GenMiner user guide

Ricardo Martinez[1], Nicolas Pasquier[1] and Claude Pasquier[2]
[1]I3S Laboratory, UNSA/CNRS UMR-6070, 2000 route des Lucioles, 06903 Valbonne, France
[2]IDBC, UNSA/CNRS UMR-6543, Parc Valrose, 06108 Nice, France

1. Installation

1.1 prerequisite

To install and run GenMiner, two mandatory pieces of software are needed: the Java Runtime Environment (JRE) and the The R Statistical Computing software.

JAVA runtime environment
If you don't have a java2 runtime installed on you machine, you can download and install the latest version (v.1.6) from this page: http://www.java.com/en/download/. All distributions include a graphical installer which covers all the installation procedure.
Note: to check whether java is installed on you machine, you can visit this page http://www.java.com/en/download/help/testvm.xml

R Statistical Computing software
R can be found at http://www.r-project.org/. The site provides precompiled versions of R for Windows, Mac OS X and several Linux platforms. The installation can be made from source code as well.
GenMiner uses three external R libraries that must be downloaded and installed before running the program. To install these libraries, launch the R Graphical interface; select the ***packages*** menu on top of the main windows, then ***install packages***. On the new windows that appears, choose a mirror for the download, then select the three packages **outliers**, **tseries** and **nortest**. You are now ready to execute GenMiner (the R program can be closed. R will be automatically called by GenMiner).

1.2 Installation of GenMiner

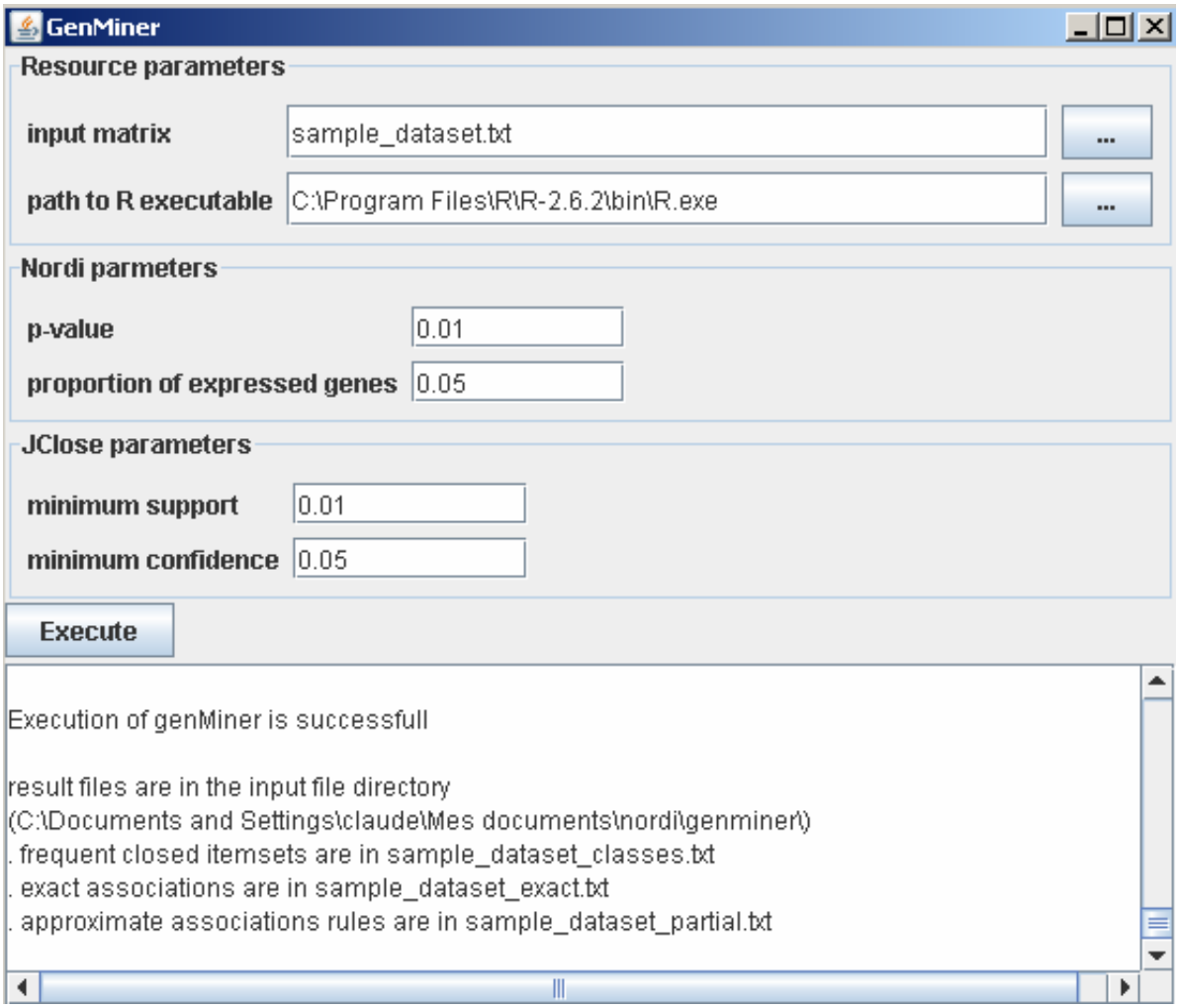
The installation procedure is very simple: just uncompress the file **genminer.zip** into the location of your choice. This will create a folder called **genminer** with the following content:

- **genminer.jar**: the GenMiner executable program,
- **genminer.bat**: the windows launch file,
- **genminer**: the linux launch file,
- **userguide.pdf**: the GenMiner manual (this file),
- **sample_dataset.txt**: a very simple dataset that can be used to test the program
- **sample_dataset_classes.txt**: the frequent closed itemsets obtained by applying GenMiner on the sample_dataset data,
- **sample_dataset_exact.txt**: the exact associations rules obtained by applying GenMiner on the sample_dataset data
- **sample_dataset_partial.txt**: the approximate associations rules obtained by applying GenMiner on the sample data.

2. Execution of GenMiner

To launch GenMiner, just execute the file **genminer.bat** on Windows or **genminer** on Linux or Mac. GenMiner can also be conveniently launched by double-clicking on the ***genminer.jar*** archive or by typing the command **java -jar genminer.jar** from the command line.

2.1 The GenMiner interface



GenMiner interface

Several parameters needed by GenMiner can be specified on this window:

- Input parameters
  - The **input matrix** represents the location of the file to be processed. By default, the file is *sample_dataset.txt* from the genminer folder. Another file can be selected by clicking on the button to the right.
  - The **path to R executable** contains the location of the R program. By default, the field is initialized with the default location of the 2.6.2 version of R on windows. Another location can be specified by clicking the button on the right.
- NorDi parameters
  - The **pValue** is used by NorDi for determining the gene expression matrix column outliers using the Grubbs outliers test.
  - The **proportion of expressed genes** represents the quantiles for a standardized normal distribution corresponding to certain α for determining the upper and lower

discretization thresholds in each column matrix. For example, with a value of 0.05, 5% of genes are identified as expressed and the remaining 95% are considered unexpressed.

- JClose parameters
  - **Minimum support** is used to filter useful rules. It represents the minimum proportion of objects containing all items of the rule.
  - **Minimum confidence** is used to extract only meaningful rules. It represents the minimum Proportion of objects containing the consequent among those containing the antecedent.

Detailed explanation of NorDi and JClose parameters can be found in the paper ***GenMiner: Non-Redundant Association Rules Mining from Genomic Data*** that is submitted to Bioinformatics.

2.1 Results

The execution of GenMiner is performed by clicking the execute button. The results are presented in three different files:

- a file with suffix **_classes.txt** that contains frequent closed itemsets,
- a file with suffix **_exact.txt** that contains exact associations rules,
- a file with suffix **_partial.txt** that contains approximate associations rules.

The format of these outputs is described in section 4.

3. Input Data Format

The data has to be presented in a matrix. The following matrix is used to describe the data in *sample_dataset.txt*:

Gene name	P1	P2	P2	V1	V2
Gene1	good	A		2.65	15.0
Gene2	average	A	B	1.02	-1.85
Gene3	good	B	C	3.00	-0.76
Gene4	unknown	C		0.23	1.42
Gene5	poor	C		-25.54	1.00

The first row represents the header. The other rows are the description of objects. For each row, the first column identifies the object and the following columns represent the properties of the object.
Cells are separated by tabs. Missing values should be left empty or identified with the character **'?'**. The columns that must be discretized (representing expression levels) are automatically identified depending on the content of the first value of each column. The rule is simple: the column is discretized if the first cell contains a decimal number. If the first cell is an integer and you want the column to be discretized, write the integer in decimal form (i.e.: replace **1** with **1.0**).

The dataset above contains the description of 5 genes with 4 properties:

- P1 is a descriptive property that can have four different values : '**good**', '**average**', '**poor**' or '**unknown**'
- P2 is a multivalued descriptive property that can be assigned with one or two values from the following list ('**A**', '**B**', '**C**')
- V1 and V2 are numerical values

4. Output Format

4.1 frequent closed itemsets

Each line of this file represents a frequent closed itemset that is identified with its generator and its closure. For example, the file *sample_dataset_classes.txt*, obtained by applying GenMiner on the sample data contains the following lines:

```
[V1=under] [P2=C, P1=poor, V1=under] 1
[P1=poor] [P2=C, P1=poor, V1=under] 1
[P1=unknown] [P2=C, P1=unknown] 1
[P2=C] [P2=C] 3
[P2=B] [P2=B] 2
[P2=B, P2=C] [P1=good, P2=B, P2=C] 1
[P1=average] [P2=A, P1=average, P2=B] 1
[V2=over] [P1=good, P2=A, V2=over] 1
[P2=A] [P2=A] 2
[P2=A, P2=B] [P2=A, P1=average, P2=B] 1
[P1=good] [P1=good] 2
[P1=good, P2=C] [P1=good, P2=B, P2=C] 1
[P1=good, P2=B] [P1=good, P2=B, P2=C] 1
[P1=good, P2=A] [P1=good, P2=A, V2=over] 1
```

The first line displays a frequent closed itemset that has ***V1=under*** as generator and ***P2=C, P1=poor, V1=under*** as closure.

4.2 exact associations rules

Each line of this file represents an exact rule with its antecedent, its consequent. For example, the file *sample_dataset_exact.txt*, obtained by applying GenMiner on the sample data contains the following lines:

```
[V1=under] => [P2=C, P1=poor] supp=1 conf=1
[P1=poor] => [P2=C, V1=under] supp=1 conf=1
[P1=unknown] => [P2=C] supp=1 conf=1
[P2=B, P2=C] => [P1=good] supp=1 conf=1
[P1=average] => [P2=A, P2=B] supp=1 conf=1
[V2=over] => [P1=good, P2=A] supp=1 conf=1
[P2=A, P2=B] => [P1=average] supp=1 conf=1
[P1=good, P2=C] => [P2=B] supp=1 conf=1
[P1=good, P2=B] => [P2=C] supp=1 conf=1
[P1=good, P2=A] => [V2=over] supp=1 conf=1
```

The first line displays a rule stating that all items annotated with ***under*** for the attribute ***V1*** are also annotated ***C*** for the attribute ***P2*** and ***poor*** for the attribute ***P1***. Support and confidence are both equals to 1 as this file only contains exact rules

4.3 approximate associations rules

Each line of this file represents a rule with its antecedent, its consequent. For example, the file *sample_dataset_partial.txt*, obtained by applying GenMiner on the sample data contains the following lines:

```
[P2=C] -> [P1=poor, V1=under] supp=1 conf=0,33
[P2=C] -> [P1=unknown] supp=1 conf=0,33
[P2=C] -> [P1=good, P2=B] supp=1 conf=0,33
[P2=B] -> [P1=good, P2=C] supp=1 conf=0,50
[P2=B] -> [P2=A, P1=average] supp=1 conf=0,50
[P2=A] -> [P1=average, P2=B] supp=1 conf=0,50
[P2=A] -> [P1=good, V2=over] supp=1 conf=0,50
```

```
[P1=good] -> [P2=B, P2=C] supp=1 conf=0,50
[P1=good] -> [P2=A, V2=over] supp=1 conf=0,50
```

The first line displays a rule stating that items annotated with ***C*** for the attribute ***P2*** are also annotated ***poor*** for the attribute ***P1*** and ***under*** for the attribute ***V1***. The support of this rule is 1 and the confidence is equals to 33%