

前提：已经完成data profiling部署步骤，即完成Metanome, metanome-algorithms,metanome-ms,metanome-cli四个项目的部署。

命令行使用的是metanome-cli的框架，利用metanome-cli的jar包进行命令行使用，以BINDER算法为例，详细参数及步骤 如下：

1. 准备阶段：

- metanome-cli产生的jar包

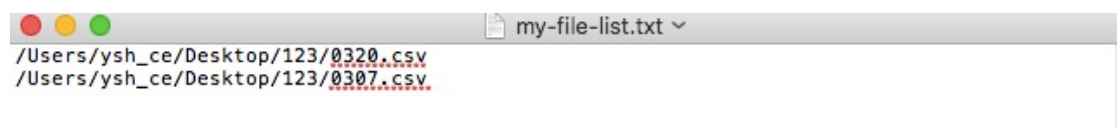
从metanome-cli/target/目录下拷贝jar包**metanome-cli-1.1.1-SNAPSHOT.jar**到run_data文件夹中。

- BINDER算法产生的jar包

从metanome-algorithms/_COLLECTION_目录下拷贝相应的jar包（以BINDER算法为例）**BINDER-1.1-SNAPSHOT.jar**到run_data文件中。

- 要出了数据的路径列表my-files-list.txt

在run_data文件中创建my-files-list.txt,并将要处理的**csv完整路径**添加到my-files-list.txt中，每一完整路径独占一行。如下格式：



2. 命令行介绍：

```
java -cp metanome-cli-1.1.1-SNAPSHOT.jar::BINDER-1.1-SNAPSHOT.jar  
de.metanome.cli.App
```

```
--algorithm de.metanome.algorithms.binder.BINDERFile
```

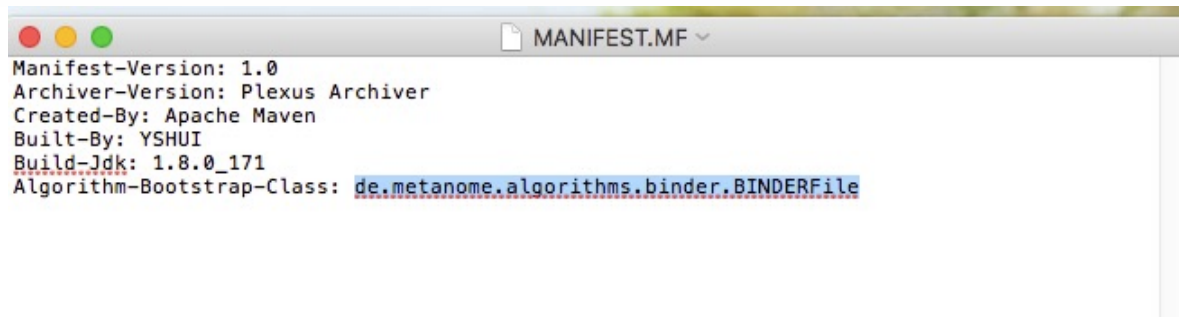
```
--algorithm-config INPUT_ROW_LIMIT:-1 TEMP_FOLDER_PATH:BINDER CLEAN_TEMP:true  
DETECT_NARY:false MAX_NARY_LEVEL:-1 FILTER_KEY_FOREIGNKEYS:false  
NUM_BUCKETS_PER_COLUMN:10 MEMORY_CHECK_FREQUENCY:100  
MAX_MEMORY_USAGE_PERCENTAGE:60
```

```
--file-key INPUT_FILES
```

```
--files load:my-files-list.txt
```

```
--output print
```

- 其中**标黄**部分为不可缺少固定部分，无需改动；
- 其中**下划线**部分为不可缺少部分，但是又跟所用算法部分相对应。其中BINDER-1.1-SNAPSHOT.jar为所要执行的算法相对应的jar包名。de.metanome.algorithms.binder.BINDERFile是利用解压软件将BINDER-1.1-SNAPSHOT.jar打开，打开BINDER-1.1-SNAPSHOT/META-INF/MANIFEST.MF，对应的Algorithm-Bootstrap-Class:**de.metanome.algorithms.binder.BINDERFile**

A screenshot of a code editor window titled 'MANIFEST.MF'. The window contains the following text: Manifest-Version: 1.0, Archiver-Version: Plexus Archiver, Created-By: Apache Maven, Built-By: YSHUI, Build-Jdk: 1.8.0_171, and Algorithm-Bootstrap-Class: de.metanome.algorithms.binder.BINDERFile. The last line is highlighted with a blue selection background.

```
Manifest-Version: 1.0
Archiver-Version: Plexus Archiver
Created-By: Apache Maven
Built-By: YSHUI
Build-Jdk: 1.8.0_171
Algorithm-Bootstrap-Class: de.metanome.algorithms.binder.BINDERFile
```

- **--algorithm-config** 对应的为算法参数设定（有默认值，可省略）。格式为 参数名1:参数值1 参数名2:参数值2，参数之间由空格隔开。具体参数需要从每个算法对应的源代码中寻找(需要从参数传递过程中寻找对应参数，有一种方法是利用web界面上的参数对应的寻找，如算法BINDER对应的web参数设定界面，为保证正确性，建议使用默认值)。

Additional configuration

TEMP_FOLDER_PATH

BINDER_temp

INPUT_ROW_LIMIT

-1

MAX_NARY_LEVEL

-1

NUM_BUCKETS_PER_COLUMN

10

MEMORY_CHECK_FREQUENCY

100

MAX_MEMORY_USAGE_PERCENTAGE

60

☒ CLEAN_TEMP

☐ DETECT_NARY

☐ FILTER_KEY_FOREIGNKEYS

Result handling

☒ Cache result and write it to disk when the algorithm is finished

☐ Write result immediately to disk.

☐ Just count the results.

Memory (in MB):

TEMP_FOLDER_PATH:BINDER——产生临时文件的路径，默认为BINDER_temp(根据算法不同默认路径不同)

INPUT_ROW_LIMIT:-1——默认为-1，即输入文档没有行数限制

MAX_NARY_LEVEL:-1——默认为-1，表示1-ary的包含依赖关系的处理，2——表示最多两个元素进行包含关系检测。

NUM_BUCKETS_PER_COLUMN:10——默认为10，每列的bucket数量（设定不同的值，发现结果没变化）

MEMORY_CHECK_FREQUENCY:100——内存检测频率

MAX_MEMORY_USAGE_PERCENTAGE:60 ——最大内存使用占比

——以上六个参数，对算法结果有影响的只有MAX_NARY_LEVEL，其他保存默认即可。

CLEAN_TEMP:true——是否清除临时文件，默认为true

DETECT_NARY:false——暂时还没搞懂是什么作用。

FILTER_KEY_FOREIGNKEYS:false——是否使用foreign keys

———综上9个参数，唯一对结果有形象的是MAX_NARY_LEVEL，DETECT_NARY，FILTER_KEY_FOREIGNKEYS，其他参数保持默认即可。

- --file-key 这里也是从源代码中寻找到的参数：INPUT_FILES（在metanome-algorithms/BINDER/BINDERFilesrc/main/java/de/metanome/algorithms/binder/BINDERFile.java的方法Identifier中可以查找到）

注：因为处理文件都是csv，所以这里不介绍--table-key和

- —output(或-o) 默认输出为file文件，可省略。有三种方式：**print** 即在终端中打印出，**file**生成文件file，**crate**:目前不知道这个怎么用。

综上命令行使用如下：

```
1 java -cp metanome-cli-1.1.1-SNAPSHOT.jar:BINDER-1.1-SNAPSHOT.jar
  de.metanome.cli.App --algorithm de.metanome.algorithms.binder.BINDERFile
  --file-key INPUT_FILES --files load:my-file-list.txt
```

对共享平台数据的0307.csv和0320.csv文档进行BINDER算法处理结果如下：

```
$ java -cp metanome-cli-1.1.1-SNAPSHOT.jar:BINDER-1.1-SNAPSHOT.jar
de.metanome.cli.App --algorithm de.metanome.algorithms.binder.BINDERFile --file-
key INPUT_FILES --files load:my-file-list.txt -o print
```

```

Results:
[0320.csv.empty_value_column_count]=[0307.csv.empty_value_column_count]
[0320.csv.source]=[0307.csv.source]
[0320.csv.job_status]=[0307.csv.job_status]
[0320.csv.remark]=[0320.csv.name]
[0320.csv.remark]=[0320.csv.size]
[0320.csv.remark]=[0320.csv.row_count]
[0320.csv.remark]=[0320.csv.repeat_row_count]
[0320.csv.remark]=[0320.csv.column_count]
[0320.csv.remark]=[0320.csv.single_value_column_count]
[0320.csv.remark]=[0320.csv.empty_value_column_count]
[0320.csv.remark]=[0320.csv.table_type]
[0320.csv.remark]=[0320.csv.source]
[0320.csv.remark]=[0320.csv.job_status]
[0320.csv.remark]=[0320.csv.create_date]
[0320.csv.remark]=[0307.csv.name]
[0320.csv.remark]=[0307.csv.size]
[0320.csv.remark]=[0307.csv.row_count]
[0320.csv.remark]=[0307.csv.repeat_row_count]
[0320.csv.remark]=[0307.csv.column_count]
[0320.csv.remark]=[0307.csv.single_value_column_count]
[0320.csv.remark]=[0307.csv.empty_value_column_count]
[0320.csv.remark]=[0307.csv.table_type]
[0320.csv.remark]=[0307.csv.source]
[0320.csv.remark]=[0307.csv.job_status]
[0320.csv.remark]=[0307.csv.remark]
[0320.csv.remark]=[0307.csv.create_date]
[0320.csv.table_type]=[0307.csv.table_type]
[0320.csv.single_value_column_count]=[0307.csv.single_value_column_count]
[0307.csv.column_count]=[0320.csv.column_count]
[0307.csv.single_value_column_count]=[0320.csv.single_value_column_count]
[0307.csv.table_type]=[0320.csv.table_type]
[0307.csv.empty_value_column_count]=[0320.csv.empty_value_column_count]
[0307.csv.remark]=[0320.csv.name]
[0307.csv.remark]=[0320.csv.size]
[0307.csv.remark]=[0320.csv.row_count]
[0307.csv.remark]=[0320.csv.repeat_row_count]
[0307.csv.remark]=[0320.csv.column_count]
[0307.csv.remark]=[0320.csv.single_value_column_count]
[0307.csv.remark]=[0320.csv.empty_value_column_count]
[0307.csv.remark]=[0320.csv.table_type]
[0307.csv.remark]=[0320.csv.source]
[0307.csv.remark]=[0320.csv.job_status]
[0307.csv.remark]=[0320.csv.remark]
[0307.csv.remark]=[0320.csv.create_date]
[0307.csv.remark]=[0307.csv.name]
[0307.csv.remark]=[0307.csv.size]
[0307.csv.remark]=[0307.csv.row_count]
[0307.csv.remark]=[0307.csv.repeat_row_count]
[0307.csv.remark]=[0307.csv.column_count]
[0307.csv.remark]=[0307.csv.single_value_column_count]
[0307.csv.remark]=[0307.csv.empty_value_column_count]
[0307.csv.remark]=[0307.csv.table_type]
[0307.csv.remark]=[0307.csv.source]
[0307.csv.remark]=[0307.csv.job_status]
[0307.csv.remark]=[0307.csv.create_date]
[0307.csv.job_status]=[0320.csv.job_status]
[0307.csv.source]=[0320.csv.source]

```

如果选择--output file 则会在run_data中生成一个results文件，里面有此次执行结果的文件，内容与上图类似。

从结果可以看出，metanome-cli这个项目只返回最终结果，没有返回相应的概率值。

可能有两个原因：1) metanome-cli确实只返回的是这种不知道如何解释的结果。2) metanome-cli产生的结果需要进一步处理，而metanome-cli未集成，或者留有接口但未设定。

接下来工作：

- 1) 寻找metanome-cli的展示概率大小的接口。如metanone-cli为设定概率输出的端口，尝试从Metanome中寻找对应的端口。
- 2) 对参数意义进一步探究，尽快确定BINDER对应参数的意义。