
Alpha_49	1.21	8.12%	6.70%	19.25%	7.45%
Alpha_55	1.81	23.37%	12.88%	8.93%	19.55%
Alpha_57	1.88	20.51%	10.90%	15.30%	16.26%
Alpha_59	1.35	10.30%	7.62%	20.30%	8.57%
Alpha_60	1.38	13.81%	9.98%	13.34%	16.29%
Alpha_62	2.05	17.67%	8.63%	15.12%	9.06%
Alpha_70	1.26	13.36%	10.63%	8.60%	15.43%
Alpha_72	1.24	6.43%	5.18%	7.96%	10.57%
Alpha_75	1.35	7.19%	5.34%	8.97%	8.45%
Alpha_84	1.86	23.08%	12.43%	7.79%	19.45%



A final aggregation of all Alphas found by this experiment could lead to an alpha model to be submitted to production. If the alpha model is good enough and has low correlation to the existing alphas, it should add value to the pool of alphas already in production.

For this example, applying the PLS aggregation method to these alphas generates the following alpha model:

Alpha Name	SR	AnnRet	AnnVol	Daily TV	Max DD
PLS	1.7	22.63%	13.32%	15.48%	21.30%