
Siren User Guide

Release 3.1.0

Esther Galbrun and Pauli Miettinen

December, 2015

1	Introduction	1
2	Getting Started	3
2.1	Importing data	3
2.2	Mining redescrptions from scratch	3
2.3	Expanding a redescription	3
2.4	Filtering redescrptions	4
2.5	Exporting redescrptions	4
2.6	Saving as a package	4
3	Interface	5
3.1	Windows and Tabs	5
3.2	Menus	8
3.3	More	11
4	Formats	15
4.1	Data formats	15
4.2	Redescrptions formats	16
5	Preferences	19
5.1	Mining parameters	19
5.2	Interface parameters	19
5.3	Preferences file	20

INTRODUCTION



Redescription mining is a powerful data analysis tool that aims at finding alternative descriptions of the same entities.

For example, in biology, an important task is to identify the bioclimatic constraints that allow some species to survive, that is, to describe geographical regions in terms of both their bioclimatic conditions and the fauna that inhabit them.

Siren is a tool for interactive mining and visualization of redescriptions. It is based on the ReReMi mining algorithm.

GETTING STARTED

Siren allows you to interactively mine and visualize redescrptions from your data.

We outline here some high-level interactions offered by Siren.

2.1 Importing data

After you get *Siren* installed and running, it will open the *tools window*, with tabs for variables and redescrptions, all of them empty. Hence, the first thing to do is to import data or open an existing siren package to start working.

If you already have some siren package (i.e. with a `.siren` extension) you can open it via the interface menu *File* → *Open*.

Otherwise, or if you want to work on a new data set, you can *import data* to *Siren*, this will populate the `Entities` and `Variables` tabs.

2.2 Mining redescrptions from scratch

Once the two sets of variables are loaded and can be seen in the tabs you may want to let the tool mine redescrptions in an fully automated way, using currently enabled variables. The menu entry *Process* → *Mine* redescrptions let you do just that.

Alternatively, if both queries in a visualization are empty *Siren* simply mines redescrptions on your data when clicking on the `Expand` button.

Results will be appended to the redescrptions list in the `Expansions` tab.

Before running a mining task make sure you have adjusted the mining preferences...

2.3 Expanding a redescription

You can also press the expansion button to automatically find expansions of any redescription your are currently visualizing and editing. *Siren* will try to append literals to the current redescription, using the enabled variables.

Again, results will be appended to the redescrptions list in the `Expansions` tab.

Before running an expansion task make sure you have adjusted the mining preferences...

2.4 Filtering redescrptions

A list of redescription can be filtered automatically. That is, the algorithm will go through the redescrptions, from top to bottom in the current order, an check for each redescription, whether it is redundant given the previous ones. If a redescription is found redundant it will be disabled. This is done via the interface menu *Edit* → *Filter redundant*.

It is also possible to select a redescription and filter following redescrptions which are redundant to that particular redescription only, via the interface menu *Edit* → *Filter redundant* to current.

2.5 Exporting redescrptions

The redescrptions from the *Redescrptions* tab can be exported under *different formats*.

2.6 Saving as a package

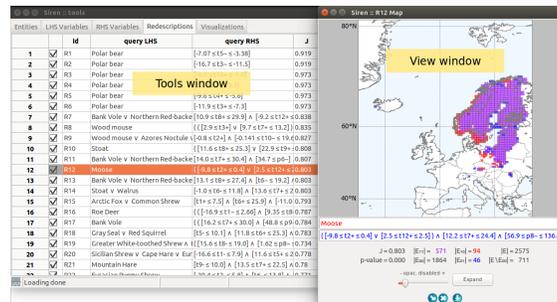
You can save your current project, i.e. the data, current redescription list and preferences as a siren package (i.e. with a `.siren` extension) via the interface menu *Edit* → *Save as...*

If you continue working on the current project you can save changes to the current siren package via the interface menu *Edit* → *Save*.

Existing siren packages can be opened via the interface menu *File* → *Open*.

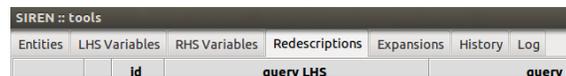
3.1 Windows and Tabs

Siren has two types of windows: **tools** and **view**, which are presented below.

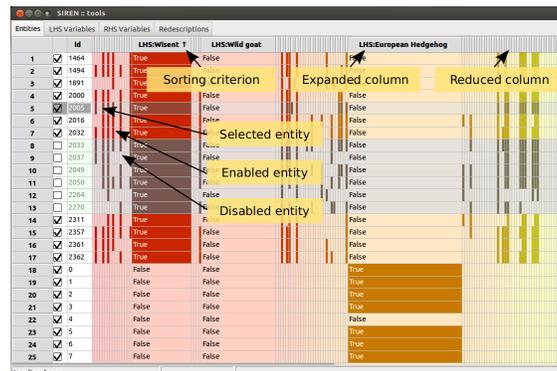


3.1.1 Tools window

The **Tools** window is unique, it is the main siren window. It contains several tabs.



- The **Entities** tab contains the list of entities.



Two tabs contain lists of variables.

- The **LHS Variables** tab,
- and the **RHS Variables** tab contain the list of left-hand side variables and right-hand side variables respectively.

id	name	type	density
134	White-tailed deer	Boolean	0.023
123	White-bellied bat	Boolean	0.0039
60	Water deer	Boolean	0.0047
133	Walrus	Boolean	0.0155
9	Ural Field Mouse	Boolean	0.0049
91	Tundra Vole	Boolean	0.1219
78	Tristram's Jird	Boolean	0.0000
96	Thomas's Pine Vole	Boolean	0.0124
95	Tatra pine vole	Boolean	0.0000
172	Tajga Shrew	Boolean	0.0171
4	Striped Field Mouse	Boolean	0.2012
106	Stoat	Boolean	0.5759
102	Steppe Polecat	Boolean	0.0000
104	Steppe Mouse	Boolean	0.0361
178	Speckled ground squirrel	Boolean	0.0019
185	Spanish Mole	Boolean	0.0097
23	Spanish Ibex	Boolean	0.0167
10	Southwestern water Vole	Boolean	0.1452
50	Southern White-breasted hedgehog	Boolean	0.1942
92	Southern Vole	Boolean	0.0536
166	Southern Birch Mouse	Boolean	0.0066
58	Small Asian Mongoose	Boolean	0.0023
28	Sika Deer	Boolean	0.0416
39	Sicilian Shrew	Boolean	0.0039
188	Siberian chipmunk	Boolean	0.0066
151	Siberian Flying Squirrel	Boolean	0.0416
48	Serotine bat	Boolean	0.3373

Three tabs contain lists of redescriptions.

- The **Redescriptions** tab is the main list of redescriptions. Redescriptions are imported and exported to and from that list.
- The **Expansions** tab lists redescriptions that have been generated by mining or extending queries.
- The **History** tab lists all edits made to queries, allowing to undo changes.

id	query LHS	query RHS	J	prob	IC-1	track
87	Wood mouse	$([0.5 \leq x \leq 1.5] \wedge [1.5 \leq y \leq 2.5] \wedge [1.5 \leq z \leq 2.5])$	0.0	1712	0.8114,18	
812	Roe Deer	$([0.35 \leq x \leq 0.75] \wedge [0.35 \leq y \leq 0.75] \wedge [0.35 \leq z \leq 0.75])$	0.0	1604	0.26,1,24,31	
817	Red Squirrel	$([0.5 \leq x \leq 1.5] \wedge [1.5 \leq y \leq 2.5] \wedge [1.5 \leq z \leq 2.5])$	0.0	1583	0.166,1,4,11	
816	European Pygmy Shrew	$([0.5 \leq x \leq 1.5] \wedge [1.5 \leq y \leq 2.5] \wedge [1.5 \leq z \leq 2.5])$	0.0	1459	0.17,0,1,1,5,6	
815	European Hare	$([0.7 \leq x \leq 1.3] \wedge [1.3 \leq y \leq 2.3] \wedge [1.3 \leq z \leq 2.3])$	0.0	1332	0.66,1,26,36	
89	Stoat	$([1.1 \leq x \leq 2.1] \wedge [1.1 \leq y \leq 2.1] \wedge [1.1 \leq z \leq 2.1])$	0.0	1300	0.106,13,11	
813	Common Shrew	$([1.1 \leq x \leq 2.1] \wedge [1.1 \leq y \leq 2.1] \wedge [1.1 \leq z \leq 2.1])$	0.0	1162	0.168,118,2	
821	Field Vole	$([1.5 \leq x \leq 2.5] \wedge [1.5 \leq y \leq 2.5] \wedge [1.5 \leq z \leq 2.5])$	0.0	1087	0.81,1,17,11	
830	European Pine Marten	$([1.5 \leq x \leq 2.5] \wedge [1.5 \leq y \leq 2.5] \wedge [1.5 \leq z \leq 2.5])$	0.0	998	0.6,10,19,2	
837	European Hedgehog	$([1.5 \leq x \leq 2.5] \wedge [1.5 \leq y \leq 2.5] \wedge [1.5 \leq z \leq 2.5])$	0.0	888	0.82,1,15,11	
822	European Water Shrew	$([1.5 \leq x \leq 2.5] \wedge [1.5 \leq y \leq 2.5] \wedge [1.5 \leq z \leq 2.5])$	0.0	875	0.127,118,1	
836	European Polecat	$([1.5 \leq x \leq 2.5] \wedge [1.5 \leq y \leq 2.5] \wedge [1.5 \leq z \leq 2.5])$	0.0	875	0.120,174,1	
819	Wild boar	$([4.1 \leq x \leq 10.1] \wedge [4.1 \leq y \leq 10.1] \wedge [4.1 \leq z \leq 10.1])$	0.0	1205	0.607,168,3	
852	European Otter	$([4.1 \leq x \leq 10.1] \wedge [4.1 \leq y \leq 10.1] \wedge [4.1 \leq z \leq 10.1])$	0.0	1188	0.11,1,12,12	
844	Common Pipit	$([0.7 \leq x \leq 1.3] \wedge [0.7 \leq y \leq 1.3] \wedge [0.7 \leq z \leq 1.3])$	0.0	1188	0.145,16,11	
831	European Water Vole	$([1.0 \leq x \leq 1.7] \wedge [1.0 \leq y \leq 1.7] \wedge [1.0 \leq z \leq 1.7])$	0.0	1177	0.75,10,14	
827	Beech Marten	$([1.0 \leq x \leq 1.7] \wedge [1.0 \leq y \leq 1.7] \wedge [1.0 \leq z \leq 1.7])$	0.0	1177	0.75,10,14	
818	European Mole	$([1.4 \leq x \leq 2.4] \wedge [1.4 \leq y \leq 2.4] \wedge [1.4 \leq z \leq 2.4])$	0.0	1162	0.168,118,2	
841	European Rabbit	$([0.7 \leq x \leq 1.3] \wedge [0.7 \leq y \leq 1.3] \wedge [0.7 \leq z \leq 1.3])$	0.0	1087	0.136,11,4	
833	Yellow-necked Mouse	$([1.8 \leq x \leq 2.8] \wedge [1.8 \leq y \leq 2.8] \wedge [1.8 \leq z \leq 2.8])$	0.0	998	0.6,10,19,2	
820	House mouse	$([1.5 \leq x \leq 2.5] \wedge [1.5 \leq y \leq 2.5] \wedge [1.5 \leq z \leq 2.5])$	0.0	992	0.10,1,12,1	
851	Red Deer	$([1.0 \leq x \leq 1.7] \wedge [1.0 \leq y \leq 1.7] \wedge [1.0 \leq z \leq 1.7])$	0.0	972	0.27,19,18	
832	Brown long-eared bat	$([1.3 \leq x \leq 2.3] \wedge [1.3 \leq y \leq 2.3] \wedge [1.3 \leq z \leq 2.3])$	0.0	963	0.147,118,2	
823	Common Vole	$([2.0 \leq x \leq 3.0] \wedge [2.0 \leq y \leq 3.0] \wedge [2.0 \leq z \leq 3.0])$	0.0	888	0.82,1,15,11	
824	Common Vole	$([2.0 \leq x \leq 3.0] \wedge [2.0 \leq y \leq 3.0] \wedge [2.0 \leq z \leq 3.0])$	0.0	888	0.82,1,15,11	
840	Daubenton's Bat A Eurasian Pygmy Shrew	$([1.4 \leq x \leq 2.4] \wedge [1.4 \leq y \leq 2.4] \wedge [1.4 \leq z \leq 2.4])$	0.0	875	0.120,174,1	

Finally,

- the **Log** tab contains logging output generated by the mining algorithm.

```

(0) Redescription: 0 = 46 terms: (0) 1 18-14-0-24-4 & 22-5-3+6.3 | 0.7588571428571 1.1022302463e-16 64 147 664 ..... 0:1;1;122:118
(0) LHS extension RHS extension score Accuracy Query pv Red pv Torble torled
(0) + <<<<< 8.41+34.718+118.38 +0.035822 0.7939394 0.000000 655 97
(0) + <<<<< 124+2.8667+2.86 +0.0116908 0.7705479 0.000000 11 1
(0) Extensions Batch:
(0) Redescription: 0 = 46 terms: (0) 1 22-5-3+6.3 | 24-2.8667+2.86 0.70423991727 1.1022302463e-16 47 239 681 ..... 0:1;1;122:124
(0) LHS extension RHS extension score Accuracy Query pv Red pv Torble torled
(0) + <<<<< 8.18+14-0+24.4 +0.0619887 0.7714386 0.000000 675 47
(0) + <<<<< 125-0.35+0.35 -0.0103413 0.7145812 0.000000 0.000000 10 0
(0) Extensions Batch:
(0) Redescription: 0 = 45 terms: (0) 1 18-14-0-24-4 & 22-5-3+6.3 & 43-34-714-118.38 | 0.79393939399 1.1022302463e-16 53 147 675 ..... 0:1;1;122:124;118
(0) LHS extension RHS extension score Accuracy Query pv Red pv Torble torled
(0) + <<<<< 8.41+34.718+118.38 +0.0358482 0.8071227 0.001059 0.000000 666 97
(0) + <<<<< 125-0.35+0.35 -0.0114286 0.7828571 0.000000 0.000000 10 0
(0) Extensions Batch:
(0) Redescription: 0 = 44 terms: (0) 1 18-14-0-24-4 & 22-5-3+6.3 | 24-2.8667+2.86 0.770547945205 1.1022302463e-16 53 148 675 ..... 0:1;1;122:118;124
(0) LHS extension RHS extension score Accuracy Query pv Red pv Torble torled
(0) + <<<<< 8.42+52.455+133.75 +0.0168630 0.7874109 0.000000 0.000000 663 114
(0) + <<<<< 125-0.35+0.35 -0.014155 0.7819635 0.000000 0.000000 10 0
(0) Extensions Batch:
(0) Redescription: 0 = 45 terms: (0) 1 22-5-3+6.3 | 24-2.8667+2.86 0.7588571428571 1.1022302463e-16 47 239 681 ..... 0:1;1;122:124;118
(0) LHS extension RHS extension score Accuracy Query pv Red pv Torble torled
(0) + <<<<< 8.30+8.7188+18.308 +0.0434654 0.758046 0.000000 0.000000 683 173
(0) + <<<<< 133+8.85+8.8667 +0.0023619 0.7169491 0.000000 0.000000 3 1
(0) Extensions Batch:
(0) Redescription: 0 = 46 terms: (0) 1 18-14-0-24-4 & 22-5-3+6.3 | 0.56
(0) LHS extension RHS extension score Accuracy Query pv Red pv Torble torled
(0) + <<<<< 8.43+52.335+105.4 +0.1954902 0.75812
(0) + <<<<< 110+8+0.8 0.0083601 0.5543672 0.000066 0.000000 3 22
(0) Extensions Batch:
(0) Redescription: 0 = 45 terms: (0) 1 18-14-0-24-4 & 22-5-3+6.3 & 43-32-333-105.4 | 0.788127438231 1.1022302463e-16 82 104 583 ..... 0:1;1;18;22:143
(0) LHS extension RHS extension score Accuracy Query pv Red pv Torble torled
(0) + <<<<< 8.40+26.706+01.938 +0.0382463 0.7963738 0.000000 0.000000 51 32
(0) + <<<<< 144+183.06+238.78 +0.0184668 0.7767742 0.000000 0.000000 19 6
    
```

Tabs can be shown or hidden via the interface menu *Windows* → *Tabs*.

The main means to manipulate variables or redescriptions, depending on which tab you are currently viewing, are available via the *Edit* menu and the contextual menu opened by right click.

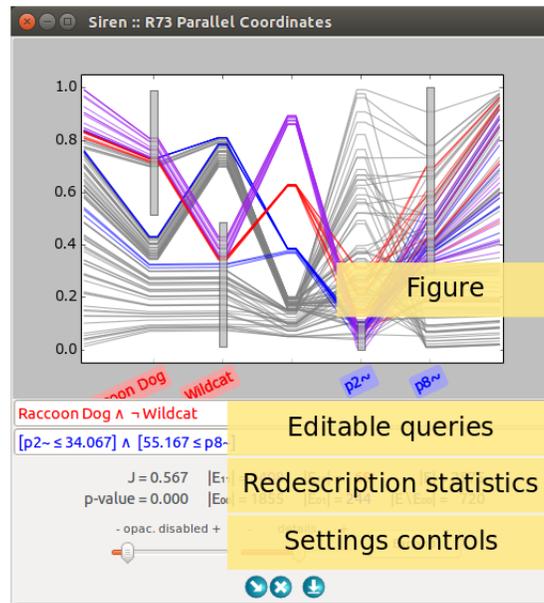
List items can be enabled/disabled by checking/unchecking the corresponding box in the left column. All items in a list can be enabled/disabled simultaneously. Lists can be sorted based on the value of the different fields displayed by clicking on the column header.

Redescriptions can be cut, copied, and pasted from and to different positions in the lists and from one list to the other and vice-versa. All disabled redescriptions can be deleted at once.

3.1.2 View window

A **view** window allows to visualize a redescription, to edit it and launch mining and expansion. Several views of different types of the same redescription can be opened simultaneously. They are linked together and to the original redescription in the list so that edits and selections made in one view are reflected in the list and other views.

A View window can be opened with a double click on a variable or redescription in a list from the Tools window.



Both queries can be edited using the text boxes. If the *syntax of a query* is incorrect, *Siren* will not be able to parse it and it will fall back on the previous correct query. Queries are parsed when ENTER is pressed, in order to avoid parsing error due to partial edits.

Redescriptions statistics are shown below the queries.

Expansion can be started by pressing the Expand button. The expansion will be delegated to a background process. It can be interrupted via the menu *Process* → *Stop expander XXX*, where running task are listed. Redescriptions generated during the expansion will be appended to the list of redescriptions in the Expansions tab.

Clicking on the entities, represented as lines or dots, highlights them across the different views for the redescription with labels showing the entities's ID to allow identifying them. It is also possible to enable highlighting and labeling the entities on hover (by activating *Hover entities* in the *Interface* preferences), and jumping to the entity in the entities list on click (by toggling *Edit* → *View details*).

Some actions are visualization-specific. A slider allows to set the opacity of disabled entities in all visualizations except the tree diagrams. For parallel coordinates plots, another slider allows to adjust the level of details by limiting the fraction of entities drawn.

In case of projections, setting controls allow to set parameters and the data can be projected anew by clicking the Reproject button. A polygon drawing tool allows to select groups of contiguous areas.

In tree diagrams, clicking on a leaf allows to add and remove the corresponding branch to the query. Then, Simplify LHS and Simplify RHS buttons allow to simplify the diagrams of the tree associated to the left-hand side and right-hand side queries, respectively.

In parallel coordinates plots, the range of literals can be adjusted interactively by dragging the corresponding grey boxes.

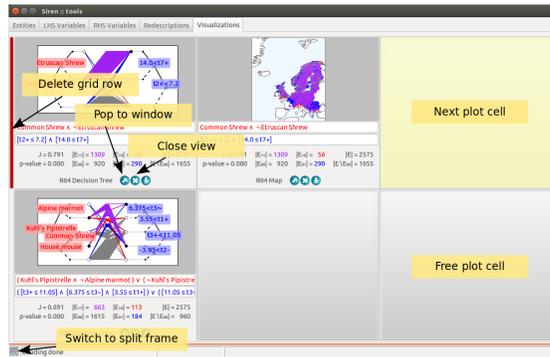
3.1.3 Display modes

Plotting the visualizations to separate windows is one possibility. In such a case, all controls and stats are available, as shown above.

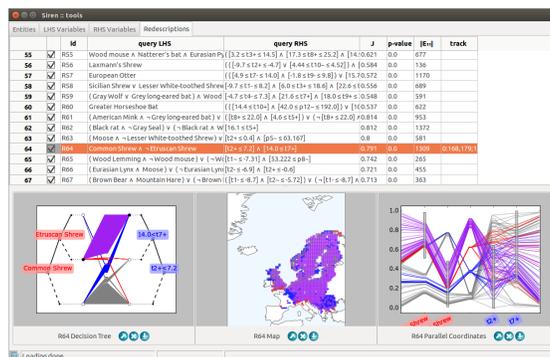
An alternative is to plot the visualization to a grid inside a dedicated tab of the main window. In that case, to accommodate more visualizations, controls are not shown, but the queries and stats are still visible.

An entry of the Edit menu allows to choose whether visualizations should be plotted by default into new separate windows or inside the visualization tab.

The plots can be popped out of the tab into separate windows, and vice-versa using the arrow buttons. The cross allows to close the view. Rows and columns can be added and dropped from the visualization grid by clicking on the green bars to the bottom and to the right of the grid, and the red bars to the top and to the left of the grid respectively. On deletion of a row or column of the grid, all views it contains are closed. The grid cell where the next visualization will be plotted is highlighted, a free cell can be chosen by simply clicking on it. If no free cell is available, the next plot will replace the oldest views in the grid, effectively deleting it.



Finally, clicking on the colored button on the bottom-left corner of the main window allows to split the main window horizontally into two frames, so as to show the grid of visualization beneath a list, for instance the list of redescrptions. To save further space the stats and queries are not shown in that case. Clicking anew on the colored button restores the main window to a single frame.



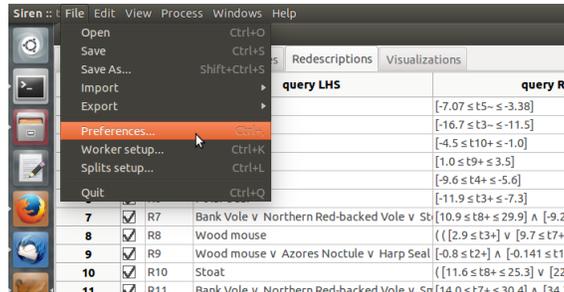
3.2 Menus

The main menu of Siren is at the top of the Tools window.

Here is a summary of functionalities available through the menu.

3.2.1 File

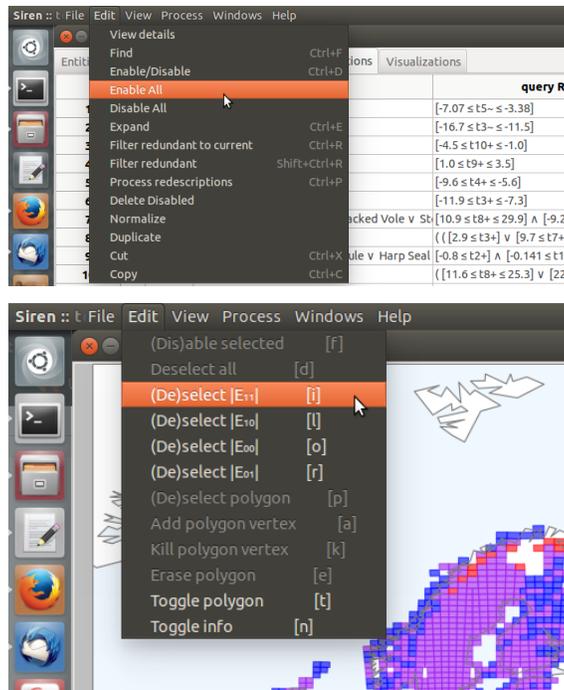
The `File` submenu provides import, export, opening and saving functionalities and setting the preferences.



3.2.2 Edit

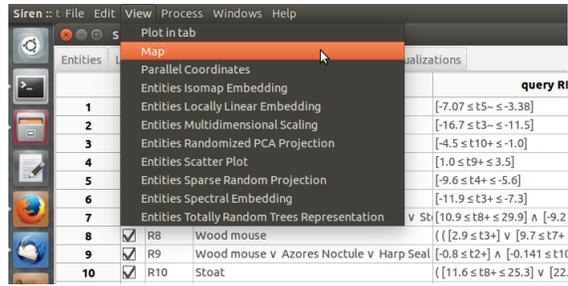
The content of the `Edit` submenu depends on the tab or view currently active.

If the tab contains redescriptions, it will also allow to filter the redescription and to copy, cut and paste them.



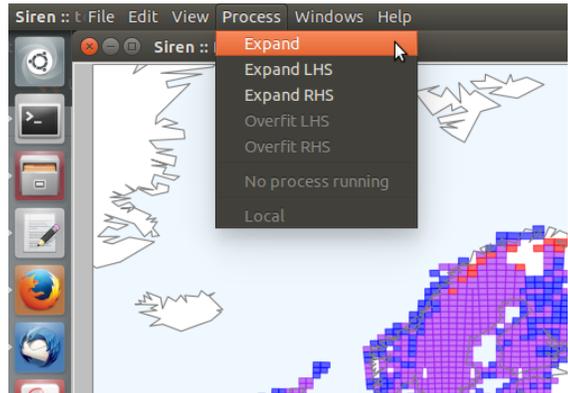
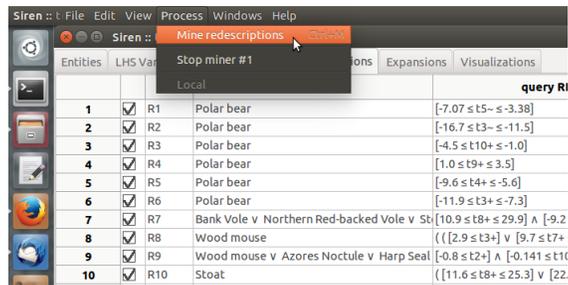
3.2.3 Views

If the tab contains redescriptions or variables the `View` menu will allow to open a *view window* to visualize the selected item. Possible visualizations for an item might differ. For instance, if a redescription cannot be represented as a tree, the tree diagram entry of the menu will be disabled, the `normalize` function should be applied first.



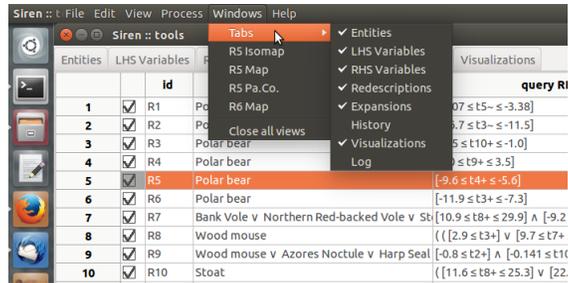
3.2.4 Process

The Process menu allows to start mining redescrptions and contains a list of running task and allows to interrupt any of them.



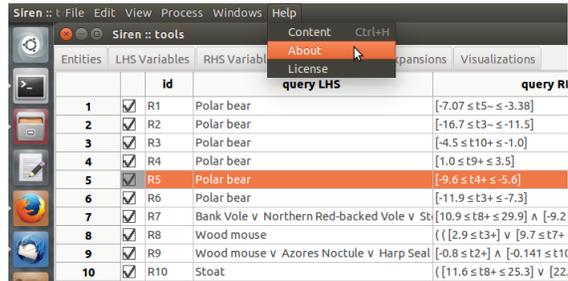
3.2.5 Windows

The Windows menu contains a list of tabs and allows to show or hide any of them. It also contains a list of Views currently opened sorted by redescription, allowing to access any of them and close all at once.



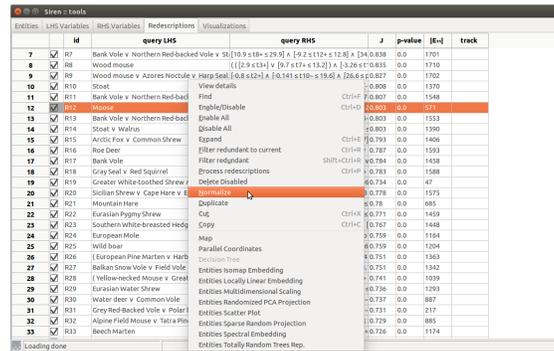
3.2.6 Help

The `Help` menu provides access to this help and to more information about *Siren* and licensing.



3.2.7 Contextual menu

Many `Edit` and `Views` functionalities can also be accessed via the contextual menu upon right-click on a redescription.



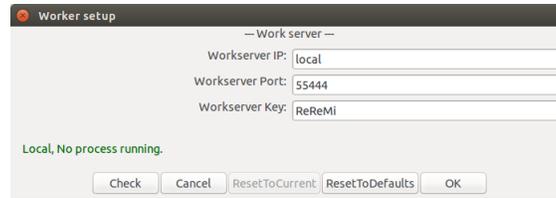
3.3 More

Siren provide means to send computations to an external server, thereby helping to maintain the responsiveness of the interface

In addition, to help study how well the redescrptions mined using a particular set of parameters generalize to unseen entities, Siren allows to easily partition the data, run the algorithm on a selected subset of the data and compare accuracy and support of the results.

3.3.1 Mining server

To pass mining and projection computation on to an external server, the settings for communicating with that server can be set in the `Worker setup...` in the `File` menu. This allows to set the IP address (set to local to disable the client-server mode), port and authentication key, and to test the connection and check whether some processes are already being run on the server.



Of course, the mining server should be setup with the relevant settings using the provided `server_siren.py` script. Running the following, for instance, initializes a server on the feedback-loop listening on port 55444 with authentication key `atk` and having 4 workers (i.e. handling at most 4 different delegated tasks simultaneously).

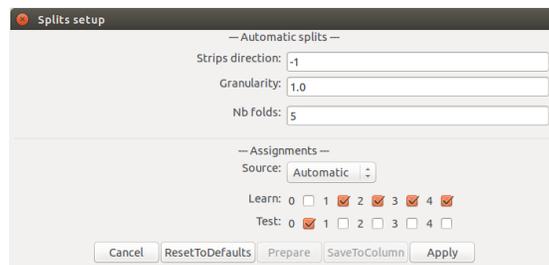
```
python server_siren.py 127.0.0.1 --portnum=55444 --authkey=atk --max_k=4
```

3.3.2 k-fold mining

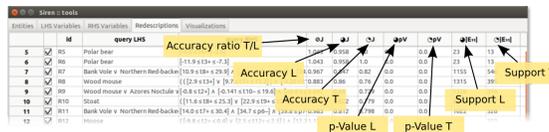
Finally, *Siren* also allows to partition the data to study how redescription characteristics vary on different data subsets. The settings for doing this can be accessed through the `Splits setup...` in the `File` menu.

The upper half of the dialogue box allows to choose parameters to generate a data partition automatically, by selecting a column or coordinate to whose values will direct the partition. A value greater than zero identifies a data column by its position, while a value lower than zero identifies a coordinate, a value of zero means that the data will be partitioned randomly. That is, 1 corresponds to the first column on the left-hand side, while 12 corresponds to second column on the right-hand side assuming the left-hand side contains 10 columns, and -1 corresponds to the first coordinates dimension, i.e. the longitudes. The granularity allows to tune the width of the bands, while the number of folds defines the number of partitions returned.

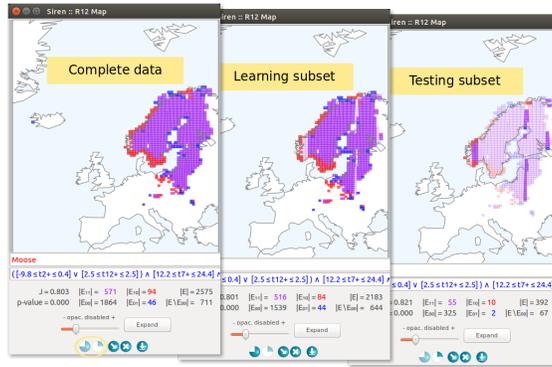
The lower half of the dialogue box allows to choose the source, either deactivating the splits, using the automatic settings described above to generate partitions or extracting it from a data column (a categorical variable, where each category will generate a partition). Once this is set, the partitions can be generated. Next, the resulting partitions can be assigned to the learn (L) or test (T) folds. A partition can be stored in the data by encoding it as an additional column, so that it can be loaded again later on.



After the splits setup as been applied, new columns will appear in the redescription lists, showing the statistics (accuracy, p-Values, support) on the learn and test folds.



In views, a pair of buttons will appear allowing to focus on the complete data or on either split.



4.1 Data formats

For redescription mining, one considers entities described by variables divided into two sets, hereafter arbitrarily called left-hand side and right-hand side. This can be seen as a pair of data matrices, where entities are identified with rows and variables with columns. Both sets of variables describe the same entities, hence, the matrices have the same number of rows.

In *Siren*, data include:

- **Variables:** The variables describing the entities are divided in two sets. They can be of three types:
 1. Boolean,
 2. categorical,
 3. or real-valued.

Obviously, this is required.

- **Entities names:** Optional additional information, providing names for the entities.
- **Variable names:** Optional additional information, providing names for the variables.
- **Coordinates:** Optional location information, i.e. geographic coordinates of the entities. This makes the data geospatial.

Data can be imported to *Siren* via the interface menu *File* → *Import* → *Import Data*. Below, we present the data formats supported by *Siren*.

Data can be imported into *Siren* as CSV files. The program expects a pair of files, one for either side in [character-separated values](#), as can be imported and exported to and from spreadsheet programs, for instance.

There are two main formats,

- **Full:** standard table format, or
- **Sparse:** compact format for dataset with few non-zeros entries.

The two data files need not be in the same format.

If entities names and/or coordinates are provided, they will be used to match entities across the two sides. Otherwise, rows will be match in order and an error will occur if the two side do not contain the same number of rows.

4.1.1 Full format

The data is stored as a table with one column for each variable and one row each entity. The first row can contain the names of the variables. The entities names can be included as columns named *id*. Similarly the coordinates can be

included as a pair of columns named *longitude* and *latitude*, respectively.

4.1.2 Sparse format

This format allows to store data that contains few non-zero entries more compactly, as in the Matlab sparse format (or like the edge list of a bipartite graph).

Each line contains an entry of the data as a triple (entity, variable, value). This way, the data is stored as in three columns and as many rows as there are entries. In this case the first line of the data file must contain *id*, *cid* and *value*, indicating the three columns containing the entities, variables and corresponding value, respectively. Coordinates can be provided in a similar way under the variable names *longitude* and *latitude*.

Variable names can be provided inline, that is, simply by using the name of the variable for each entry involving it. Alternatively, variable names can be specified separately with a special “-1” entity. Similarly, entity names can be provided inline or separately with a special “-1” variable. For example, the following five lines

```
id; cid; value
Espoo; population; 260981
Helsinki; population; 614074
Tampere; population; 220609
Turku; population; 182281
```

are equivalent to the following:

```
id; cid; value
20; -1; Espoo
7; -1; Tampere
2; -1; Turku
13; -1; Helsinki
-1; 3; population
2; 3; 182281
7; 3; 220609
13; 3; 614074
20; 3; 260981
```

Finally, in case of fully Boolean data without coordinates, the value can be left out. Each pair of (entity, variable) appearing is considered as True, the rest as False.

For both full and sparse formats a mention of type can be appended to the first row, in such case all variables will be parsed to the given type. For instance, in the example above the first line would be turned to `id; cid; value; type=N` to ensure that all variables, including population are interpreted as numerical (N) variables. Respectively B and C can be used to ensure that all variables are Boolean and categorical, respectively.

This can be useful when handling a dataset of numerical variables where some contains only two distinct values and might otherwise be interpreted as Boolean variables. It can also be a handy way to turn a dataset to fully Boolean based on zero/non-zero values. However, be warned that this can cause some troubles...

4.2 Redescriptions formats

The product of redescription mining is a list of redescriptions. A redescription consists of a pair of queries over the variables describing the entities, one query for each set. The two sets of variables are arbitrarily called left-hand side and right-hand side, and so are the corresponding queries.

4.2.1 Supports

The support of a query is the set of entities for which the query holds. Any given redescription partitions the entities into four sets:

- E_{10} is the set of rows for which only the left hand side query holds,
- E_{01} is the set of rows for which only the right hand side query holds,
- E_{11} is the set of rows for which both queries hold,
- and E_{00} is the set of rows for which neither of the queries hold.

Redescriptions can be imported to *Siren* via the interface menu *File* → *Import* → *Import Redescriptions*. More importantly, they can be exported via the interface menu *File* → *Export Redescriptions*. Below, we present the redescription formats supported by *Siren*.

4.2.2 Queries

A query is formed by combining literal using Boolean operators.

While *ReReMi* only generate linearly parsable query (see references for more details), *Siren* can actually evaluates arbitrary queries, as long as they are well formed following the informal grammar below. In particular, parenthesis should be used to separated conjunctive blocks and disjunctive block, alternating between operators. For example, while the later cannot be generated by *ReReMi*, $(a \wedge b) \vee \neg c$ and $(a \wedge b) \vee (c \wedge d)$ are both supported. $(a \wedge b) \wedge (c \wedge d)$ is not, because of incorrect alternance of operators between parenthesis blocks. It should simply be written as $a \wedge b \wedge c \wedge d$.

We consider three types of literals, defined over a Boolean, categorical or numerical variable respectively.

Below is an unformal grammar of *Siren*'s query language. The actual grammar can be found in the `redquery.ebnf` file in the `siren.reremi` source repertory.

```

query = disjunction | conjunction | literal ;
conjunction = conj_item { ( "&" | " ^ " ) conj_item }+ ;
disjunction = disj_item { ( " | " | " v " ) disj_item }+ ;
conj_item = literal \ ( "(" disjunction ")" ) ;
disj_item = literal \ ( "(" conjunction ")" ) ;
literal = categorical_literal | realvalued_literal | boolean_literal ;
categorical_literal = ( "[" )? variable_name ( " = " | " ≠ " | " ∈ " | " ∉ " ) category ( "]" )? ;
realvalued_literal = [ neg ] ( "[" )? [ variable_value lth ] variable_name lth variable_value ( "]" )? ;
realvalued_literal = [ neg ] ( "[" )? variable_value lth variable_name ( "]" )? ;
boolean_literal = [ neg ] ( "[" )? variable_name ( "]" )? ;
variable_name = STRING \ ?/vd+/? ;
category = STRING \ ?/d+/? ;
variable_value = ?/[+-]?d+(.[.])?d*([Ee][+-]d+)?/? ;
lth = "<" | " ≤ " ;
neg = "!" | "¬" ;

```

Naturally, the type of literal and the type of variable should match, i.e., $[4.0 \leq Va \leq 8.32]$ is a valid numerical literal only if the corresponding variable Va is a numerical variable. Furthermore, the upper bound of a numerical variable should always be greater or equal to the lower bound and either of them should be specified.

4.2.3 Redescription statistics

The statistics of a redescription include:

- accuracy, as measured by Jaccard coefficient $|E_{11}|/(|E_{10}| + |E_{11}| + |E_{01}|)$,
- p-value,
- cardinality of the *support sets* E_{10} , E_{01} , E_{11} , E_{00} (sometimes also referred to as alpha, beta, gamma and delta, respectively).

4.2.4 Exporting Redescriptions

Redescriptions from the `Redescriptions` tab can be exported to a file, one redescription per line, with both queries and basic statistics tab separated. Three of formatting options are available, determined by the provided filename:

- **named**: Uses the names of the variables instead of variable ids in the queries. Activated if the filename matches the pattern `*[a-zA-Z]named[a-zA-Z]*`.
- **all** By default disabled redescriptions will not be printed when exporting redescriptions. If the filename matches the pattern `*[a-zA-Z]all[a-zA-Z]*`, disabled redescriptions will also be printed.
- **tex** Rather than tab separated format, if the filename as `.tex` extension, a tex file is produced that can be compiled to obtain a table of the redescriptions. (Cannot be imported back)

Inside a siren package, the redescriptions are stored in tab separated format together with disabled status.

4.2.5 Importing Redescriptions

Tab separated formats can be imported into *Siren*, *TeX* cannot.

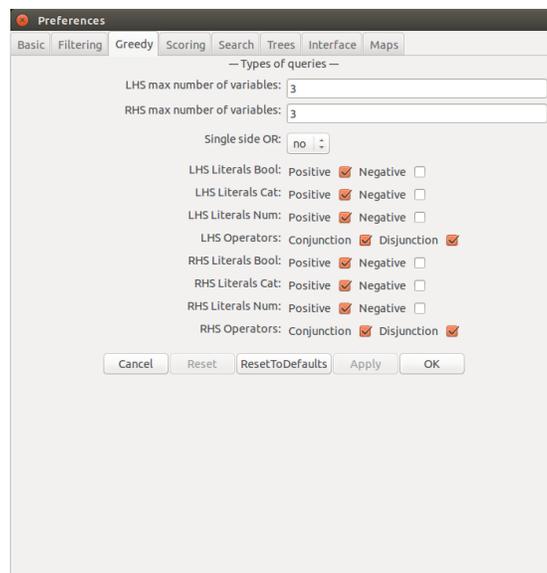
PREFERENCES

Preference parameters can be set via the interface menu Edit → Preferences.

An XML preferences file stores all non-default parameters.

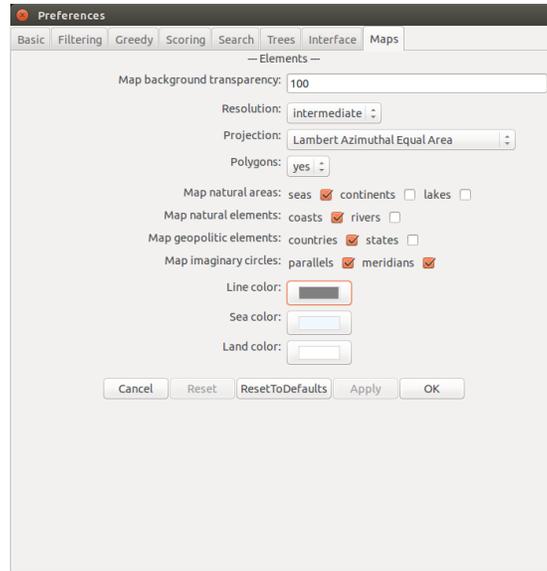
5.1 Mining parameters

Mining parameters are specified here and can be set through the interface.

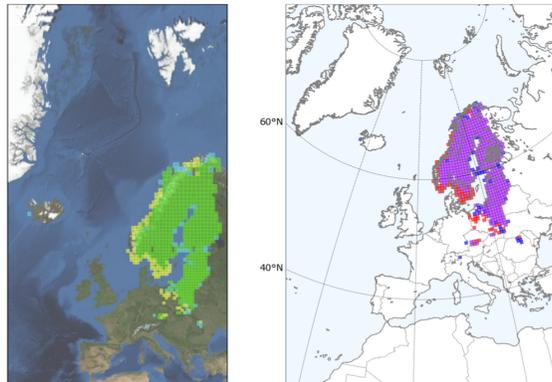


5.2 Interface parameters

Interface parameters are specified here and can be set through the interface.



For instance, this allows to modify the appearance of the maps.



5.3 Preferences file

The outline of a preferences file for *Siren* and *ReReMi* is as follows:

```
<root>
  <!-- start parameter -->
  <parameter>
    <!-- parameter name -->
    <name>max_red</name>
    <!-- parameter value -->
    <value>100</value>
  </parameter>
  <!-- end parameter -->
  <!-- start another parameter -->
  <parameter>
    <name>neg_query</name>
    <!-- multiple choices parameters might have several values -->
    <value>Positive</value>
    <value>Negative</value>
  </parameter>
```

```
<!-- end another parameter -->  
<!-- etc. ---->  
</root>
```