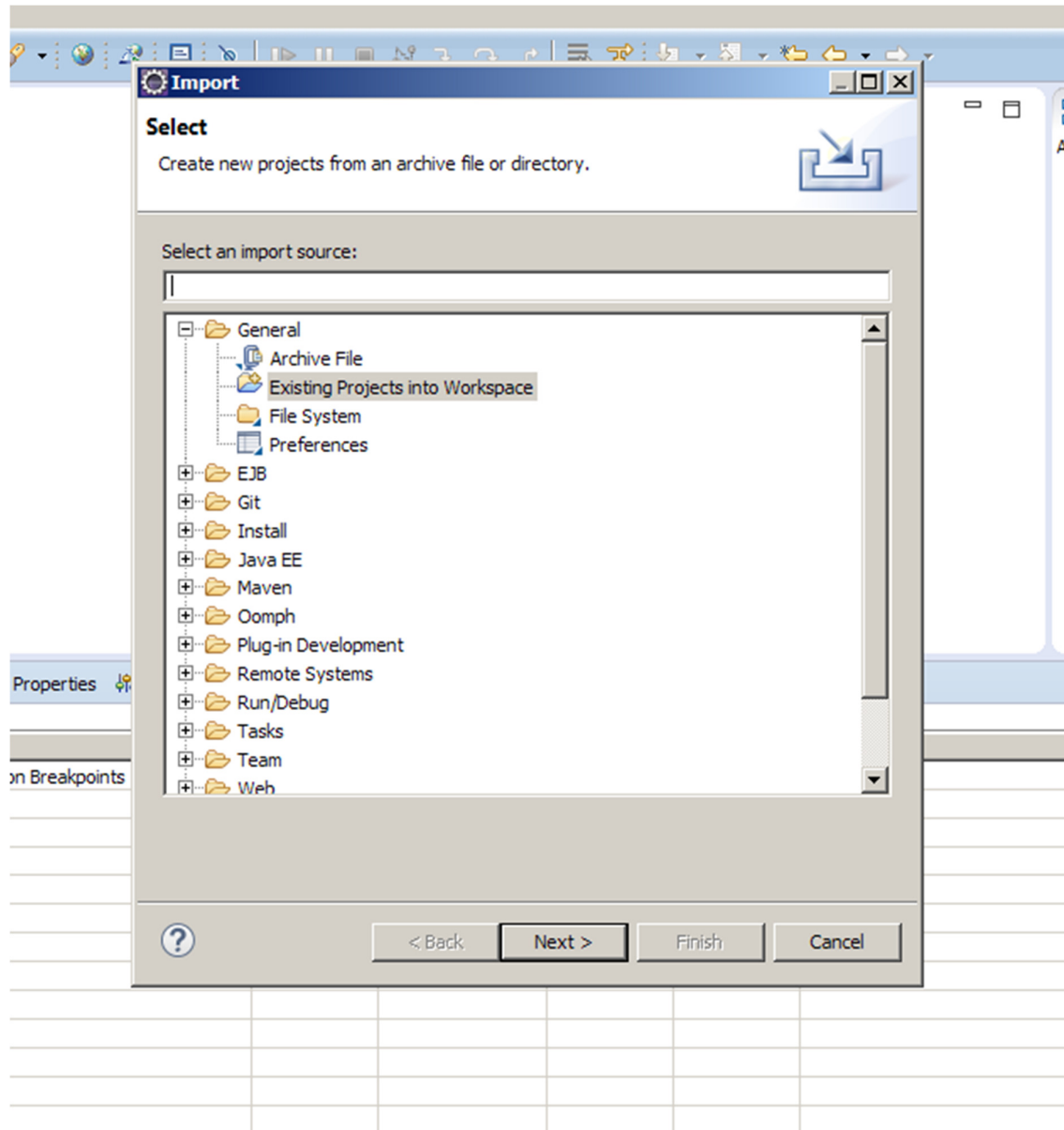
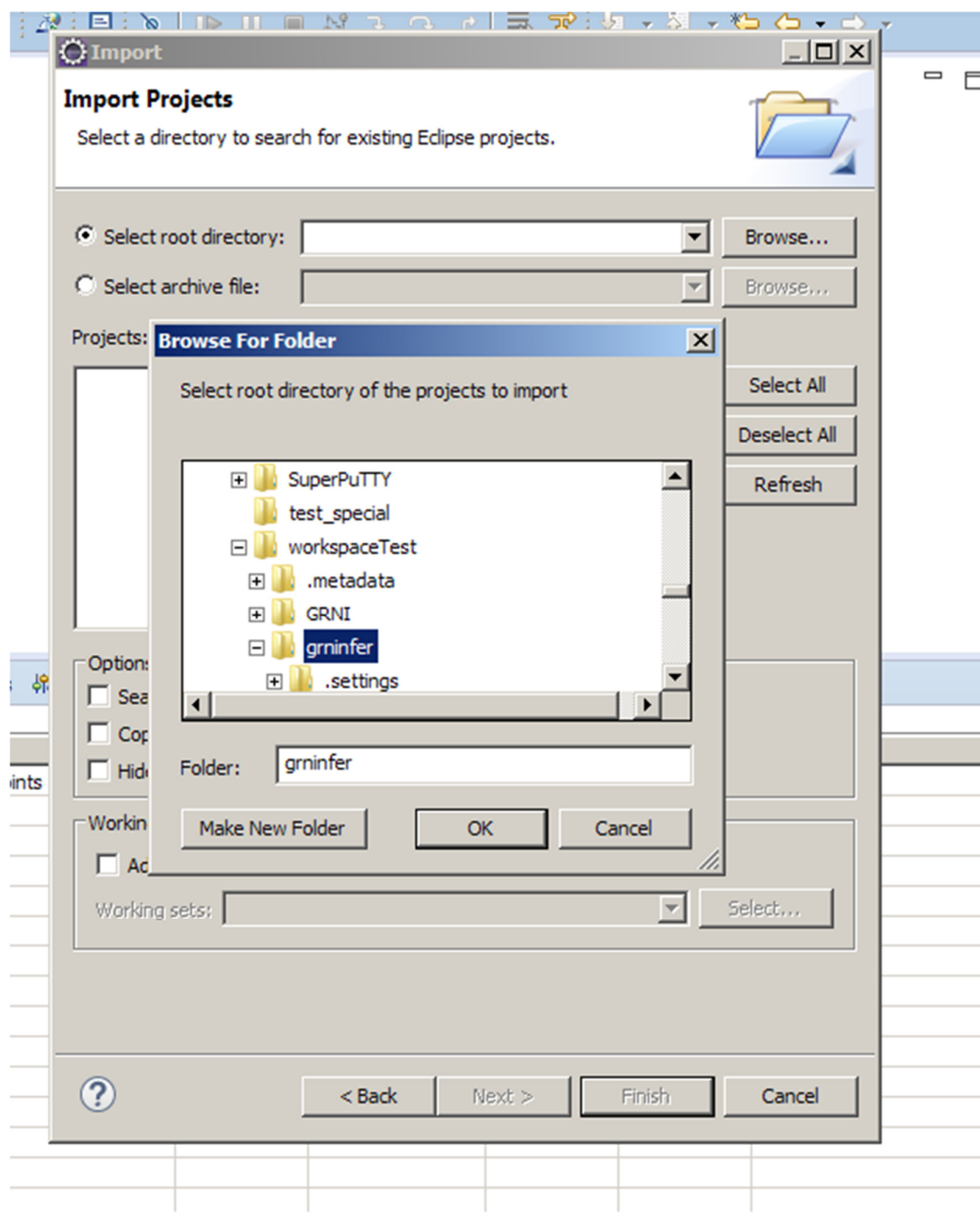


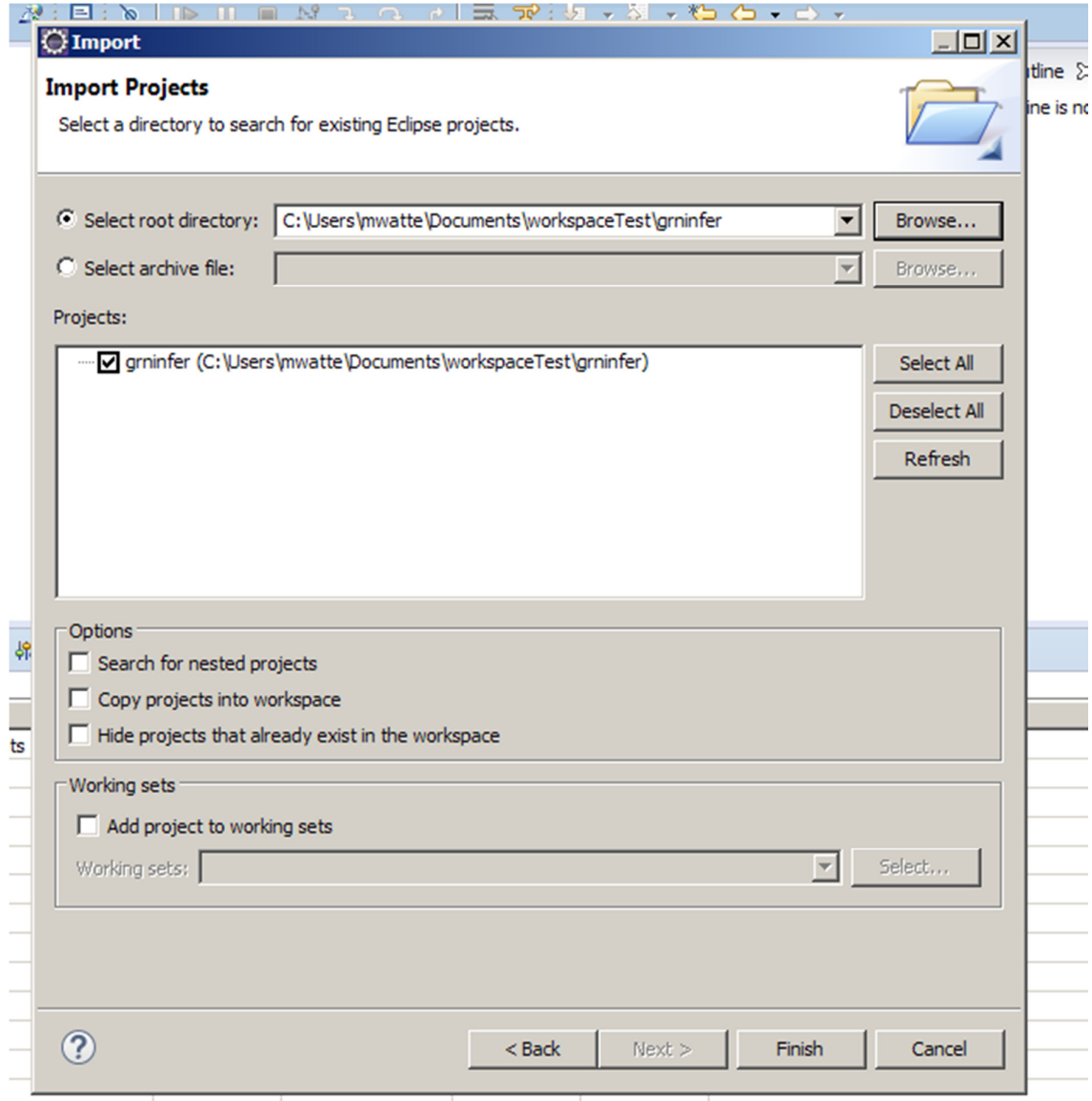
please follow below given steps to load grninfer project to eclipse and execute dbn network inference

1. start eclipse
2. import grninfer project to the eclipse as shown below

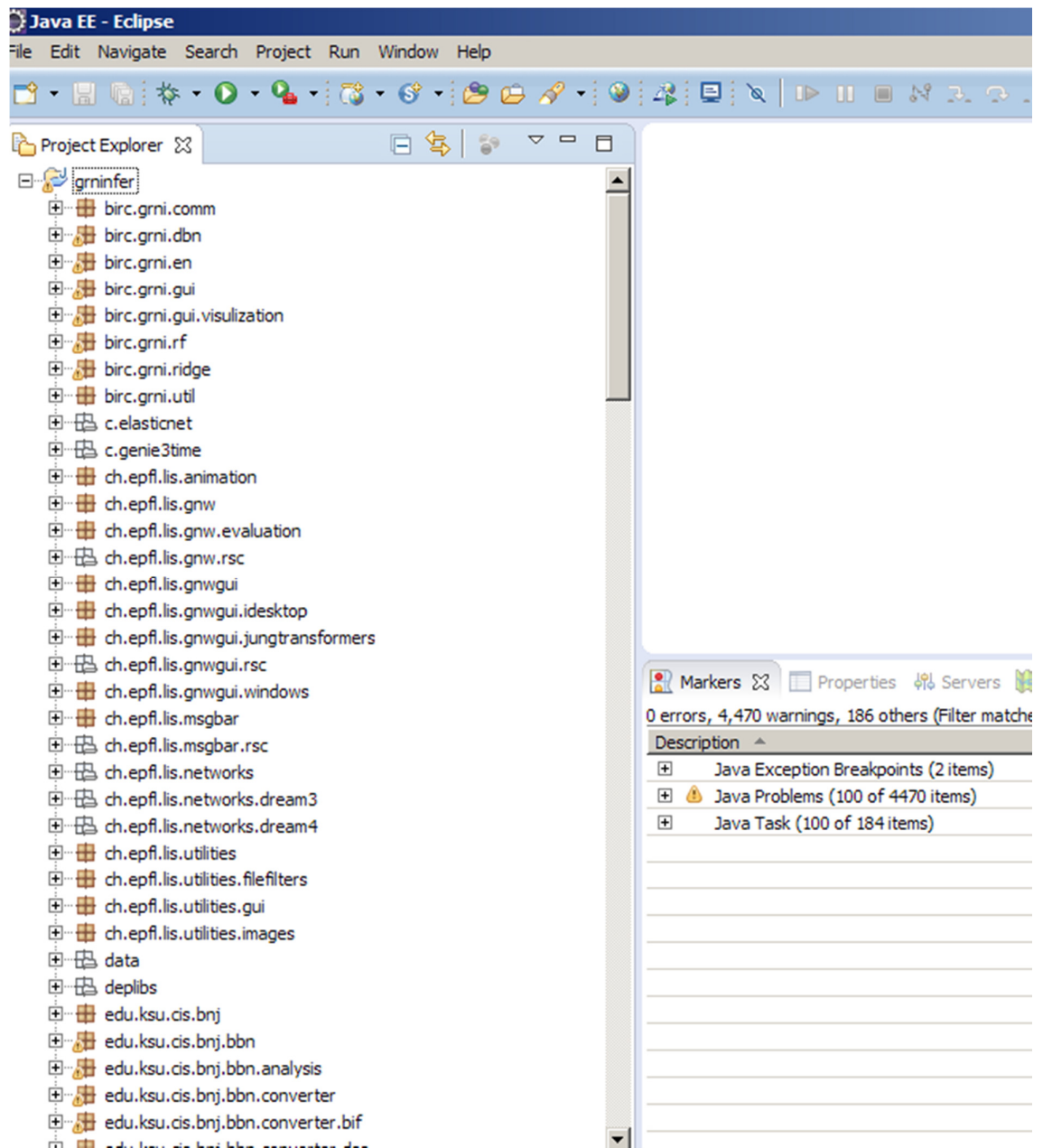


3. select existing project to eclipse and next window select your grninfer project folder

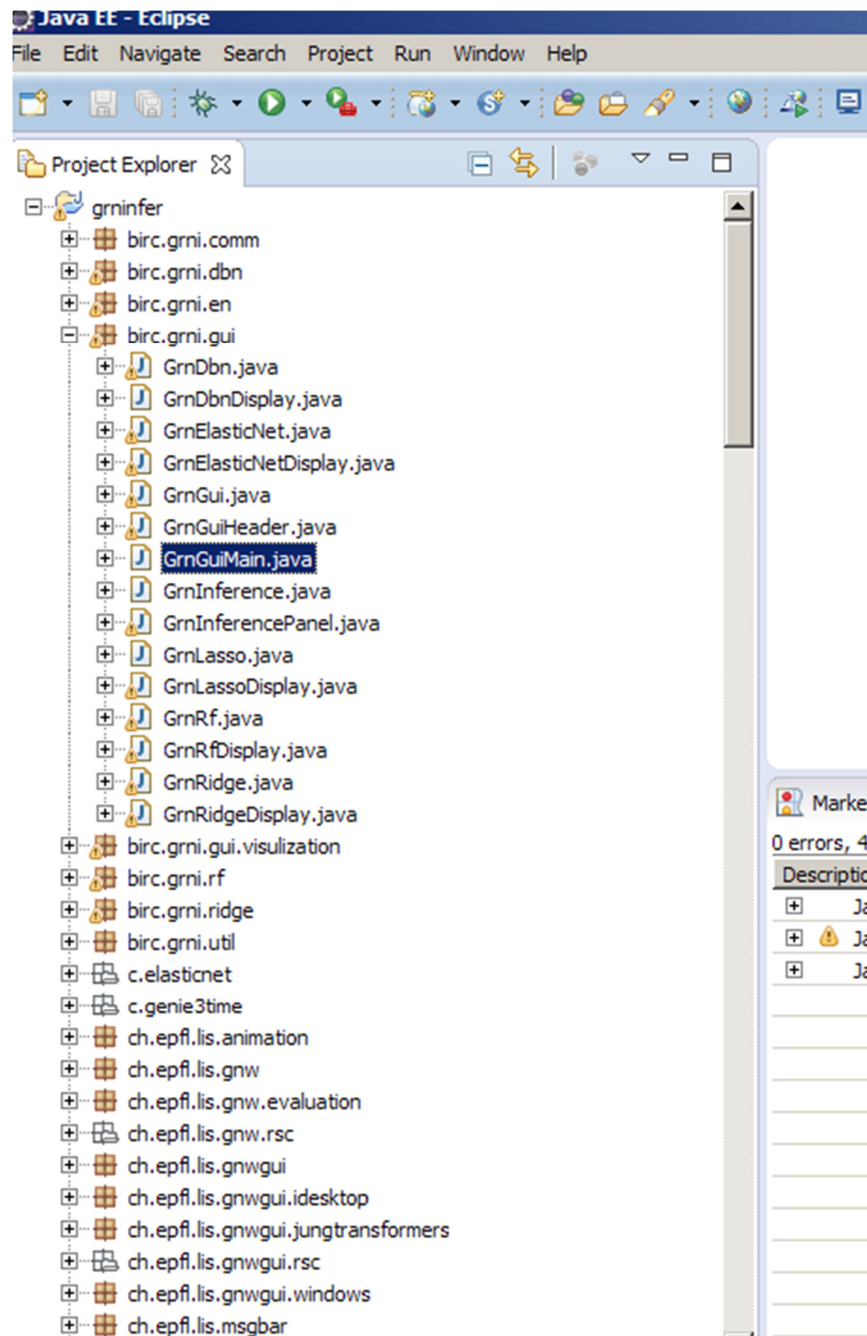




4. complete import by click finish. then eclipse will load grnifer project to the eclipse
5. after load complete you can see grnifer project like below

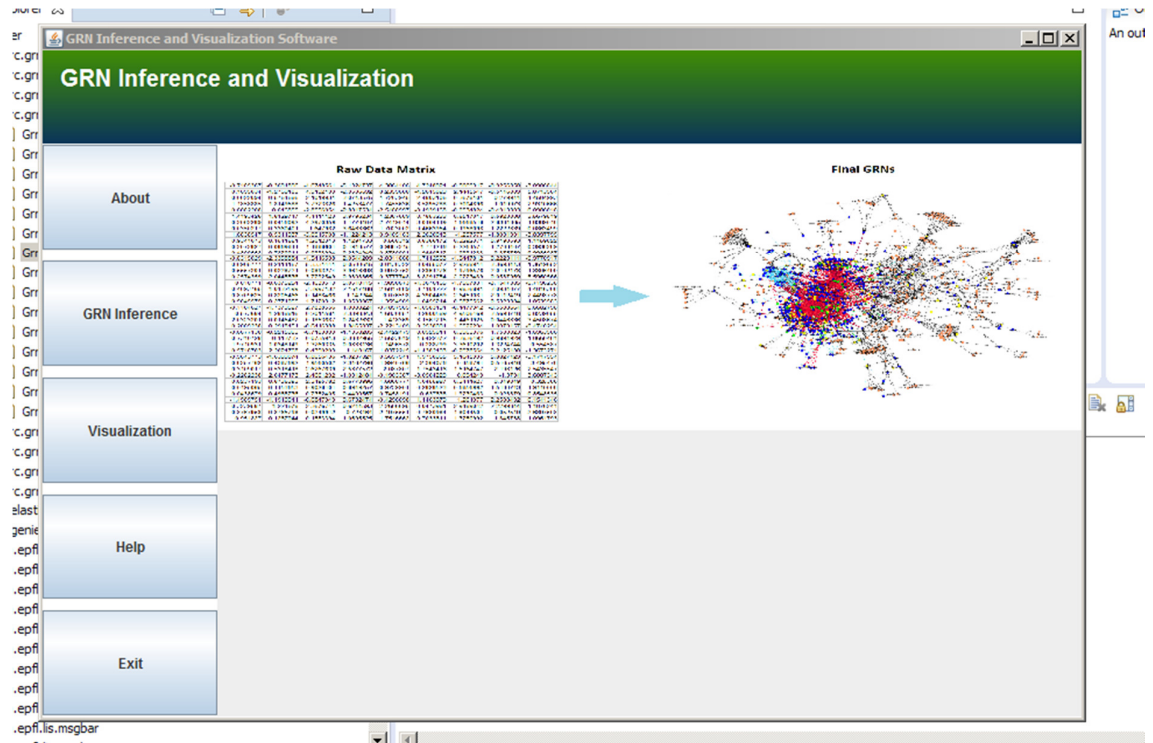


6. expand birc.grni.gui sub project and select GrnGuiMain.java class as shown below.
this class contains main method.

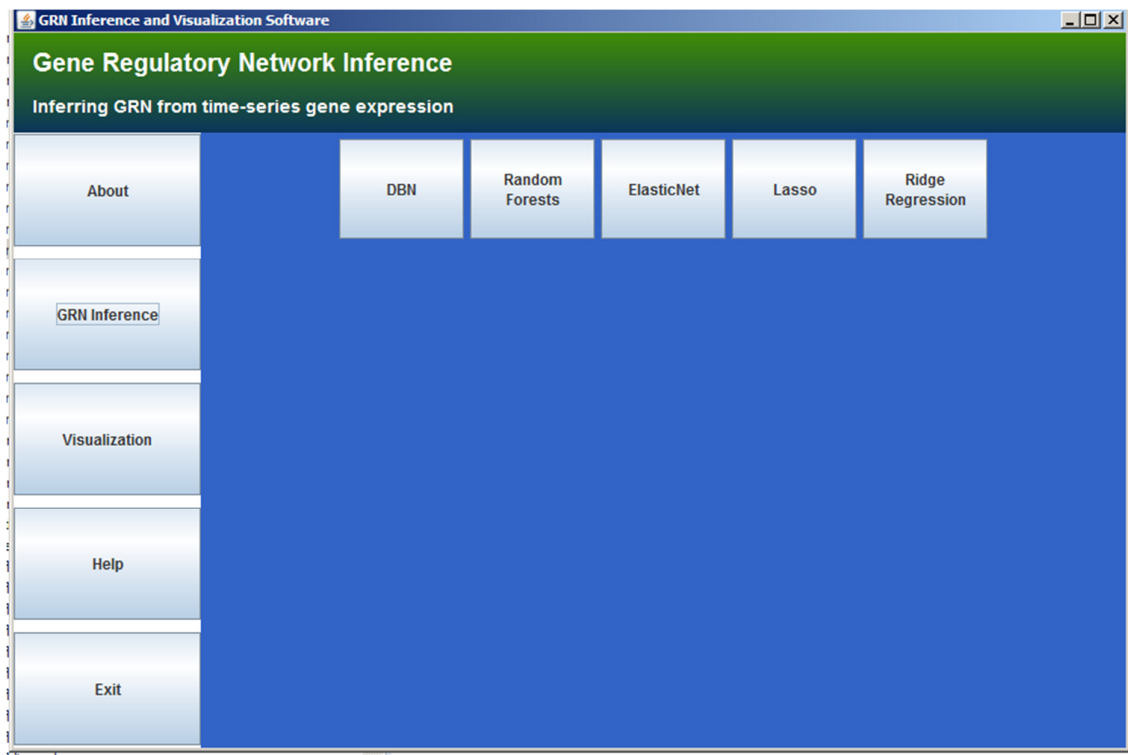


7. right click on GrnGuiMain.java → Run As → Java application

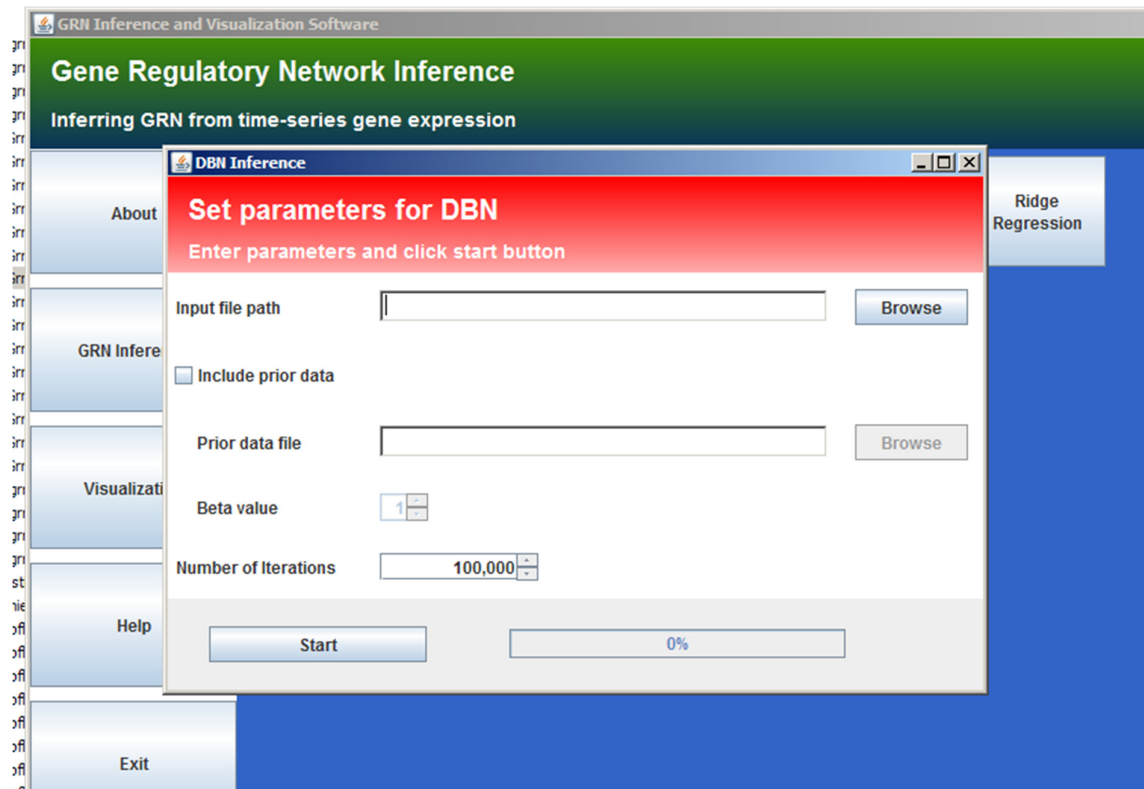
8. after main method complete, grninfering gui will start



9. click GRN Inference tab . this will pop up below window

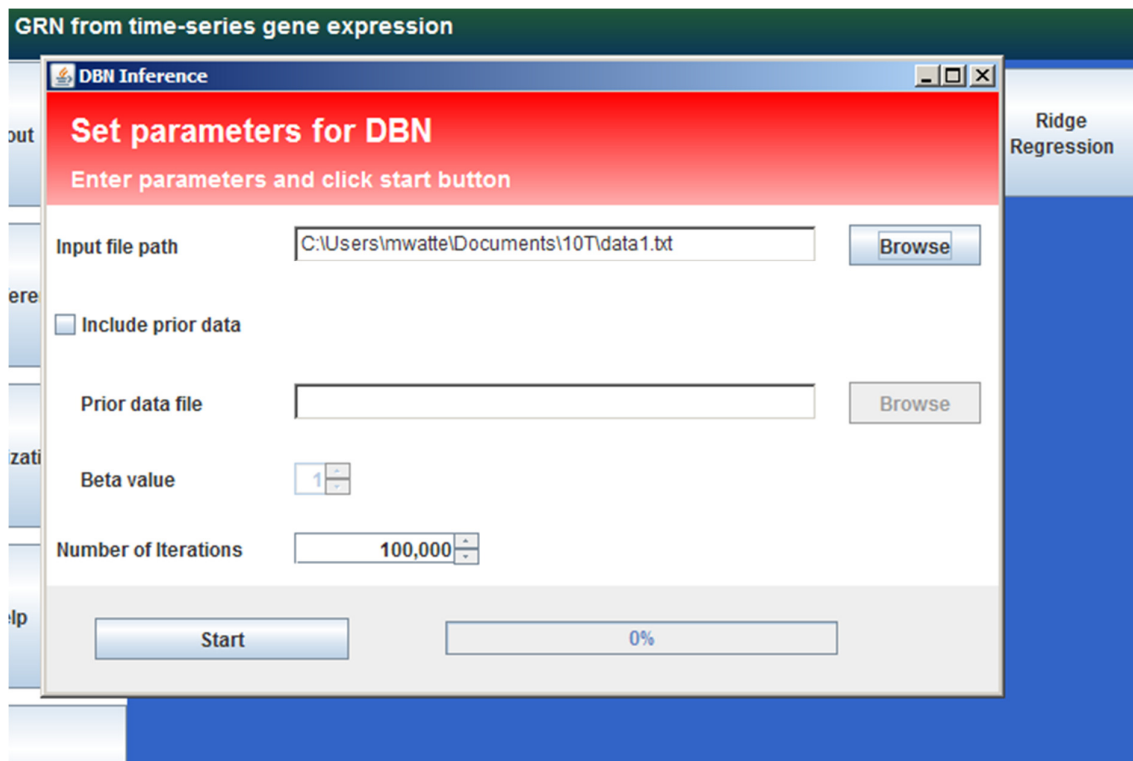
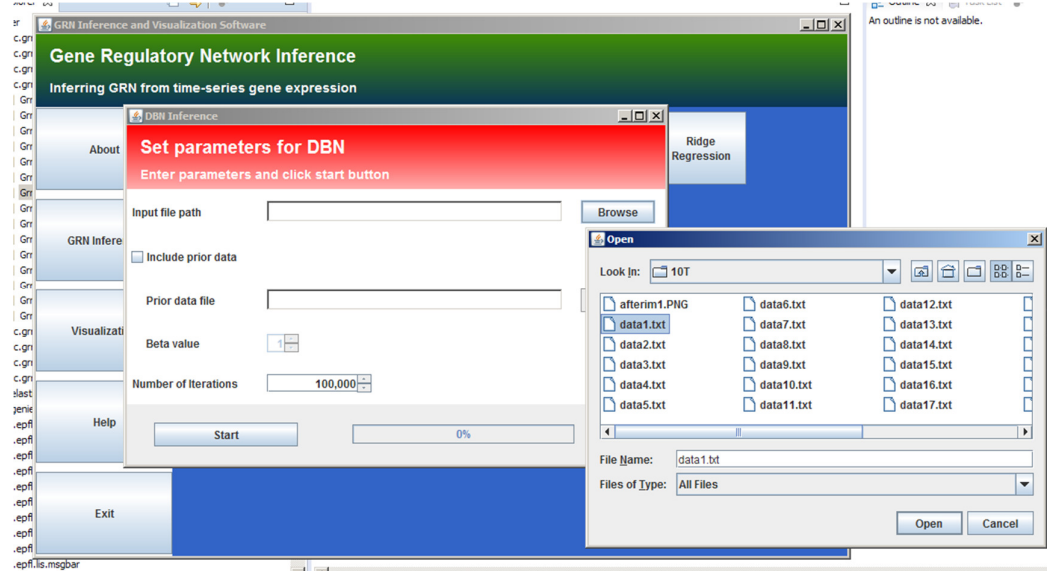


10. select dbn , this will show dbn parameters window



11. now fill dbn parameters

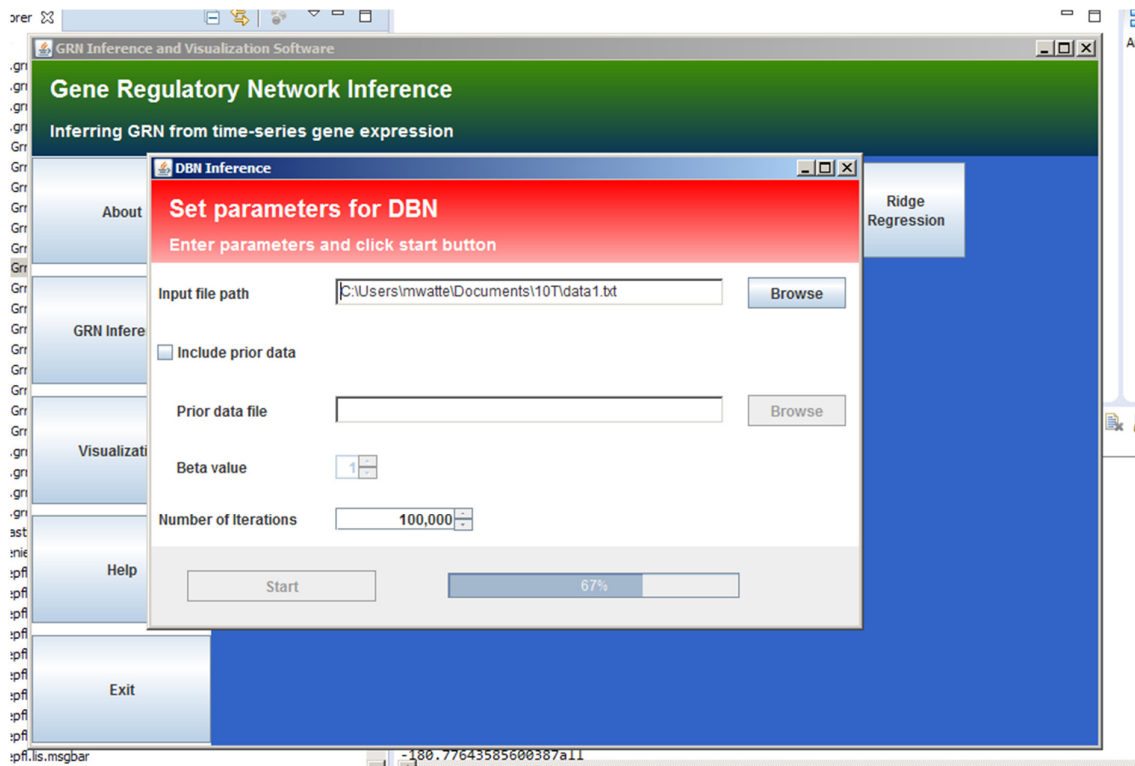
a. browse and select input file



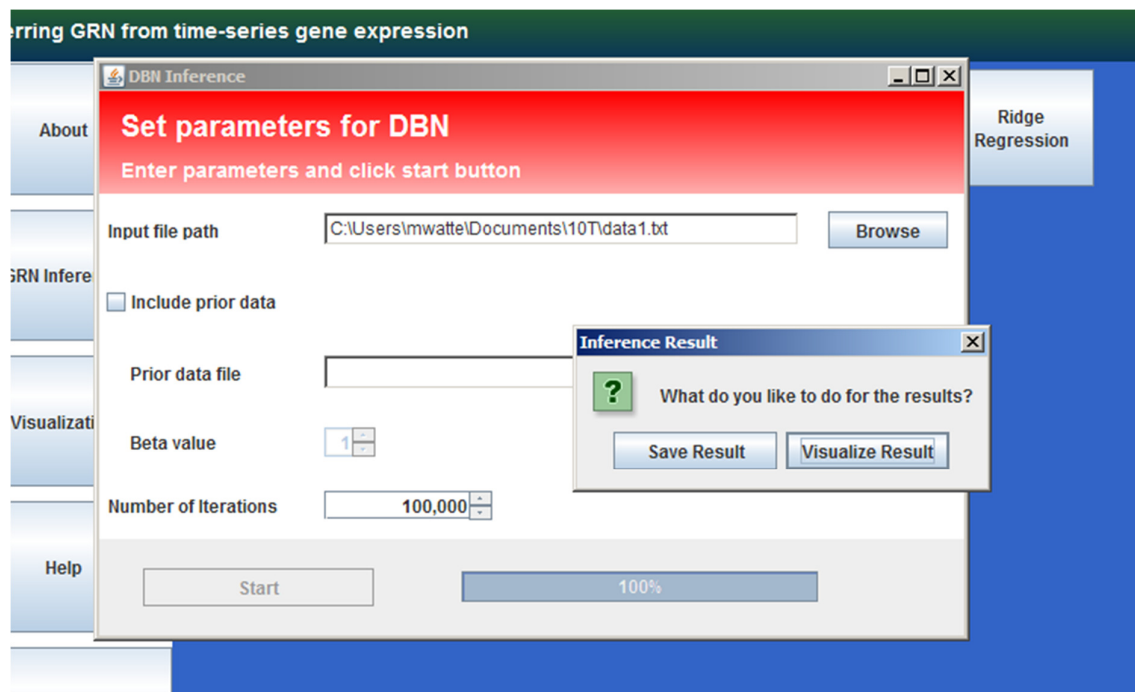
b. change number of iterations if required

c. then click start

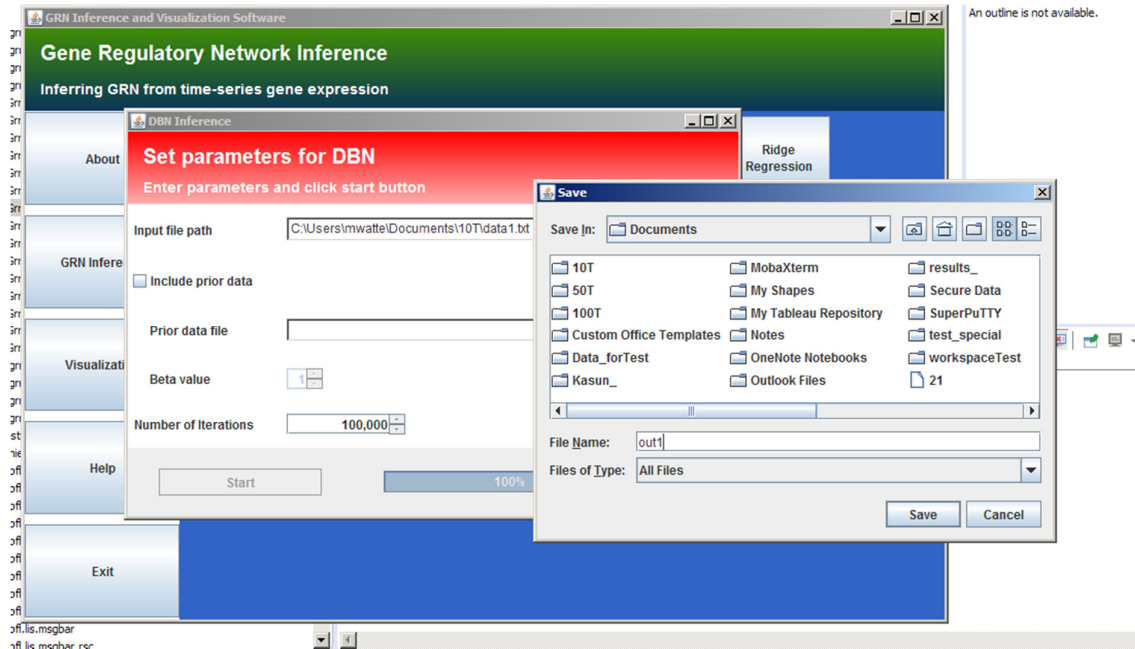
d. progress bar will update during dbn execution



e. after dbn execution complete , it will pop up save window

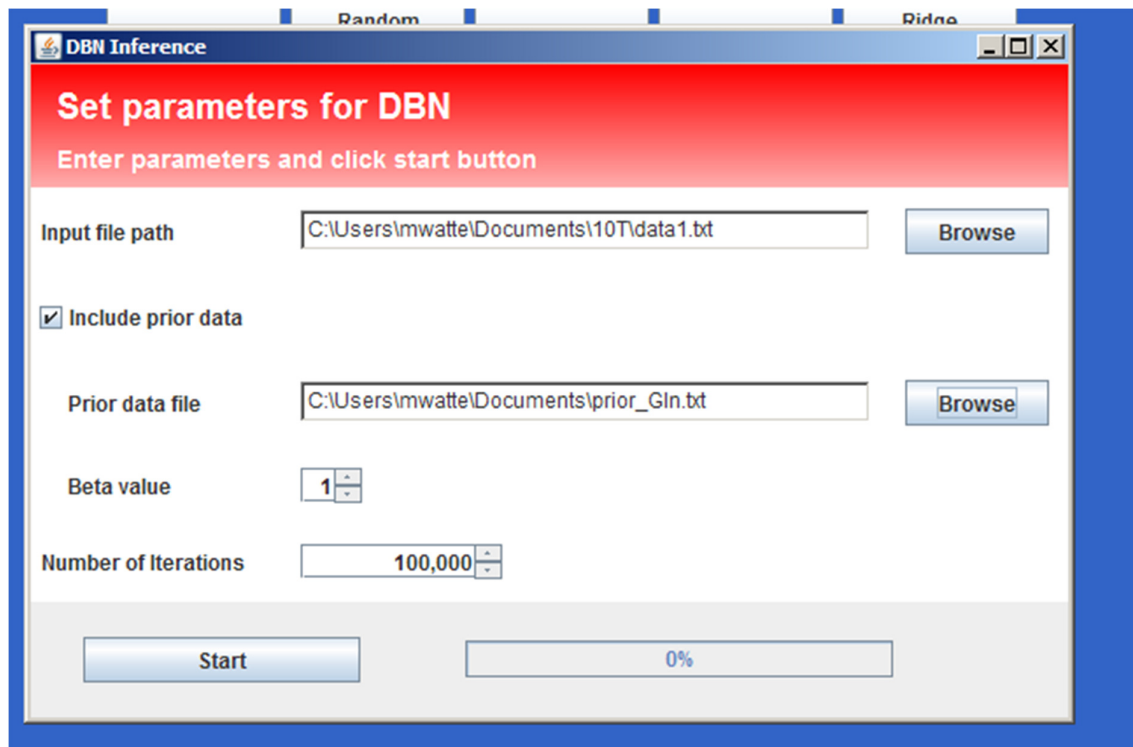


- f. then select save result
- g. do not select visualize part, this will not work as I modified GrnDbn.java class to evaluate the results (TP,FP,TN,precision,...etc)
- h. give file name to save the results



12. run dbn with prior data

- A. CLICK "include prior data" option
- B. browse and select prior data file
- C. update beta parameter
- D. then click start button



13. input file format

- a. should be continuous values
- b. columns \rightarrow genes
- c. rows \rightarrow time points
- d. `DbnDiscretization` java class will convert this data to discrete format
- e. sample input file given below 9 genes and 30 time points

```
1 -0.5672424 0.0443159 -0.0667056 -2.1682727 1.3021522 -4.5602288 2.6346165 -0.1063942 -1.2251728
2 -3.0707017 -1.0330354 0.7238890 0.4004509 -0.5532288 -0.0603258 -2.7267363 1.3537693 -0.9511030
3 2.4641521 0.9158940 -1.4309775 2.0679663 -1.1237100 -1.0187650 -2.1334996 -1.9053139 -1.1975429
4 -4.1221990 2.6836468 0.2338569 -3.1737985 0.8716300 2.0711220 0.5544709 1.5825803 -1.5934476
5 2.3907478 -3.4760457 0.6845930 -2.4607690 1.0511559 -1.7826440 -1.5324567 -0.5564544 -0.8523886
6 -0.8132309 -0.9866063 0.2379037 2.3622709 -0.8234727 -3.5020272 -3.8350771 0.0278326 -1.4918603
7 -3.6953532 -1.1972161 -0.4401855 0.8727511 -0.3479502 0.9177234 -0.2155364 0.7500722 0.5369268
8 -0.4259475 -2.3184769 1.1721206 1.7450274 -2.3901773 1.3133557 0.8877193 -1.1045262 0.4982408
9 1.0276577 1.3422711 0.7127813 2.0610216 -2.1299184 1.7294118 1.4187370 2.1651168 0.9846757
10 1.9192079 6.0978429 -0.8008835 -2.1331440 -1.6013061 2.4817550 5.2407369 1.5061834 -1.1527381
11 0.0161787 4.3751432 -0.8463808 -2.9988960 0.8359283 0.0605703 3.4781781 -1.0798726 -2.2063347
12 -0.0830125 -1.4659419 0.1485971 -2.9237699 0.9457745 1.7309025 -1.8767456 0.8855043 1.4756738
13 2.2161360 -0.0411106 1.3193502 -1.7613816 0.5087442 -5.9869987 -1.9482915 -3.3791607 0.2908154
14 -4.4110228 -2.4248483 0.7373836 0.0681285 -0.1978446 -4.0572009 -0.4678214 3.0642697 2.0617110
15 -2.2885760 -0.8619143 0.0573654 3.1096609 -0.4918259 -1.3554726 1.1378507 4.7888667 -0.2169312
16 -1.7947064 0.6517405 -1.0094309 0.0363368 1.3145738 4.7682836 3.2276249 0.9093841 -0.0921207
17 4.2365975 -0.7051272 0.6394201 -1.2345029 -2.0227281 2.8373661 1.0488386 -2.7541741 1.6323212
18 1.7463332 3.6426224 0.2636844 0.1169970 -0.6376573 -0.9413178 1.7725475 -2.0471319 0.1202596
19 -2.3070543 -0.0420693 -1.0918219 -3.4005626 -1.4846904 2.9163988 3.5879684 1.9407111 1.0453282
20 -0.6641730 1.4046648 0.2698520 -0.1546602 -0.0073221 -0.6816370 -0.5768352 -0.0146066 -1.5848994
21 -1.1832579 0.2106409 -0.2431342 -1.6400260 0.1406208 -0.1085844 -0.8340076 0.3171433 -1.1762038
22 -0.5100360 -0.0025122 0.5891647 0.5798675 -0.2935609 -0.8572766 -2.3497948 -1.4603788 -1.1887057
23 -0.6558490 -1.7150373 -0.9656329 -0.8086568 0.4576847 1.5333357 0.9722149 1.5087997 -1.2702814
24 1.4035432 -0.9726512 -0.7298489 1.1485483 1.5136924 1.0201329 -0.7087743 -1.4173570 -1.0820557
25 3.3601610 -0.2407736 0.1743894 -2.9694907 0.7166796 -0.6808018 0.0831158 -1.9100277 1.6551398
26 0.2832065 0.9056395 -0.5173718 -1.9130396 -1.2717848 -2.2974668 -0.6830111 1.6119390 2.2241180
27 -2.7356658 2.5754572 -0.9736915 -1.7056749 -1.1930616 -3.6519871 0.6154772 0.6470072 0.5181864
28 -5.1428264 -0.9154725 -0.6042712 -3.1067809 1.4501933 -0.1760493 1.4470850 3.8068568 0.9710317
29 -0.4944317 -2.5121746 -0.5429612 -0.4486102 -0.4969052 -1.8642081 -3.1457004 -2.5159128 -1.3383323
30 -2.4888590 -3.5627396 -0.4465464 2.3320895 -0.3367283 -0.6751720 -2.2557277 0.6735929 0.9302445
31
```

14. ouput file format

below results will write to the file

TP

FP

TN

FN

precision

recall

fmeasure

final output network

sample output file is given below

```

File Edit Search View Encoding Language Settings Macro Run Plugins Win
21 x 22 x 23 x 24 x 25 x data5.txt
1 TP = 4
2 FP = 11
3 TN = 53
4 FN = 13
5 Precision = 0.26666666666666666
6 Recall = 0.23529411764705882
7 F-measure = 0.25
8
9
10 Print output network as a matrix
11 0 → 0 → 0 → 0 → 0 → 0 → 0 → 0 → 0 →
12 0 → 0 → 1 → 0 → 0 → 0 → 1 → 1 → 0 →
13 0 → 0 → 0 → 0 → 0 → 0 → 0 → 0 → 0 →
14 0 → 0 → 1 → 0 → 1 → 1 → 0 → 0 → 0 →
15 0 → 0 → 0 → 0 → 0 → 0 → 0 → 0 → 0 →
16 0 → 0 → 0 → 0 → 0 → 0 → 0 → 0 → 0 →
17 0 → 0 → 0 → 1 → 1 → 0 → 0 → 0 → 0 →
18 1 → 1 → 1 → 0 → 0 → 0 → 0 → 0 → 1 →
19 0 → 0 → 0 → 1 → 1 → 1 → 0 → 0 → 0 →
20

```

special note – execution process for synthetic data

1. if you have 20 time series -> 20 input files
2. so we have to execute dbn inference 20 times
3. this will generate 20 output files
4. it means 20 precision values, 20 recall values ,...etc
5. add all of them manually and take average and std

exeution process for real data

1. real data normally have only one data set
2. so , we don't need to run dbn for 20 times because we using same data set
3. we can run 5 or 10 times and then take average of the results.

gold standard network

1. 9 gene Spellman network is hard coded in the `GrnDbn.java` class
2. go the public static void `evaluateNetwork(int outputNetwork [][], int genes, String resultSavePath)` method
3. in here you can see , how gold standard network is hard coded is done
4. If you want to put some other gold standard network , please update `actualNetwork [][]` matrix accordingly.