

# EC559 DATA MINING

## Spring 2014

### In Class Practice Session 2: Association Rule

This example illustrates some of the basic elements of associate rule mining using WEKA. The sample data set used for this example, unless otherwise indicated, is the "bank data" described in Data Preprocessing. In this case, our starting point is the discretized data obtained after performing the preprocessing tasks. Figure a1 shows the WEKA explorer interface after opening this data file ("bank-data-final.arff").

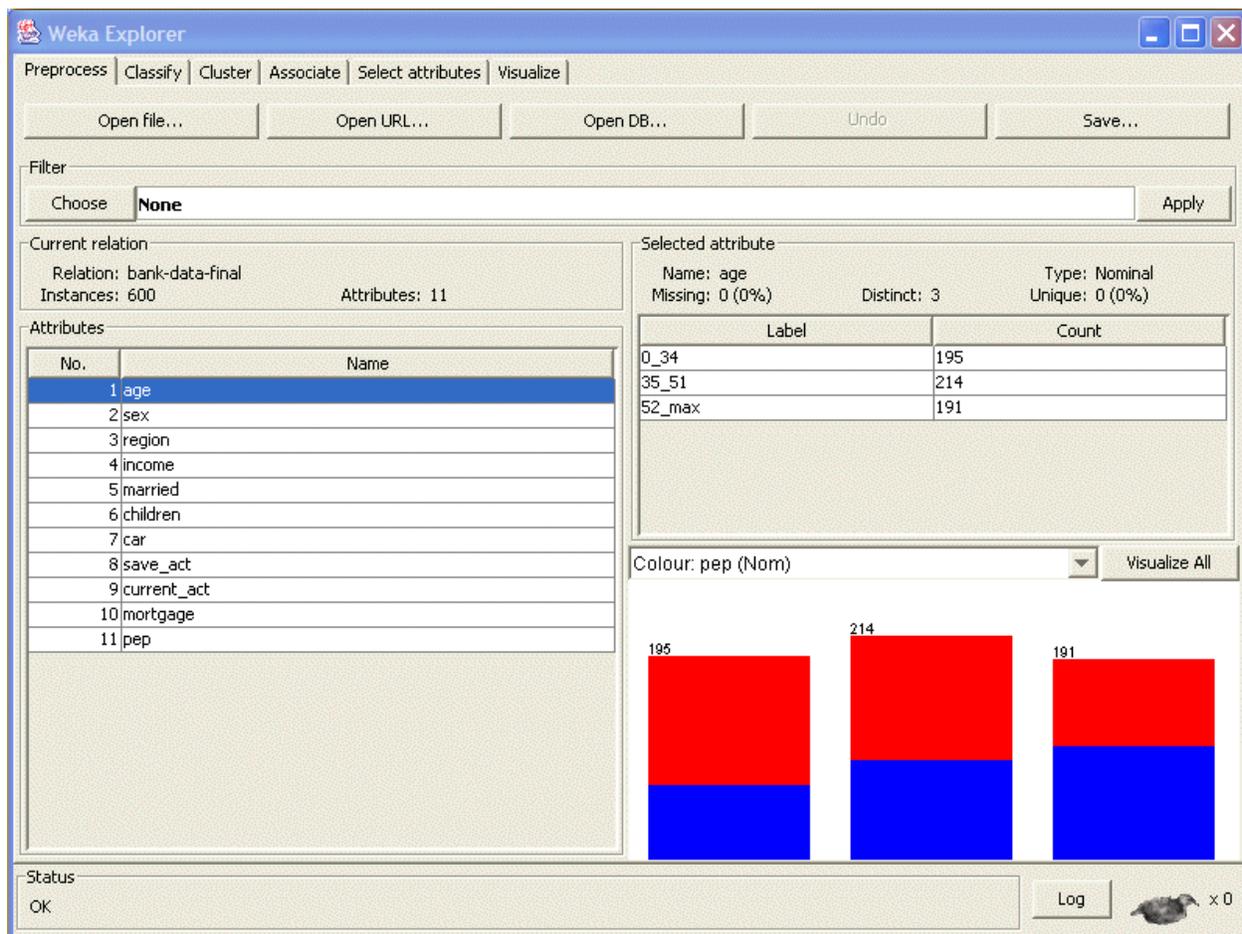
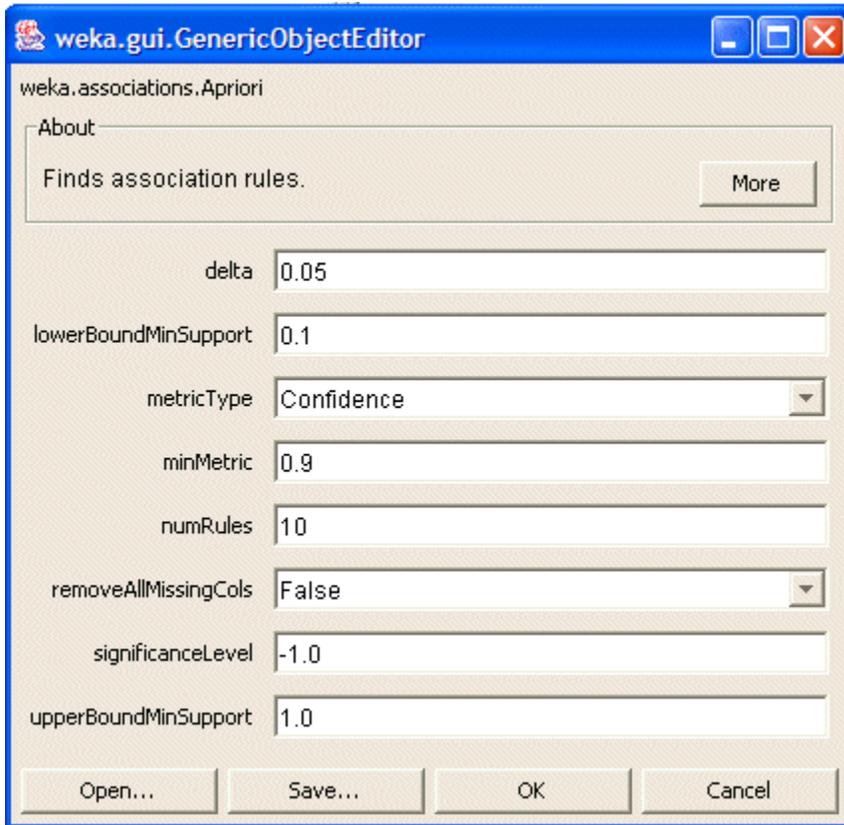


Figure a1

Clicking on the "Associate" tab will bring up the interface for the association rule algorithms. The Apriori algorithm which we will use is the default algorithm selected. However, in order to change the parameters for this run (e.g., support, confidence, etc.) we click on the text box immediately to the right of the "Choose" button. Note that this box, at any given time, shows the specific command line arguments that are to be used for the algorithm. The dialog box for changing the parameters is depicted in Figure a2. Here, you can specify various parameters associated with Apriori. Click on the "More" button to see the synopsis for the different parameters.



[Figure a2](#)

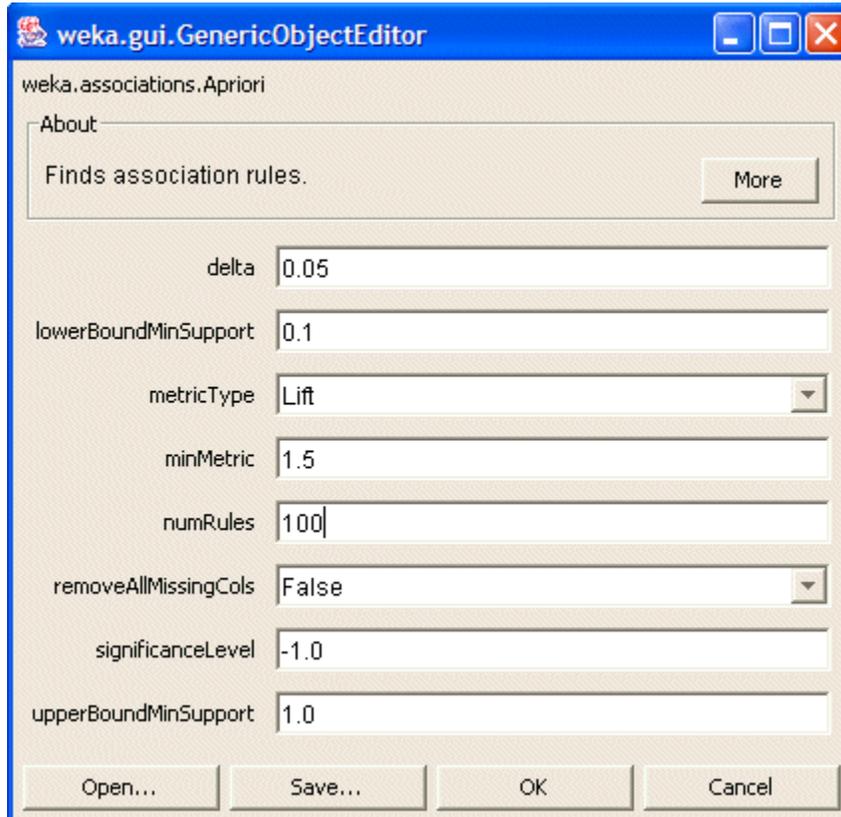
WEKA allows the resulting rules to be sorted according to different metrics such as confidence, leverage, and lift. In this example, we have selected lift as the criteria. Furthermore, we have entered 1.5 as the minimum value for lift (or improvement) is computed as the confidence of the rule divided by the support of the right-hand-side (RHS). In a simplified form, given a rule  $L \Rightarrow R$ , *lift* is the ratio of the probability that L and R occur together to the multiple of the two individual probabilities for L and R, i.e.,

$$\textit{lift} = \Pr(L,R) / \Pr(L).\Pr(R).$$

If this value is 1, then L and R are independent. The higher this value, the more likely that the existence of L and R together in a transaction is not just a random occurrence, but because of some relationship between them.

Here we also change the default value of rules (10) to be 100; this indicates that the program will report no more than the top 100 rules (in this case sorted according to their lift values). The upper bound for minimum support is set to 1.0 (100%) and the lower bound to 0.1 (10%). Apriori in WEKA starts with the upper bound support and incrementally decreases support (by delta increments which by default are set to 0.05 or 5%). The algorithm halts when either the specified number of rules is generated, or the lower bound for min. support is reached. The significance testing option is only applicable in the case of confidence and is by default not used (-1.0).

The final selection of parameters for our current run is depicted in Figure a3:



[Figure a3](#)

Once the parameters have been set, the command line text box will show the new command line. We now click on start to run the program. This results in a set of rules as depicted in Figure a4.

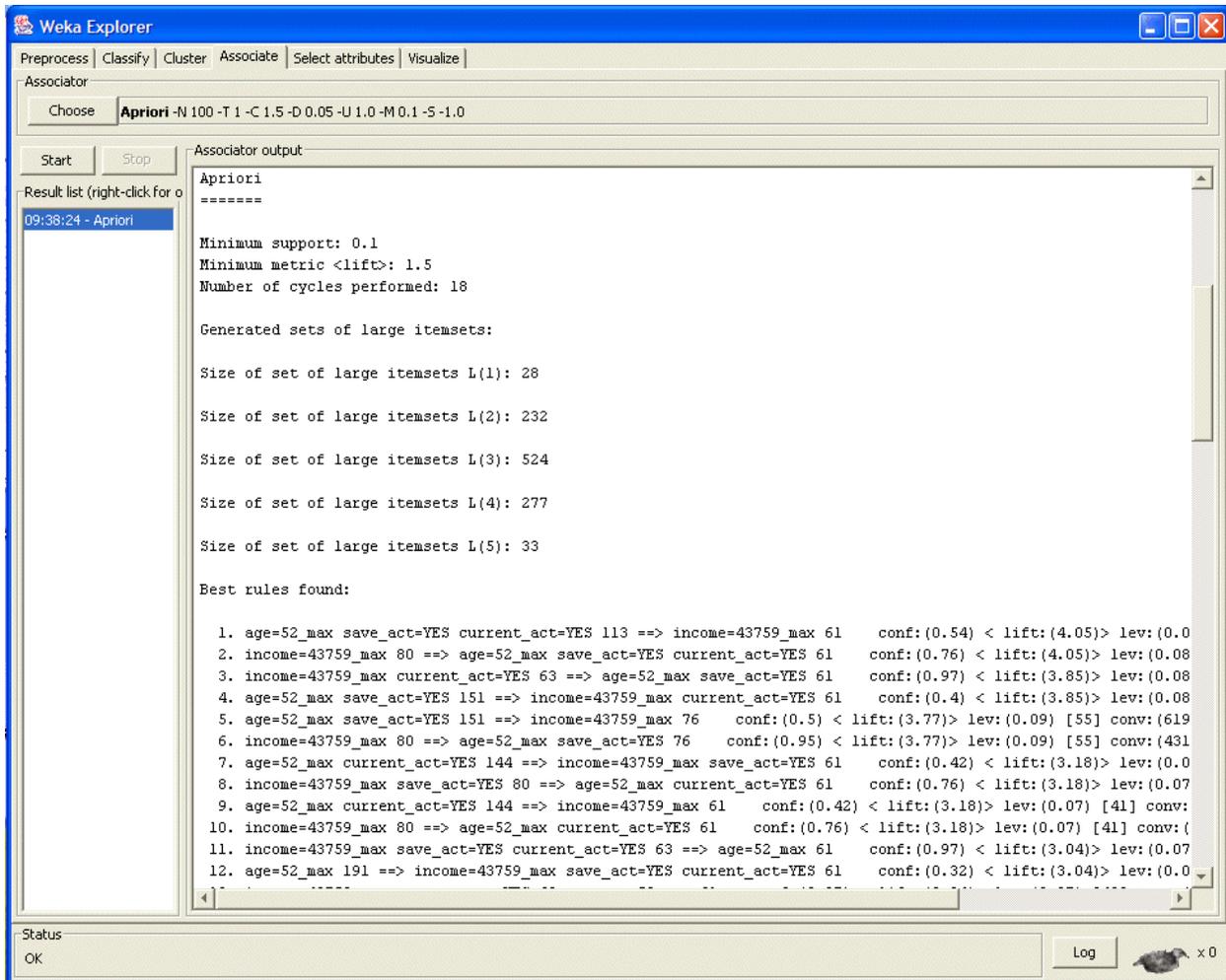


Figure a4

The panel on the left ("Result list") now shows an item indicating the algorithm that was run and the time of the run. You can perform multiple runs in the same session each time with different parameters. Each run will appear as an item in the Result list panel. Clicking on one of the results in this list will bring up the details of the run, including the discovered rules in the right panel. In addition, right-clicking on the result set allows us to save the result buffer into a separate file. In this case, we save the output in the file [bank-data-ar1.txt](#). A portion of this file is depicted in Figure a5:

```

2
3 Scheme:          weka.associations.Apriori -N 100 -T 1 -C 1.5 -D 0.05 -U 1.0 -M 0.1 -S -1.0
4 Relation:       bank-data-final
5 Instances:      600
6 Attributes:     11
7                age
8                sex
9                region
10               income
11               married
12               children
13               car
14               save_act
15               current_act
16               mortgage
17               pep
18 === Associator model (full training set) ===
19
20
21 Apriori
22 =====
23
24 Minimum support: 0.1
25 Minimum metric <lift>: 1.5
26 Number of cycles performed: 18
27
28 Generated sets of large itemsets:
29
30 Size of set of large itemsets L(1): 28
31
32 Size of set of large itemsets L(2): 232
33
34 Size of set of large itemsets L(3): 524
35
36 Size of set of large itemsets L(4): 277
37
38 Size of set of large itemsets L(5): 33
39
40 Best rules found:
41
42 1. age=52_max save_act=YES current_act=YES 113 ==> income=43759_max 61      conf:(0.54) < lift:(4.05)> lev:(0.08) [
43 2. income=43759_max 80 ==> age=52_max save_act=YES current_act=YES 61      conf:(0.76) < lift:(4.05)> lev:(0.08) [4
44 3. income=43759_max current_act=YES 63 ==> age=52_max save_act=YES 61      conf:(0.97) < lift:(3.85)> lev:(0.08) [4
45 4. age=52_max save_act=YES 151 ==> income=43759_max current_act=YES 61      conf:(0.4) < lift:(3.85)> lev:(0.08) [4
46 5. age=52_max save_act=YES 151 ==> income=43759_max 76      conf:(0.5) < lift:(3.77)> lev:(0.09) [55] conv:(619894.
47 6. income=43759_max 80 ==> age=52_max save_act=YES 76      conf:(0.95) < lift:(3.77)> lev:(0.09) [55] conv:(4310400
48 7. age=52_max current_act=YES 144 ==> income=43759_max save_act=YES 61      conf:(0.42) < lift:(3.18)> lev:(0.07) [4
49 8. income=43759_max save_act=YES 80 ==> age=52_max current_act=YES 61      conf:(0.76) < lift:(3.18)> lev:(0.07) [4
50 9. age=52_max current_act=YES 144 ==> income=43759_max 61      conf:(0.42) < lift:(3.18)> lev:(0.07) [41] conv:(534
51 10. income=43759_max 80 ==> age=52_max current_act=YES 61      conf:(0.76) < lift:(3.18)> lev:(0.07) [41] conv:(1094
52 11. income=43759_max save_act=YES current_act=YES 63 ==> age=52_max 61      conf:(0.97) < lift:(3.04)> lev:(0.07) [4
53 12. age=52_max 191 ==> income=43759_max current_act=YES 61      conf:(0.32) < lift:(3.04)> lev:(0.07) [40] conv:(469
54 13. income=43759_max current_act=YES 63 ==> age=52_max 61      conf:(0.97) < lift:(3.04)> lev:(0.07) [40] conv:(5153
55 14. age=52_max 191 ==> income=43759_max current_act=YES 61      conf:(0.32) < lift:(3.04)> lev:(0.07) [40] conv:(469
56 15. income=43759_max save_act=YES 80 ==> age=52_max 76      conf:(0.95) < lift:(2.98)> lev:(0.08) [50] conv:(3926400
57 16. age=52_max 191 ==> income=43759_max save_act=YES 76      conf:(0.4) < lift:(2.98)> lev:(0.08) [50] conv:(513724.
58 17. income=43759_max 80 ==> age=52_max 76      conf:(0.95) < lift:(2.98)> lev:(0.08) [50] conv:(3926400)
59 18. age=52_max 191 ==> income=43759_max 76      conf:(0.4) < lift:(2.98)> lev:(0.08) [50] conv:(513724.14)
60 19. income=0_24386 current_act=YES pep=NO 132 ==> age=0_34 save_act=YES 60      conf:(0.45) < lift:(2.29)> lev:(0.06
61 20. age=0_34 save_act=YES 119 ==> income=0_24386 current_act=YES pep=NO 60      conf:(0.5) < lift:(2.29)> lev:(0.06)
62 21. age=0_34 current_act=YES pep=NO 95 ==> income=0_24386 save_act=YES 60      conf:(0.63) < lift:(2.22)> lev:(0.05)
63 22. income=0_24386 save_act=YES 171 ==> age=0_34 current_act=YES pep=NO 60      conf:(0.35) < lift:(2.22)> lev:(0.05)
64 23. age=0_34 pep=NO 124 ==> income=0_24386 save_act=YES current_act=YES 60      conf:(0.48) < lift:(2.22)> lev:(0.05)
65 24. income=0_24386 save_act=YES current_act=YES 132 ==> age=0_34 pep=NO 60      conf:(0.45) < lift:(2.22)> lev:(0.05)
66 25. income=0_24386 current_act=YES pep=NO 132 ==> age=0_34 married=YES 61      conf:(0.46) < lift:(2.17)> lev:(0.05)
67 26. age=0_34 married=YES 128 ==> income=0_24386 current_act=YES pep=NO 61      conf:(0.48) < lift:(2.17)> lev:(0.05)

```

Figure a5

Note that the rules were discovered based on the specified threshold values for support and lift. For each rule, the frequency counts for the LHS and RHS of each rule is given, as well as the values for confidence, lift, leverage, and conviction. Note that *leverage* and lift measure similar things, except that leverage measures the difference between the probability of co-occurrence of L and R (see above example) as the independent probabilities of each of L and R, i.e.,

$$\textit{leverage} = \Pr(L,R) - \Pr(L).\Pr(R).$$

In other words, leverage measures the proportion of additional cases covered by both L and R above those expected if L and R were independent of each other. Thus, for leverage, values above 0 are desirable, whereas for lift, we want to see values greater than 1. Finally, *conviction* is

similar to lift, but it measures the effect of the right-hand-side not being true. It also inverts the ratio. So, conviction is measured as:

$$\text{conviction} = \frac{\Pr(L) \cdot \Pr(\text{not } R)}{\Pr(L, R)}$$

Thus, conviction, in contrast to lift is not symmetric (and also has no upper bound).

In most cases, it is sufficient to focus on a combination of support, confidence, and either lift or leverage to quantitatively measure the "quality" of the rule. However, the real value of a rule, in terms of usefulness and action ability is subjective and depends heavily of the particular domain and business objectives.

### *Using the Command Line*

In general, using WEKA from the command line provides more flexibility than using the GUI version. In the case of association rules, the GUI version does not provide the ability to save the frequent itemsets. We can do this using the command line. If we look at the output of the association rule mining from the above example (the file [bank-data-ar1.txt](#)), the actual command line options are given under the "Run information" at the top. In the example, this command line is:

```
weka.associations.Apriori -N 100 -T 1 -C 1.5 -D 0.05 -U 1.0 -M 0.1 -S -1.0
```

We can use this directly using the "Simple CLI" interface.

In the main WEKA interface, click "Simple CLI" button to start the command line interface. The main command for generating the rules as we did above is:

```
java weka.associations.Apriori options -t directory-path\bank-data-final.arff
```

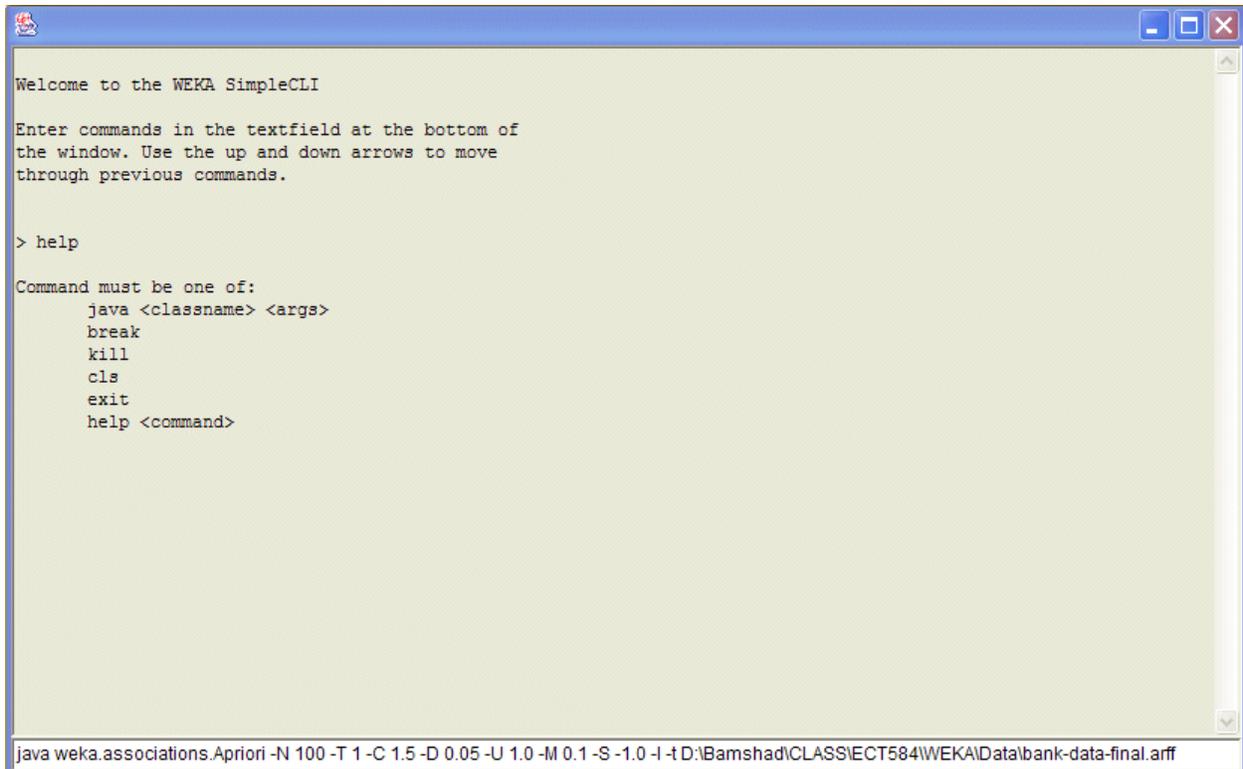
where the word *options* is replaced with the command line options, which for the above example are:

```
-N 100 -T 1 -C 1.5 -D 0.05 -U 1.0 -M 0.1 -S -1.0
```

The additional "-t *directory-path\bank-data-final.arff*" option tells WEKA to use the file "bank-data-final.arff" as the input file (located in the specified directory). This command will produce exactly the same output as the previous GUI example. However, we can add an additional option ("-I") which results in the generation of all frequent itemsets:

```
java weka.associations.Apriori options -I -t directory-path\bank-data-final.arff
```

This command as it is used in the SimpleCLI interface is depicted in Figure a6:



```
Welcome to the WEKA SimpleCLI

Enter commands in the textfield at the bottom of
the window. Use the up and down arrows to move
through previous commands.

> help

Command must be one of:
  java <classname> <args>
  break
  kill
  cls
  exit
  help <command>

java weka.associations.Apriori -N 100 -T 1 -C 1.5 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -I -I D:\Bamshad\CLASS\ECT584\WEKA\Data\bank-data-final.arff
```

[Figure a6](#)

When ready, press enter to run the program with the indicated options. The result of this command will be displayed in the top panel of the Simple CLI interface. Here, the results have been saved into a file [bank-data- ar2.txt](#). You will notice that before the rules, the output includes itemset of various sizes generated at different iterations of Apriori algorithm (in this case, L1 through L5) along with the support count for each itemset. In the case of L1, these are simply the individual items (attributes) that meet the minimum support threshold.