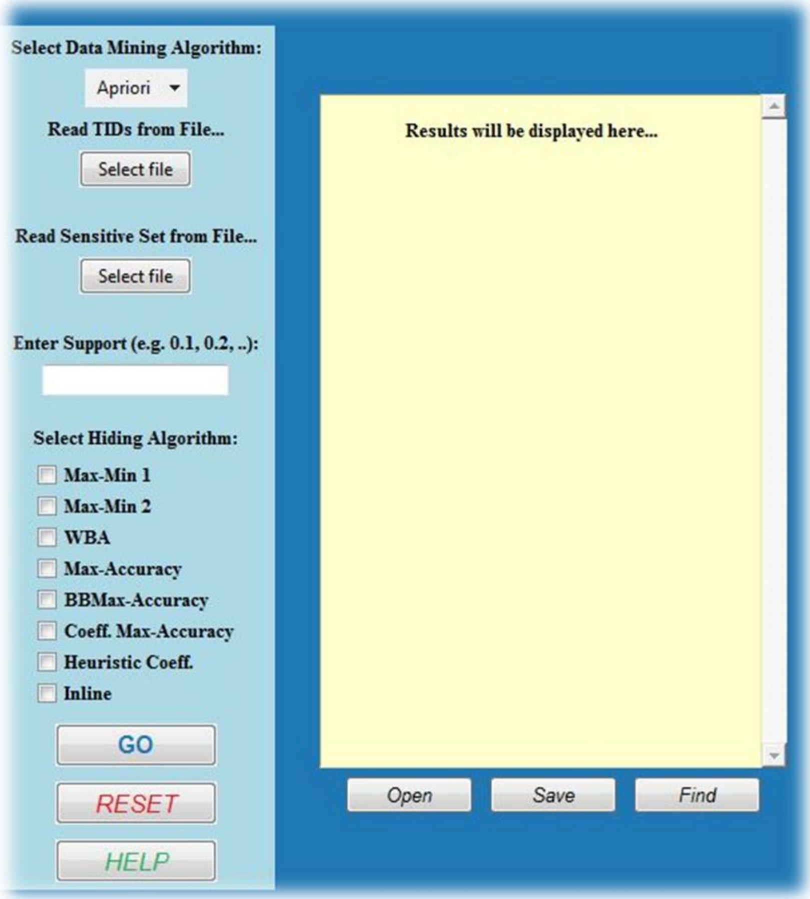
Frequent Itemset Hiding Toolbox Manual

Version 1.0

How to Use

The toolbox is designed to be as simple and convenient as possible. The next picture shows the main options given by the GUI of the toolbox.



One can simply apply a hiding algorithm by following the next easy steps:

1) Select Data Mining algorithm (currently only one available – already selected).
2) Select dataset file.
3) Select file of sensitive itemsets.
4) Give support threshold.
5) Select hiding algorithm(s).
6) Press the “GO” button.
7) Save results in text or image format. For the text format a name for the file with the results is requested. Images are saved with the name “image_YYYYMMDDHHMMSS.png”, where “YYYYMMDDHHMMSS” describes when the save button was pressed.

Supported File Formats

a) Dataset File

Datasets can be either space-separated or comma-separated files. The delimiter is declared by the extension of the file. Therefore, a space-separated file must have the extension ".ssv", while a comma-separated file must have the extension ".csv". If the input file has a different extension, then by default the space delimiter is used. Each line declares a transaction and the items of the transaction must be separated by the appropriate delimiter, as declared by the extension of the file.

Examples:

Tid	Transaction
1	a b c d
2	a c d
3	b c
4	b c d
5	a c
6	a c d

Space-separated

Tid	Transaction
1	a,b,c,d
2	a,c,d
3	b,c
4	b,c,d
5	a,c
6	a,c,d

Comma-separated

b) Sensitive Itemsets' File

The file containing the sensitive itemsets must follow the same format guidelines as described in paragraph (a). However, the dataset and the file of sensitive itemsets may have different delimiters.

If a single frequent itemset hiding algorithm is selected, then each line declares a sensitive itemset, with its items separated by the delimiter.

If multiple frequent itemset hiding algorithms are selected, then a line declares an execution of the algorithms. Multiple sensitive itemsets must be declared in a single line and be separated with ";". Multiple lines declare multiple executions*****. The table below shows all possible valid scenarios for the space-separated file format.

Scenario	1 FIH algorithm 1 Sens. Itemset	1 FIH algorithm > 1 Sens. Itemset	> 1 FIH algorithm > 1 Sens. Itemset 1 Execution	> 1 FIH algorithm > 1 Sens. Itemset > 1 Executions *
Example	a b c d	a b b c c d	a b;b c;c d	a b;b c;c d a b;b d;a d

* The last scenario declares 2 executions; the first will run the selected algorithms using sensitive itemsets *ab, bc* and *cd*, while the second will run the selected algorithms using sensitive itemsets *ab, bd* and *ad*. Note that in this case, plots will show the average of the metrics.

c) Support Threshold

In the support threshold field the input can be either a single float number or a sequence of 3 float numbers separated with ":", following the format "START:END:STEP". For instance, the sequence "0.2:0.8:0.2" indicates that the selected algorithms should be executed for support thresholds from 0.2 to 0.8 with step 0.2, i.e., for support threshold equal to 0.2, 0.4, 0.6 and 0.8.

If a single float number is given, then this is the support threshold that is going to be used for all executions declared by the sensitive itemsets' file. On the other hand, a sequence declares how many support thresholds are going to be used on each execution declared by the sensitive itemsets' file. For instance, the sequence "0.2:0.8:0.2" combined with the last scenario* of sensitive itemsets, presented in paragraph (b), declares 4 x 2 = 8 executions:

- 4 executions with support 0.2, 0.4, 0.6 and 0.8 for sensitive itemsets *ab*, *bc* and *cd*
- 4 executions with support 0.2, 0.4, 0.6 and 0.8 for sensitive itemsets *ab*, *bd* and *ad*

Note that in this case, plots will show the average of the metrics for each support threshold.

Extending the FIH Toolbox

Firstly, create a folder named “Extensions” in the same directory of the toolbox executable (FIH_Toolbox.exe). User-implemented modules should be placed into this folder and must be compatible with the Toolbox. Therefore, a specific code template must be used. We are going to give a demonstration example on how to create a compatible module.

Assume that we want to import a module named *“myNewModule”*. This module should contain the definition of a function named *“myNewModule_main”* that takes 5 parameters in the following strict order:

1. the filename of the frequent itemsets mined by the loaded dataset, under the specified threshold,
2. the name of the file with the sensitive itemsets,
3. the name of the file of the original dataset,
4. the threshold used,
5. and the name of the module itself.

The function should return the number of changes made in the original dataset, and the execution time of the algorithm. The sanitized version of the dataset must be written in a text file named “myNewModule_results.txt”. We give an abstract template of how the code file should look.

myNewModule.py

```
from time import clock

# Declare all function parameters, even if you are not going to use them all!
def myNewModule_main(freq_fname, data_fname, sens_fname, thr, name):

    # Load all necessary files/data here.
    change_raw_data = 0

    # Start measuring execution time.
    start_time = clock()

    # Implement hiding algorithm here + keep track of the number of changes

    exec_time = clock()-start_time

    # Produce output file with the sanitized version
    with open(name+'_results.txt', 'w') as out_file:
        # write sanitized version in space separated format
        # 1 transaction per line

    return(change_raw_data, exec_time)
```