Data Context Map Quick Start Guide.
version 1.0 (*Beta*)

Eric Papenhausen (epapenha@akaikaeru.com)
Klaus Mueller (mueller@akaikaeru.com)
https://akaikaeru.com

October 16, 2020

Introduction

The data context map (DCM) finds and visualizes statistically significant and temporally consistent patterns. For a data set consisting of stocks, for example, these patterns represent patterns of stock behavior (e.g. price/book ratio < 1) that are associated with unusually high or low returns. In the case of a data set with a temporal component (e.g. stocks in the NYSE over time), the patterns found are temporally consistent (i.e. they are consistently high/low over time). The DCM is implemented as a plugin to Jupyter notebook to allow for easy integration with existing data science workflows and to allow insights found within the DCM to be exported and analyzed further.

Contents

1 Prerequisites

The data context map requires python 3.7 or higher and the python `pip` package manager. The software also requires several open source packages. See the `requirements.txt` file for more detail.

2 Installation

The following installation procedure has been tested on Windows 10. Email epapenha@akaikaeru.com for any issues or questions about the installation process.

Begin by downloading either the 32-bit (data_context_map_win32_37.tar.gz) or 64-bit version (data_context_map_win64_37.tar.gz) of the data context map. The version to install depends on whether you plan to use 32-bit or 64-bit python.

2.1 Virtual Environment (Optional)

Python packages for different projects can be managed separately using virtualenv. To install virtualenv execute the following command in the terminal:

```
python -m pip install --user virtualenv
```

To create a virtual environment, navigate to the project directory and execute:

```
python -m virtualenv env
```

Then activate the virtual environment by executing:

(In Linux)

```
source env/bin/activate
```

(In Windows)

```
env\Scripts\activate
```

The terminal prompt should include `(env)` as a prefix.

2.2 The Data Context Map

Begin by extracting the data context map tarball within the project directory:

```
tar -xvzf data_context_map.tar.gz
```

Navigate to the data_context_map directory and install the required python packages.

```
pip install -r requirements.txt
```

Next, execute the following command in the terminal to install the python wheel file.

(32-bit)

```
pip install data_context_map-0.1.0a0-cp37-cp37m-win32.whl
```

(64-bit)

```
pip install data_context_map-0.1.0a0-cp37-cp37m-win_amd64.whl
```

Enable the data context map as a custom widget within Jupyter notebook.

```
jupyter nbextension install --py --sys-prefix data_context_map
jupyter nbextension enable data_context_map --py --sys-prefix
```

3 Overview of Features

3.1 Getting it to Work

There are three sample Jupyter notebooks that show how to operate the data context map. These are in the `sample` folder in the `data_context_map` directory. There are two modules that can be imported within a Jupyter notebook cell. The first is the `pattern_miner` module:

```
import data_context_map.pattern_miner as pm
```

This is needed for the multi-variate pattern miner. It is initiated by calling:

```
out = pm.DataContextMap(df, target)
```

where `df` is a pandas dataframe and `target` is the target attribute of interest. The data context map visual interface is then rendered by calling:

```
out.render()
```

The second is the `correlation_miner` module:

```
import data_context_map.correlation_miner as cm
```

It is initiated by calling:

```
ctab = cm.CorrelationTable(df, target)
```

where `df` is a pandas dataframe and `target` is the target attribute of interest. The visual interface is then rendered by calling:

```
ctab.render()
```

3.2 Pattern Mining

The data context map finds and visualizes patterns. Patterns are hypercubes of the form 'attribute' (<,>,=) 'value' (e.g. price/book < 1 and sector = Financial). Two types of pattern mining are supported within the data context map – numeric and binary. Numeric pattern mining is used when the target variable is continuous (e.g. returns). Binary pattern mining is used when the target variable is a 0 or 1 indicator (e.g. defaulting on a loan).

In both cases, patterns are found to be "interesting" when the pattern satisfies the following conditions:

1. The target variable within the pattern is statistically significantly higher or lower than the rest of the data set.
2. The effect size is large (i.e. higher than some predefined threshold)
3. The size of the pattern is large (i.e. higher than some predefined threshold)

3.2.1 Numerical

The statistical test performed with numeric pattern mining is the non-parametric Mann-Whitney U test. This is a non-parametric test and so it makes no assumption about the distribution of the target variable. The effect

size used is the common language effect size. This is a measure of the probability of an item selected from the pattern being higher / lower than an item selected from outside the pattern. For example, an effect size of 0.8 would indicate that if we were to randomly select one point within the pattern and one point outside the pattern, 80% of the time the point within the pattern will be higher. Negative effect size measures indicate the opposite relationship (e.g. -0.8 indicates that 80% of the time the point within the pattern is lower).

3.2.2 Binary

For pattern mining with a binary target variable, the target attribute is assumed to consist of 1's and 0's. The statistical test performed with binary pattern mining is the chi squared test for independence. This determines if the number of 1's within the pattern is statistically higher than the overall data set. The effect size used is the odds ratio. An odds ratio of 2 indicates that the odds of the target being a 1 within the pattern increase by a factor of 2x compared to the overall data set.

3.3 List Interface

Figure 1 shows the initial output of the data context map (DCM). The data set being analyzed consists of stocks over a 12 month period in 2016. The initial output consists of a list of patterns containing useful information like the distribution of returns within the pattern and how this pattern changes over time. As an added convenience, clicking on the table header will sort the list by the corresponding criteria. The user will then select some subset of interesting patterns for further investigation via the DCM's graph interface.

3.4 Graph Interface

The graph interface is generated once the user selects the set of patterns to investigate further from the list interface. Figure 2 shows the graph interface. The data set is the same as that of section 3.3. Each node represents a pattern selected from the list interface (see the caption of figure 2 for more detail).

The interface of figure 2 shows the state of the patterns at a specific time period (indicated in the upper left corner). At the top of this interface is a slider which can be used to change the time period of interest. This updates the size and color of the nodes to indicate the state of each pattern (see figure 3).

Left clicking on a node brings up the pattern detail view (see figure 4). This view includes more detailed information about the clicked pattern. It includes a list of the attribute ranges that define the pattern. It also includes a numerical confidence value for each attribute. This value gives insight into how discriminative each attribute is. In figure 4(a), for example, The 35.9% confidence value in the "on_bal_vol" attribute indicates that, of all the data points within the on_bal_vol attribute range, 35.9% of them are in this pattern. Adding the additional constraint of "rsi" $\geq$ 54.99 brings the confidence value to 100% – indicating that 100% of data points within these two attribute ranges are in this pattern.

The "stats" view on the right of figure 4 shows descriptive statistics for this pattern. This includes the size of the pattern (i.e. number of data points within this pattern), as well as the min, max, mean, and standard deviation of the target variable within this pattern. Also included are the p value and the common language effect size (i.e. Effect Size). Clicking on the button at the top right of the "stats" view will export the data within this pattern to a pandas dataframe. For the pattern in figure 4, for example, this will export the 142 data points within this pattern to a separate dataframe which can be retrieved by calling the `get_selected_df()` function (see section 4.3).

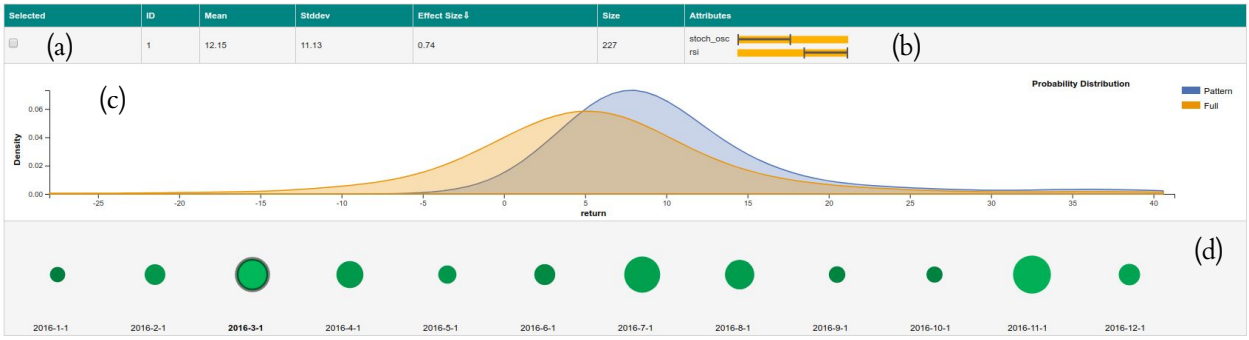


Figure 1: The first pattern from a list of patterns produced by the DCM. (a) Useful fields including descriptive statistics about the pattern. (b) List of the attributes defining the pattern along with a contextual range for each attribute. For example, this pattern is defined by low "stoch_osc" and high "rsi." (c) View showing the distribution of returns for stocks in the pattern (blue) v.s. the market (orange). (d) Time stream view in which each circle corresponds to a different month. The size of the circle is based on the size of the pattern and the color is based on returns for that month.

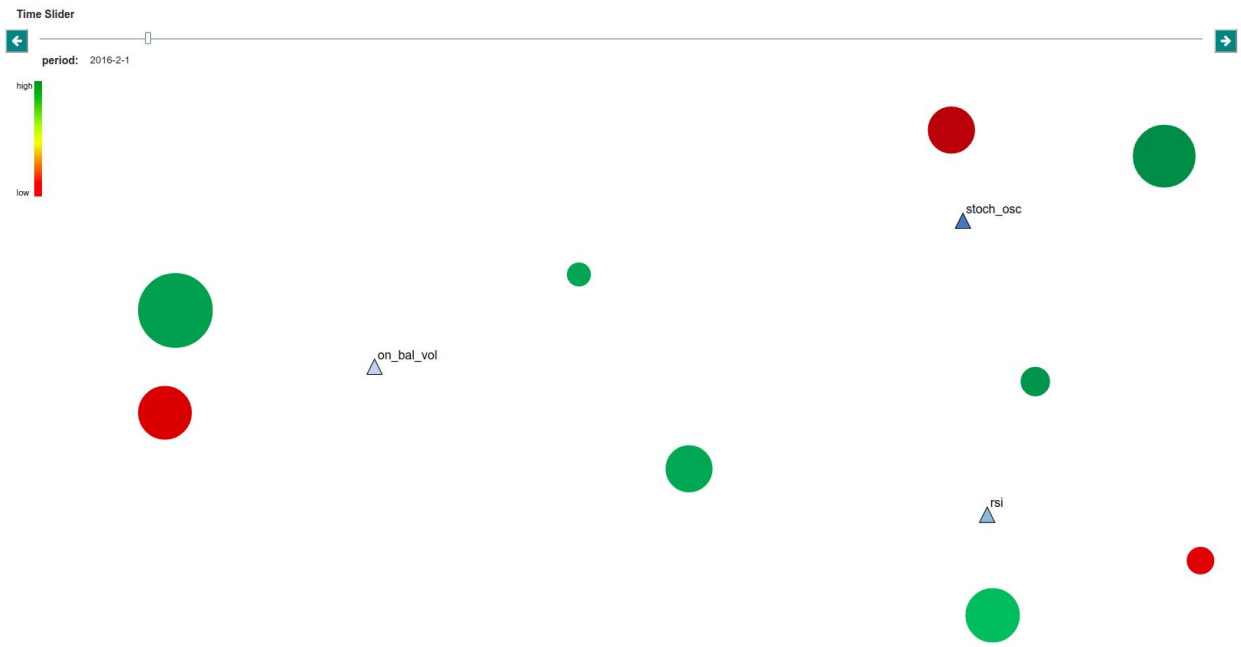
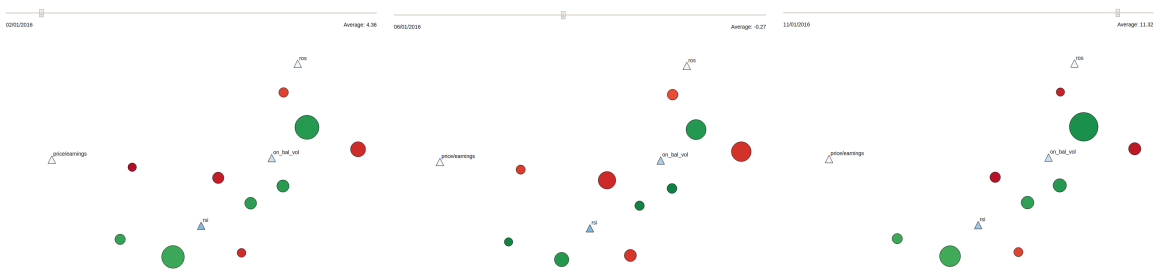
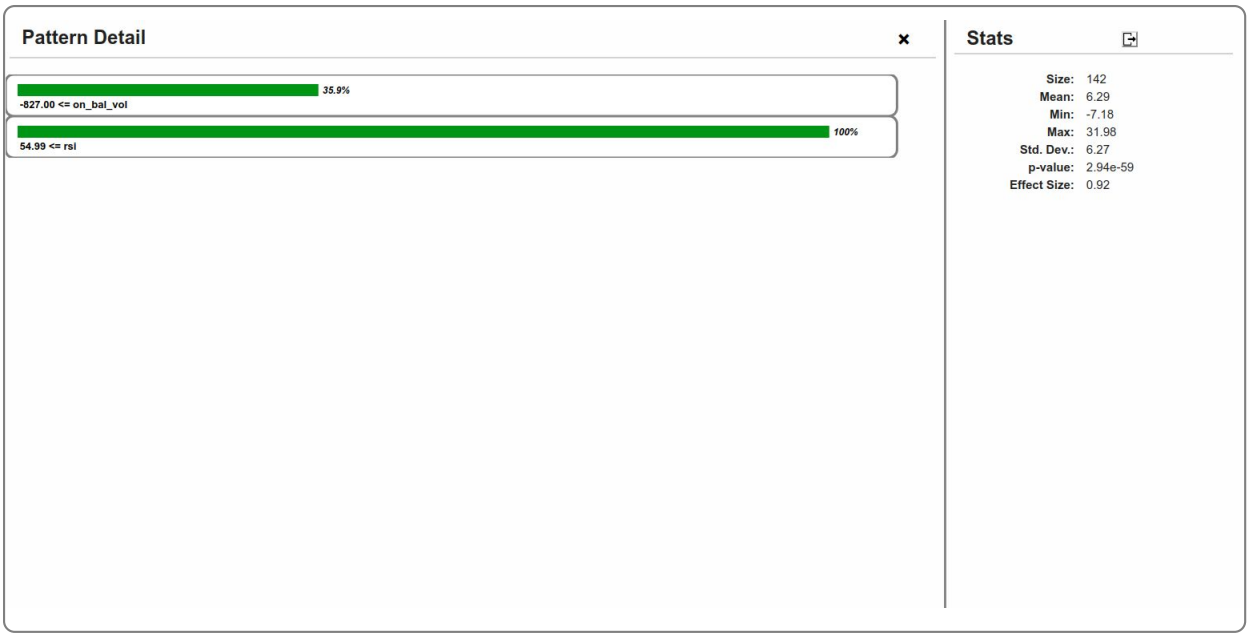


Figure 2: In the graph interface, nodes represent patterns – with size being mapped to the size of the pattern, and color representing returns (i.e. green for high and red for low returns). The labeled triangles represent attributes. Each node is placed by the attributes which define the pattern it represents. For example, the node between "rsi" and "on_bal_vol" represents a pattern defined by these two attributes. The time slider at the top allows the user to change the date (shown in the upper left).

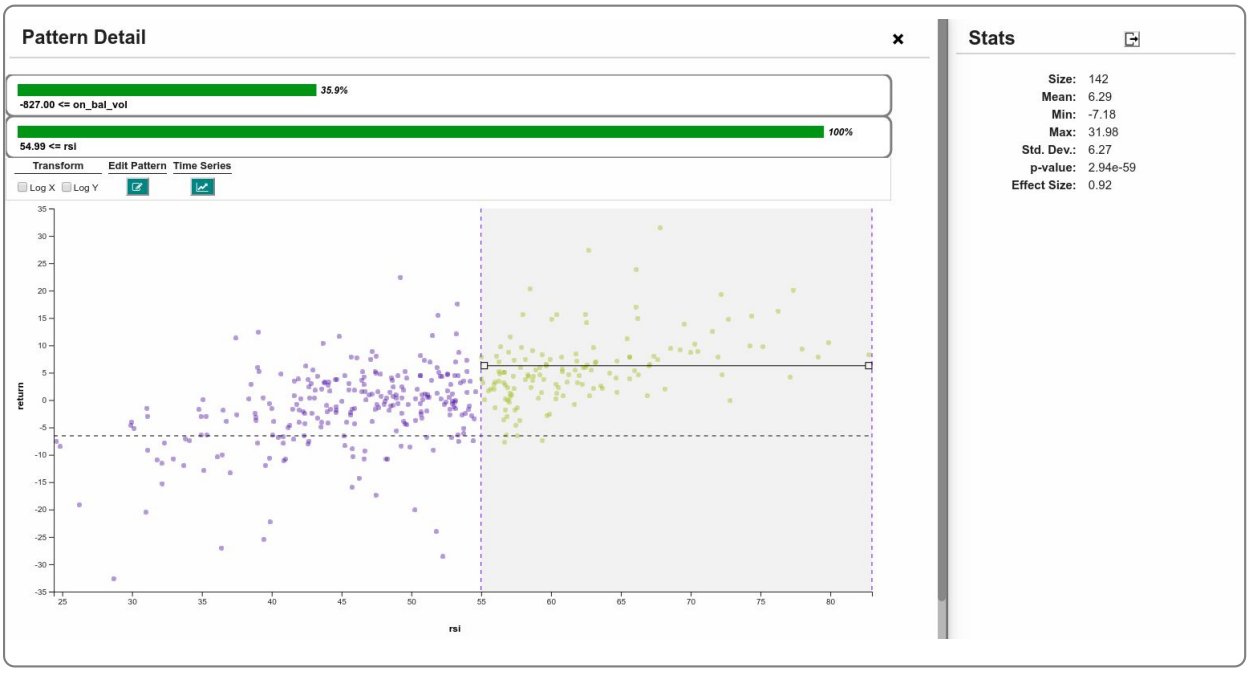


(a) February (b) June (c) November

Figure 3: The graph interface for the same patterns at different months. Note the size and slight color changes to the patterns over time.



(a)



(b)

Figure 4: The Pattern detail view shows a list of attribute ranges that define the pattern along with the confidence values (a). Clicking on an attribute shows a 2D scatterplot view, (b), with the x-axis being the clicked attribute and they y-axis being the target variable. A gray transparent box is also shown with the left and right bounds of the box matching the attribute ranges of the clicked attribute. Yellow points in the scatterplot represent data points within the pattern, while blue points are data points outside the pattern.

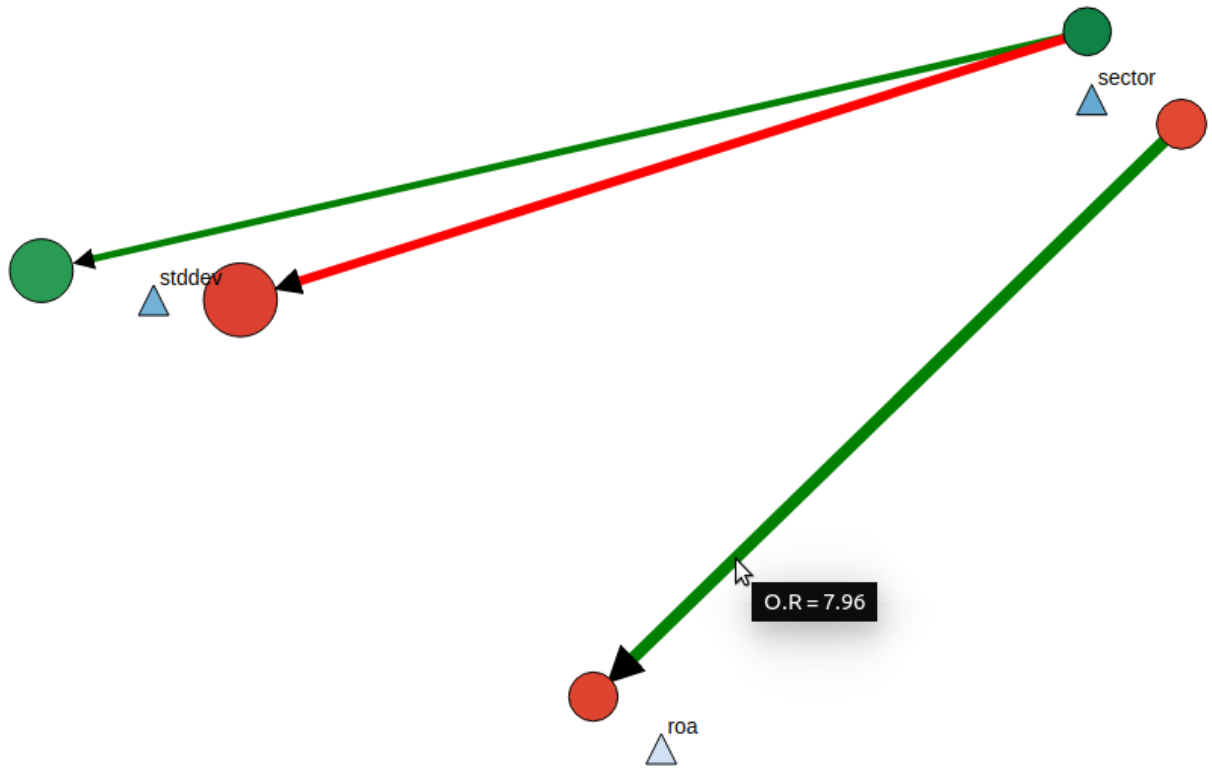


Figure 5: Green and red directed edges show positive and negative causal relationships respectively. The width of an edge indicates the strength of the causal relationship. Hovering the mouse over an edge will show its odds ratio (e.g. the odds of a data point being in the "effect" pattern is 7.96 times higher if it is in the "cause" pattern).

3.5 Causal Layout

By setting the `show_causal` attribute to true in the `DCMPatternMiner` function (see section 4.1) the data context map will also compute and display causal relationships between patterns. Figure 5 shows how the data context map represents causal relationships. The source of a directed edge represents the cause while the destination represents the effect. Green edges indicate that a data point being in the "cause" pattern will have a higher probability of also being in the "effect" pattern. Red edges indicate the opposite relationship (i.e. data points in the "cause" pattern are less likely to be in the "effect" pattern). Edges can also be undirected. This indicates that a causal relationship exists, but the direction of causality cannot be determined.

3.6 Correlation Miner

In addition to identifying patterns where the distribution of some target attribute is unusually high / low, we also identify patterns where the correlation between two attributes is high. The correlation miner identifies sub-spaces where the correlation between one attribute and a target attribute is unusually high. This is then presented as a correlation table as seen in figure 6. Clicking on the "Scatter Plot" drop-down shows the correlation pattern control interface (see figure 7).

Search:

Variable 1	Variable 2	Pearson	Spearman	Num. Sub-Pop.
stddev	fut_return	-0.48 p=6.51e-23	-0.51 p=3.36e-26	6
Scatter Plot				
price/cashflow	fut_return	-0.09 p=0.07	0 p=0.94	3
Scatter Plot				
dividend_yield	fut_return	0.22 p=1.22e-05	0.29 p=1.68e-08	2
Scatter Plot				
stoch_osc	fut_return	0.08 p=0.13	0.08 p=0.12	2
Scatter Plot				
asset_turnover	fut_return	0.16 p=0	0.18 p=0	1
Scatter Plot				
current_ratio	fut_return	-0.09 p=0.1	-0.17 p=0	1
Scatter Plot				
payout_ratio	fut_return	0.15 p=0	0.33 p=4.47e-11	1
Scatter Plot				
price/book	fut_return	-0.17 p=0	-0.14 p=0.01	1
Scatter Plot				
price/earnings	fut_return	-0.18 p=0	0.05 p=0.31	1
Scatter Plot				
ros	fut_return	-0.09 p=0.12	0.08 p=0.18	1
Scatter Plot				

Figure 6: Correlation table showing Pearson and Spearman correlations between an attribute and the target attribute of interest.

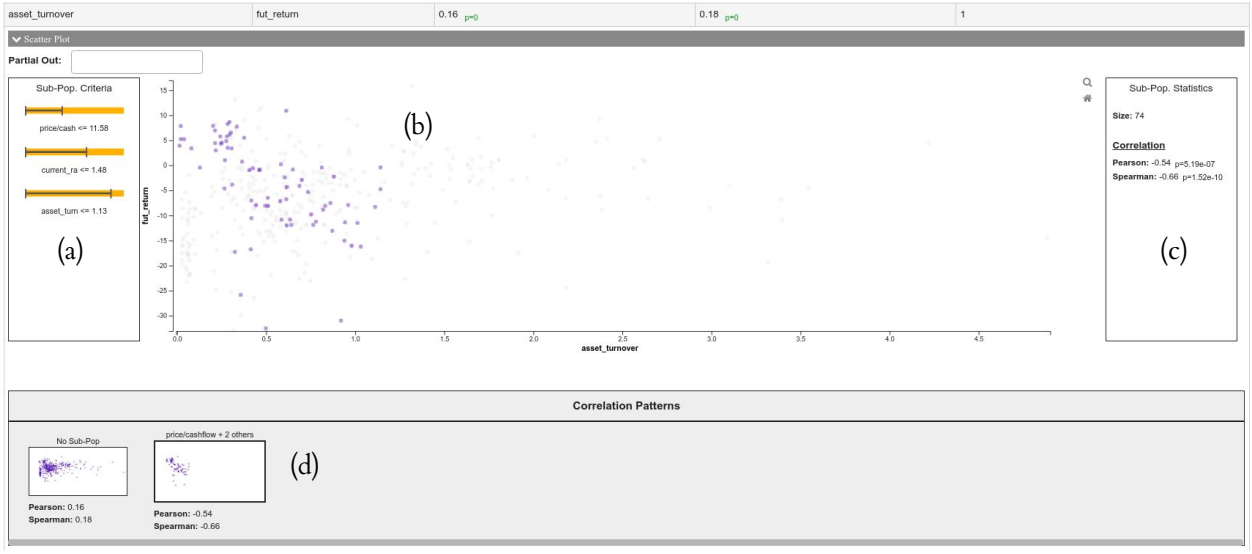


Figure 7: Correlation Mining control interface showing a pattern where asset_turnover is negatively correlated if fut_returns. (a) shows the criteria that defines the selected pattern. (b) shows a scatter-plot stratified on the criteria in (a). (c) shows statistics for this pattern (i.e. size, Pearson and Spearman correlations). (d) Shows a small multiples display illustrating the correlation patterns mined by the software. Clicking a pattern in this view will update (a), (b), and (c) accordingly.

4 API

There are several data context map specific functions that a user calls within the Jupyter notebook interface. These are explained in more detail in this section.

4.1 DataContextMap

```
data_context_map.pattern_miner.DataContextMap(df, dependent, temporal=None, pattern_json=None,
mine_type='numeric', minsup=0.01, es_thresh='auto', max_pattern=None, max_depth=None, min_stable=None,
ts_width=None, train_range=None, show_causal=False, causes_only=[], license_path=None, lib_path=None,
opt_bound=False, fdr='fast', holdout=3, causal_rule=None, verbose=False)
```

Description:

Performs pattern mining and prepares the visualization to be rendered.

Parameters:

- **df** (*pandas dataframe*): The data set to mine.
- **dependent** (*str*): The column of the dataframe that acts as the dependent variable. This is the attribute to predict.
- **temporal** (*str*): The attribute to be treated as a time variable.
- **pattern_json** (*list*): A list of pre-computed patterns of the form `{attribute:{'lb': v1, 'ub': v2}}`. When this parameter is set, pattern mining will not occur and instead this list of patterns will be visualized.
- **mine_type** (*str*): The data type of the dependent attribute. Valid mine types are 'numeric' and 'binary'.
- **minsup** (*float*): The minimum size threshold of a pattern as a percentage of the dataset size. Default is 1%.
- **es_thresh** (*float*): The minimum effect size for a pattern to be considered 'interesting'. The effect size is the common language effect size for numeric mine_type (default is 0.6) and the odds ratio for binary mine_types (default is 2).
- **max_pattern** (*int*): Maximum number of patterns to mine.
- **max_depth** (*int*): Maximum complexity of a pattern. e.g. max_depth=2 indicates that no pattern containing more than 2 attributes will be returned.
- **min_stable** (*int*): The number of consecutive time steps for which a pattern must be 'interesting' to be considered temporally consistent.
- **ts_width** (*int*): Width of the timestep. If ts_width = 1, then each unique value of the temporal attribute is a timestep. If ts_width > 1, then each time step is a range.
- **train_range** (*list*): List defining [start, end] of a training range along the temporal dimension under which to mine the patterns. If None, then patterns are mined from the full data set.
- **show_causal** (*bool*): If true, then causal analysis is performed and shown as directed green and red edges in the graph view.
- **causes_only** (*list*): List of attributes that have no cause (e.g. ['sector']).
- **license_path** (*str*): File path to the license.txt license key.
- **lib_path** (*str*): File path to the libdcm.so (libdcm.dll) shared library.
- **opt_bound** (*bool*): If true, the bounds of the patterns are optimized throughout each level of the mining process.

- **fdr** (*str*): Method for controlling the false discovery rate during pattern mining ('fast' or 'exhaustive'). The 'fast' method is a greedy strategy, whereas the 'exhaustive' looks at all significant patterns and removes false discoveries afterwards.
- **holdout** (*int*): Number of holdout sets to validate the patterns on.
- **causal_rule** (*str*): None or 'iptw'. If 'iptw' then the average treatment effect is determined using inverse probability of treatment weighting. Patterns without a significant treatment effect are pruned.
- **verbose** (*bool*): Boolean controlling verbose output.

Returns:

A `DCMPatternMiner` object.

4.2 render

`DCMPatternMiner.render()`

Description:

Draws the data context map list view to the screen.

4.3 get_selected_df

`DCMPatternMiner.get_selected_df()`

Description:

Returns the subset of the input dataframe that corresponds to a pattern. The dataframe returned corresponds to the last pattern where the export dataframe button was clicked (i.e. within the pattern detail view).

Returns:

The "exported" dataframe.

4.4 get_pattern_json

`DCMPatternMiner.get_pattern_json()`

Description:

Returns the mined patterns as list of python dictionaries of the form `{attribute:{'lb': v1, 'ub': v2}}`.

4.5 CorrelationTable

`data_context_map.correlation_miner.CorrelationTable(df, dependent, precompute=False, minsup=0.01, es_thresh=None, license_path=None, lib_path=None, fdr='fast', holdout=3, alpha=0.05, max_depth=None, max_pattern=None, opt_bound=False, verbose=False)`

Description:

Performs correlation mining and prepares the visualization to be rendered.

Parameters:

- **df** (*pandas dataframe*): The data set to mine.
- **dependent** (*str*): The column of the dataframe that acts as the dependent variable. This is the attribute to predict.
- **precompute** (*bool*): If True, all correlation patterns are identified prior to showing the correlation table.
- **minsup** (*float*): The minimum size threshold of a pattern as a percentage of the dataset size. Default is 0.01 (i.e. 1%).
- **es_thresh** (*float*): The minimum effect size for a pattern to be considered 'interesting'. The effect size is measured as the Spearman correlation.
- **license_path** (*str*): File path to the license.txt license key.
- **lib_path** (*str*): File path to the libdcm.so (libdcm.dll) shared library.
- **fdr** (*str*): Method for controlling the false discovery rate during pattern mining ('fast' or 'exhaustive'). The 'fast' method is a greedy strategy, whereas the 'exhaustive' looks at all significant patterns and removes false discoveries afterwards.
- **holdout** (*int*): Number of holdout sets to validate the patterns on.
- **alpha** (*float*): P-value significance level.
- **max_depth** (*int*): Maximum complexity of a pattern. e.g. max_depth=2 indicates that no pattern containing more than 2 attributes will be returned.
- **max_pattern** (*int*): Maximum number of patterns to mine.
- **opt_bound** (*bool*): If true, the bounds of the patterns are optimized throughout each level of the mining process.
- **verbose** (*bool*): Boolean controlling verbose output.

Returns:

A `CorrelationTable` object.

4.6 `render`

```
CorrelationTable.render()
```

Description:

Draws the correlation table to the screen.