Practical Bioinformatics

Mark Voorhies

6/18/2010

Review

- `import` vs. `reload`
- indexing multi-dimensional arrays
- defining `class` methods
- handling `exceptions`

Adding an output function to our class

```
def write(self, filename):
    fp = open(filename,"w")
    fp.write(self.header)
    for i in range(len(self.genes)):
        fp.write(self.genes[i])
        fp.write("\t"+self.annotations[i])
        for j in self.ratios[i]:
            if(j == None):
                fp.write("\t"+"")
            else:
                fp.write("\t%f" % j)
        fp.write("\n")
```

Distance functions

1. Write a function to calculate the uncentered Pearson distance between two gene profiles

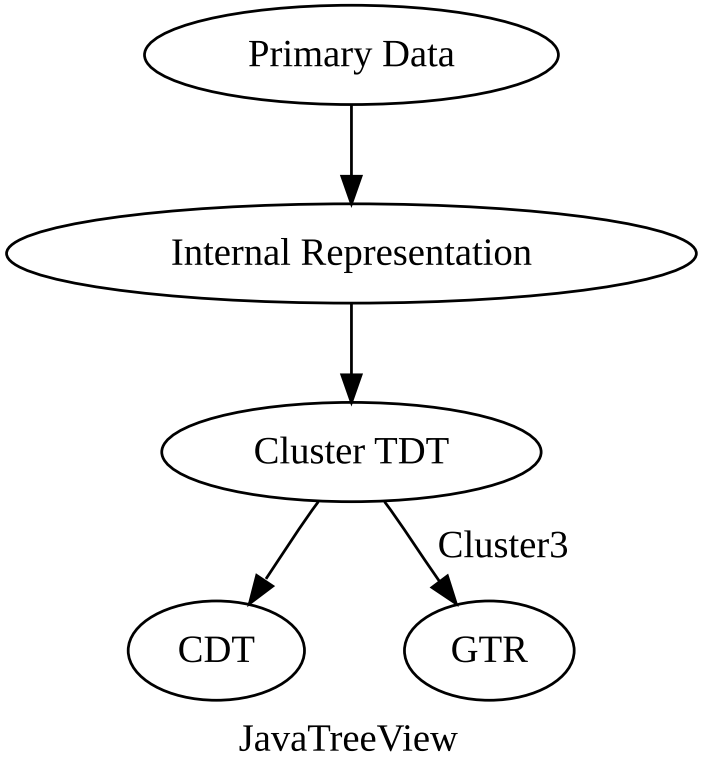
$$d(x,y) = 1 - \frac{\sum_i^N (x_i - x_{offset})(y_i - y_{offset})}{\sqrt{\sum_i^N (x_i - x_{offset})^2}\sqrt{\sum_i^N (y_i - y_{offset})^2}} \quad (1)$$

2. Amend the function to calculate the centered Pearson (or another distance metric from the Cluster3 manual)
3. Write a function to calculate all pairwise distances for the yeast expression profiles for a particular distance function.

Special cases for Pearson in Cluster3

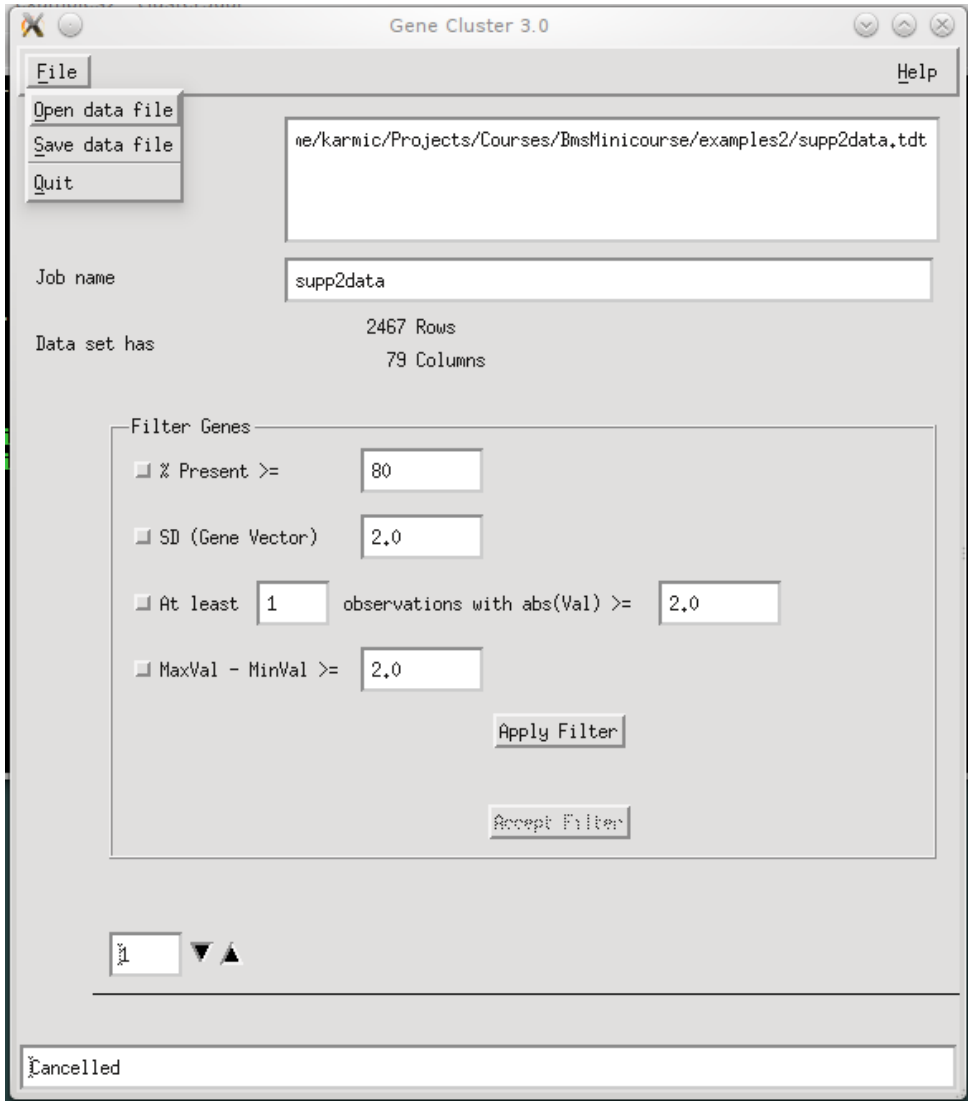
```
int flag = 0;
/* flag will remain zero if no nonzero combinations of mask1 and mask2 are
 * found.
 */
int i;
for (i = 0; i < n; i++)
{ if (mask1[index1][i] && mask2[index2][i])
  { double term1 = data1[index1][i];
    double term2 = data2[index2][i];
    double w = weight[i];
    result += w*term1*term2;
    denom1 += w*term1*term1;
    denom2 += w*term2*term2;
    flag = 1;
  }
}
if (!flag) return 0.;
if (denom1==0.) return 1.;
if (denom2==0.) return 1.;
result = result / sqrt(denom1*denom2);
result = 1. - result;
return result;
```

Clustering protocol

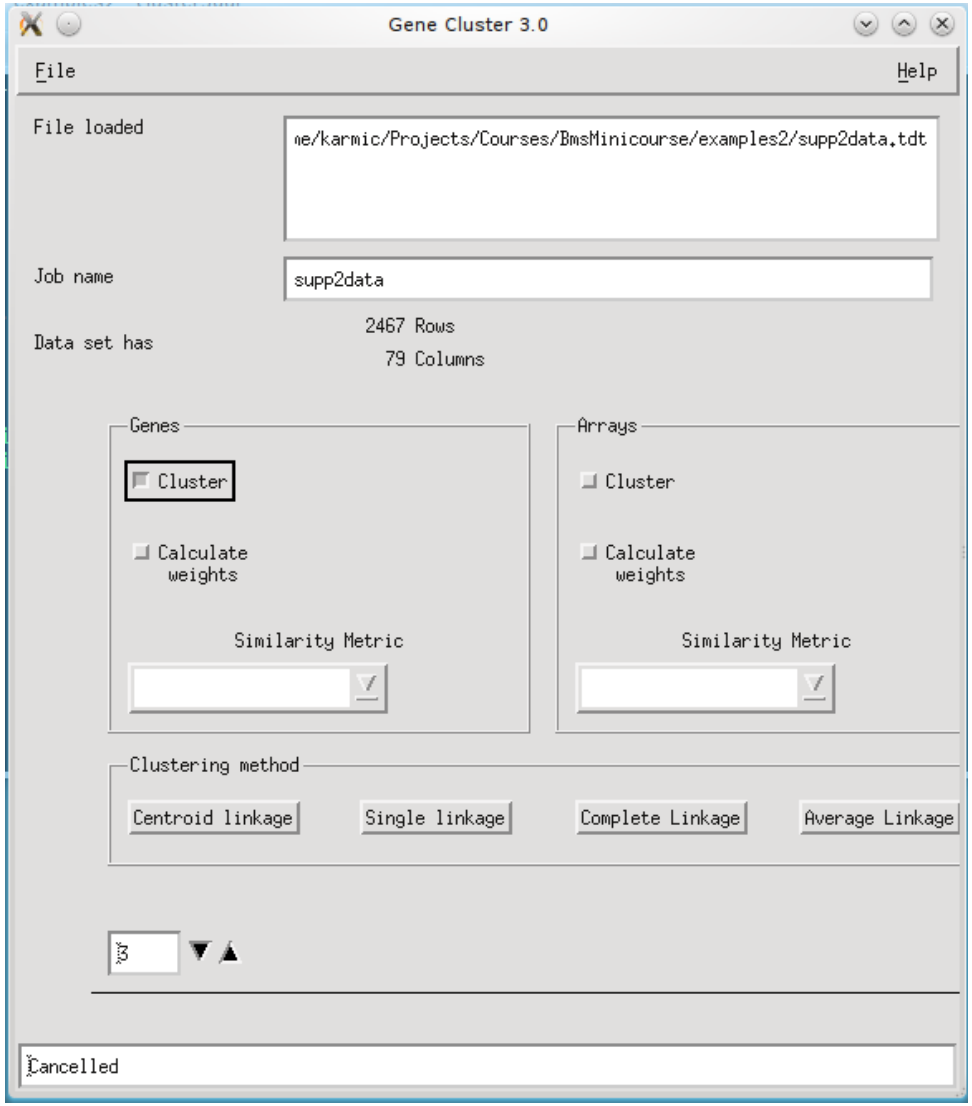


Using the Cluster3 GUI

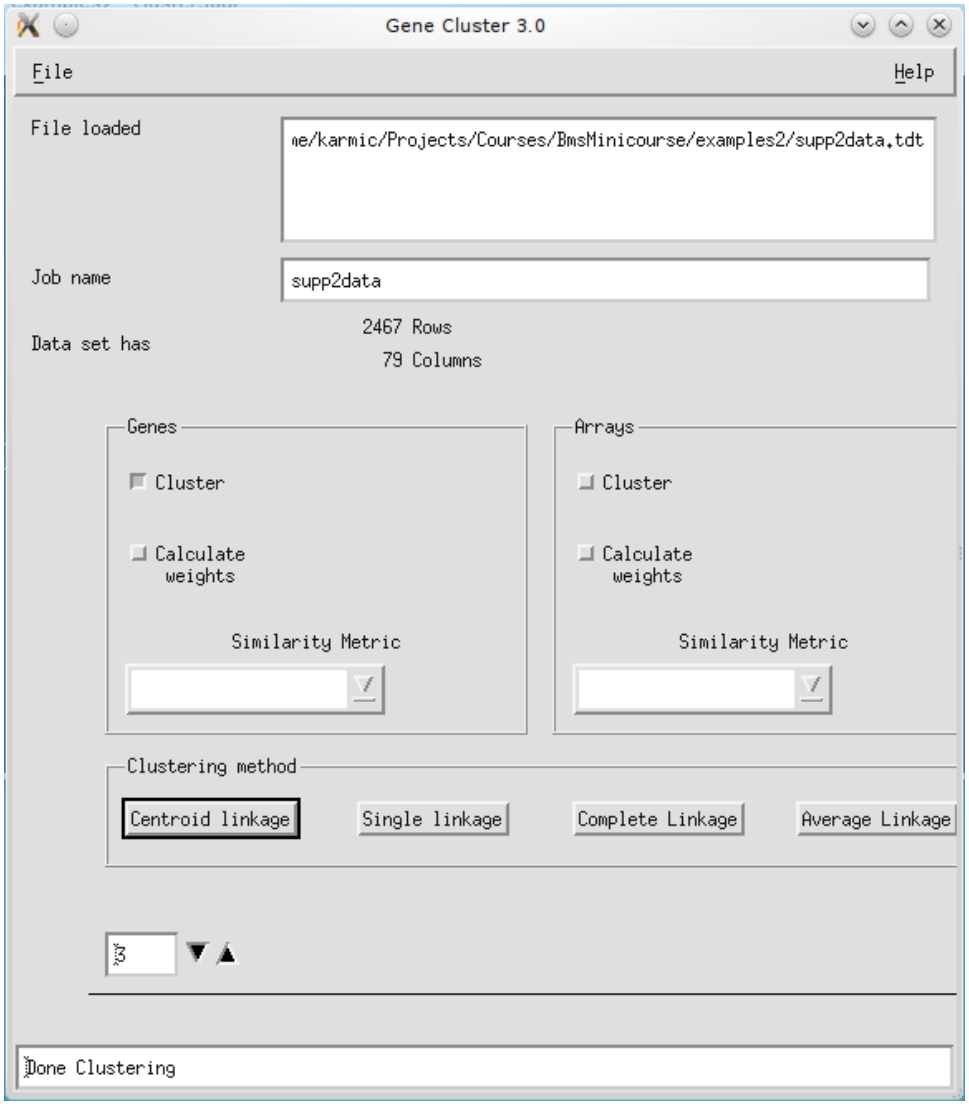
Load your data



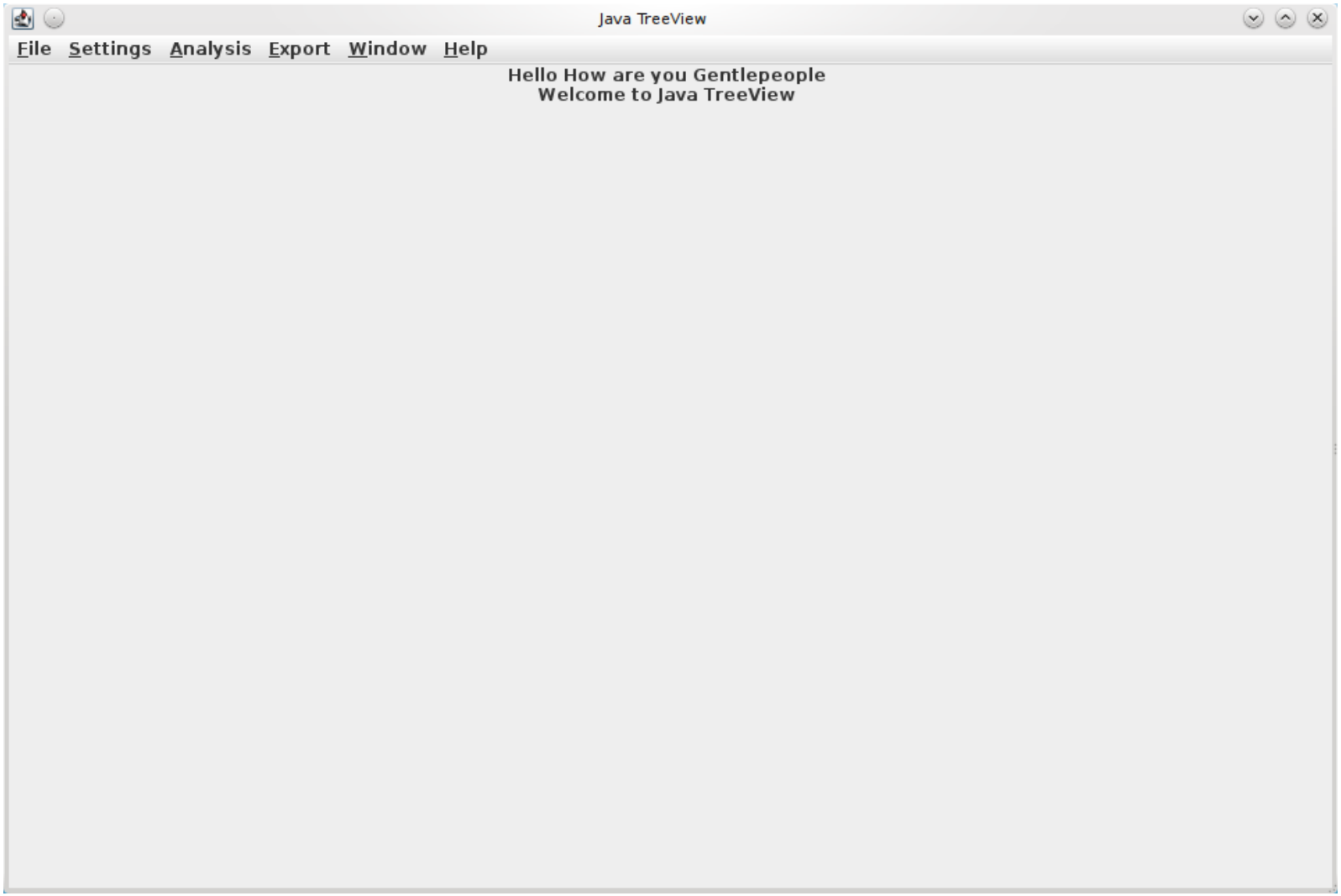
Choose distance function



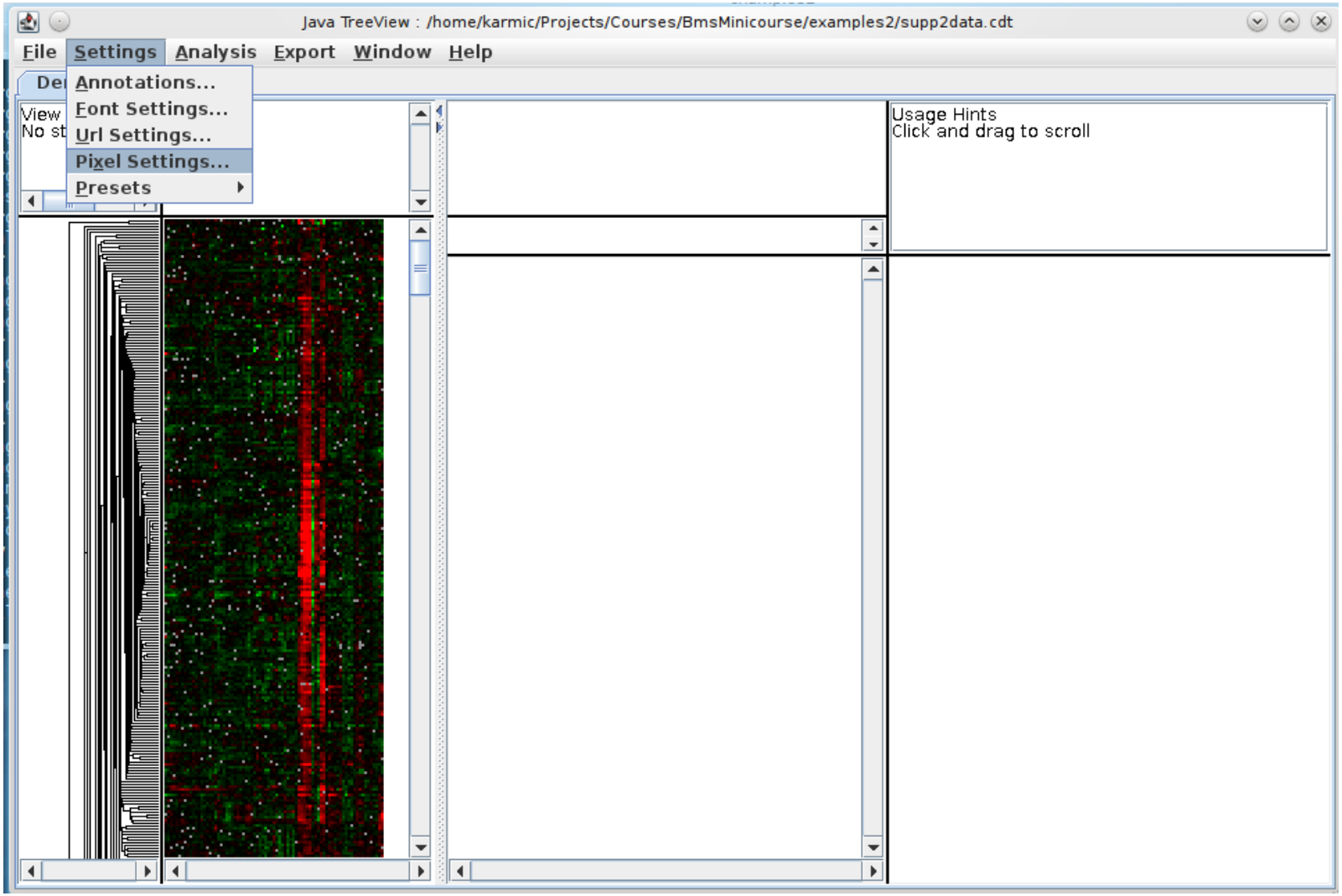
Choose linking method



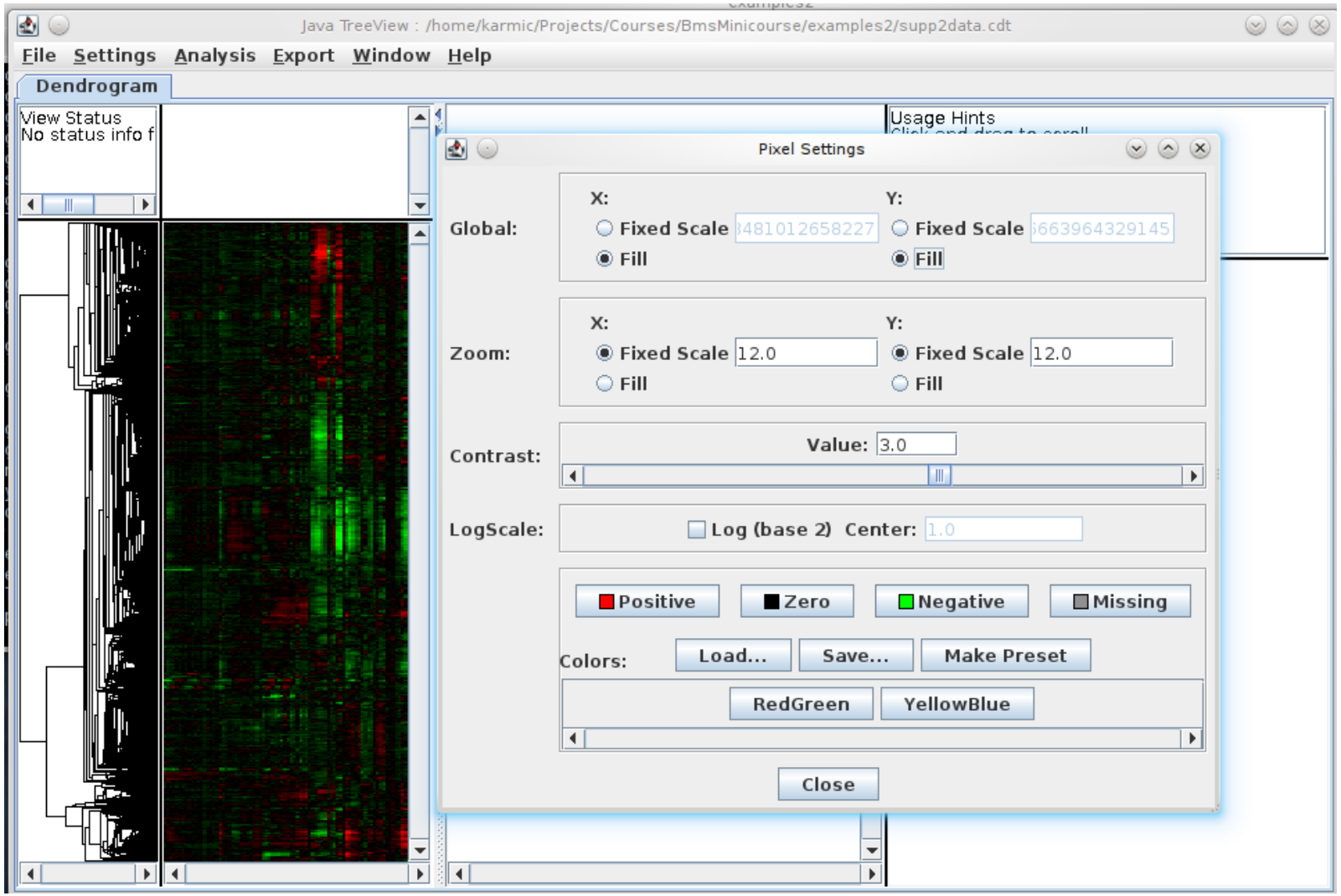
Using JavaTreeView



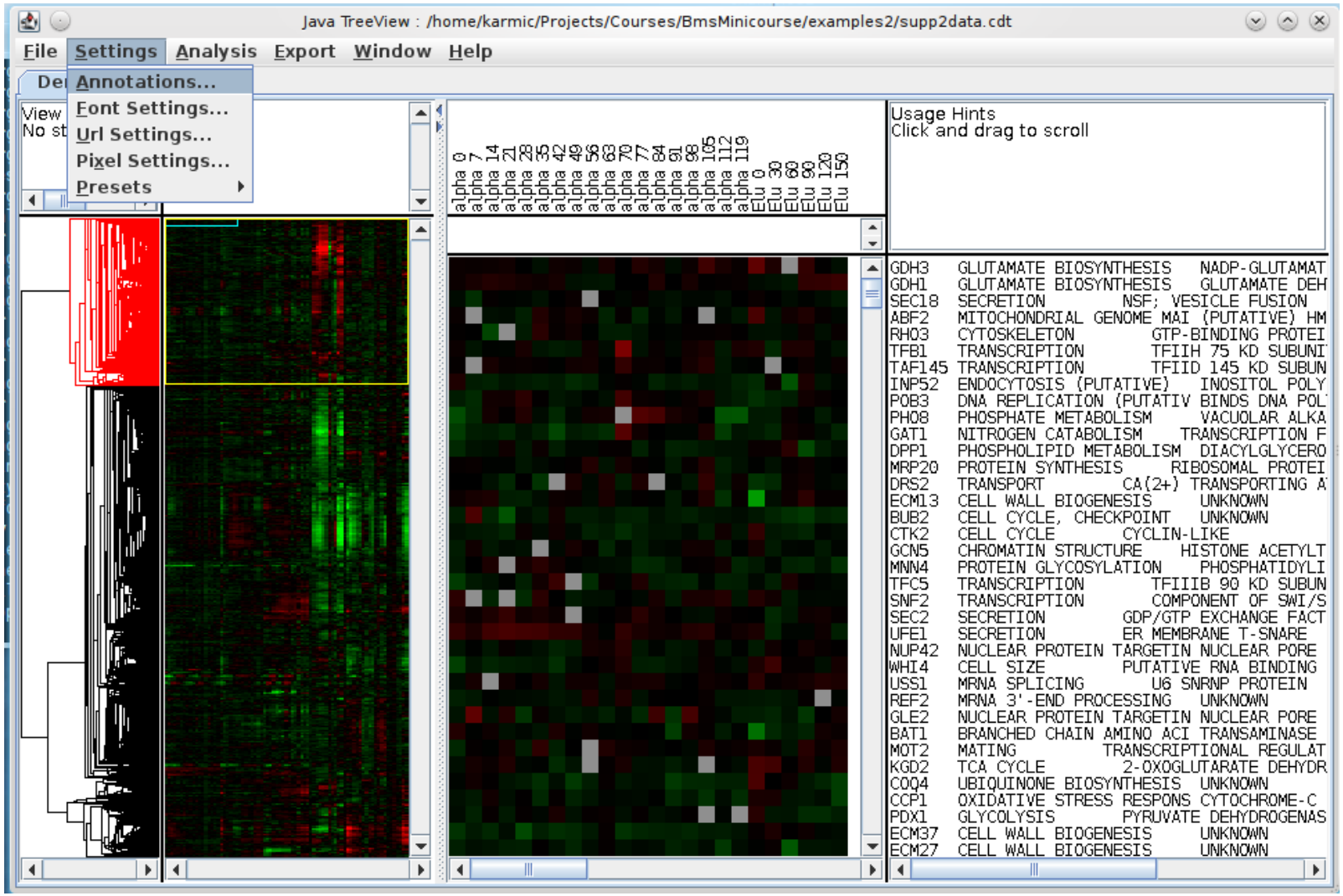
Adjust pixel settings for global view



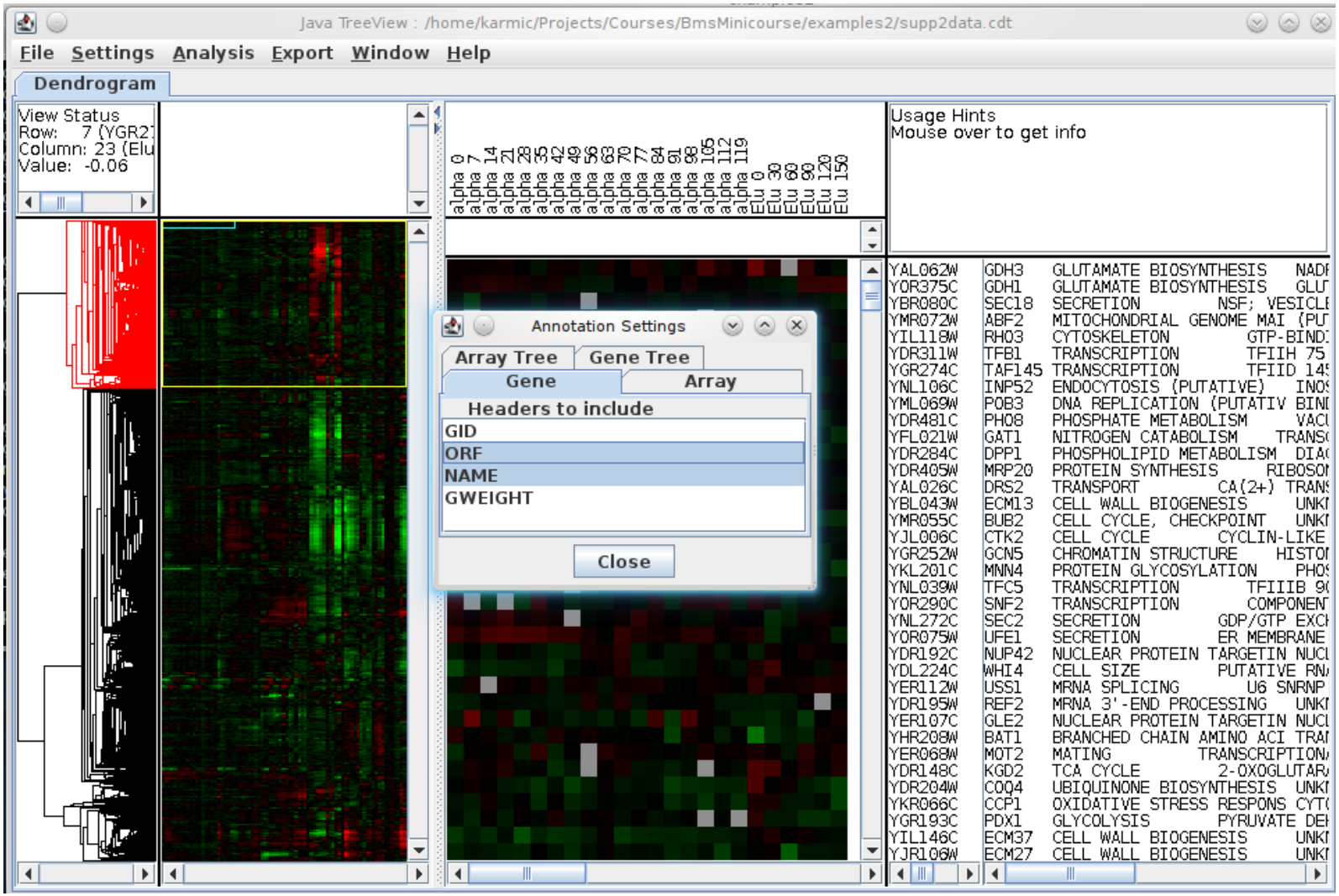
Adjust pixel settings for global view



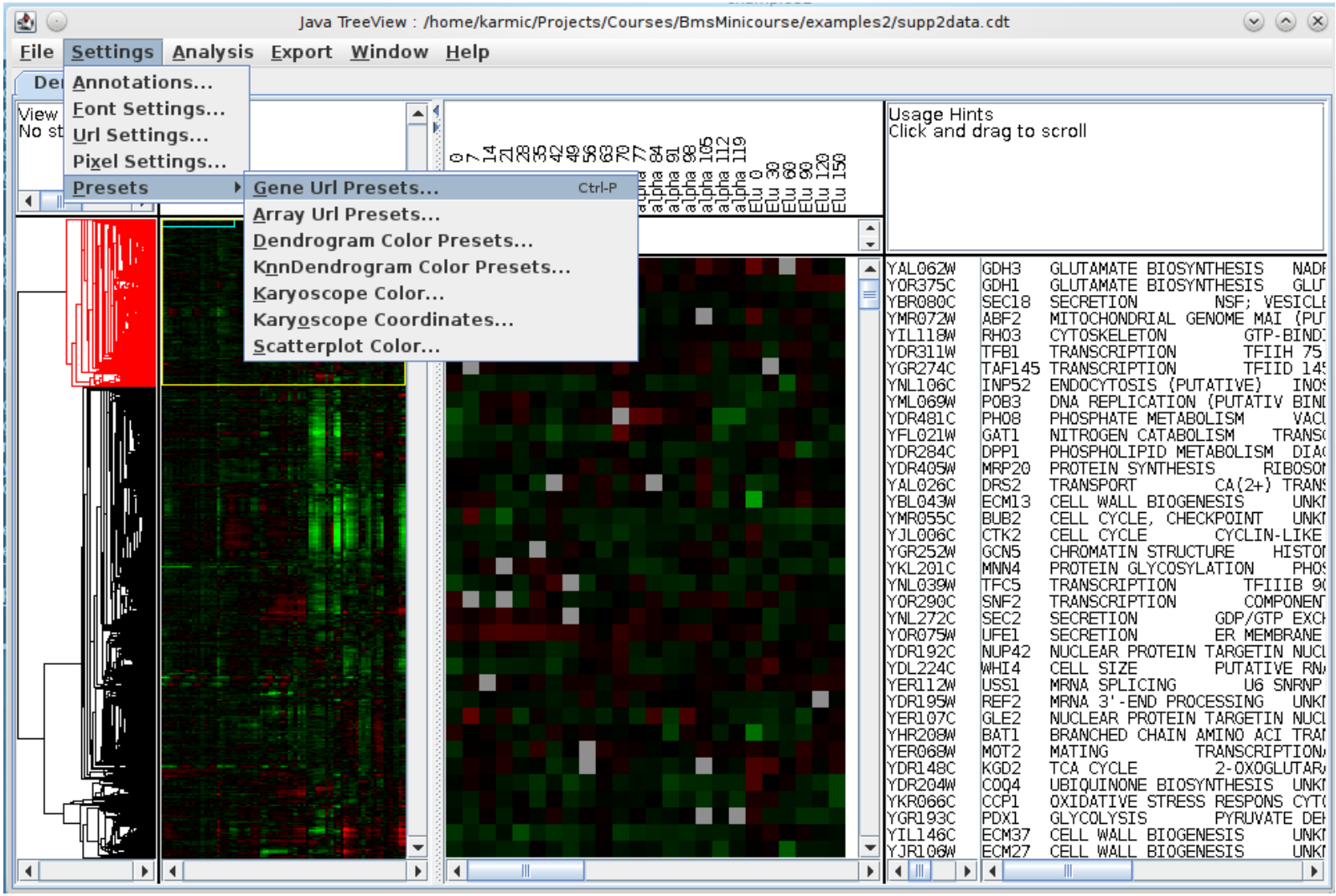
Select annotation columns



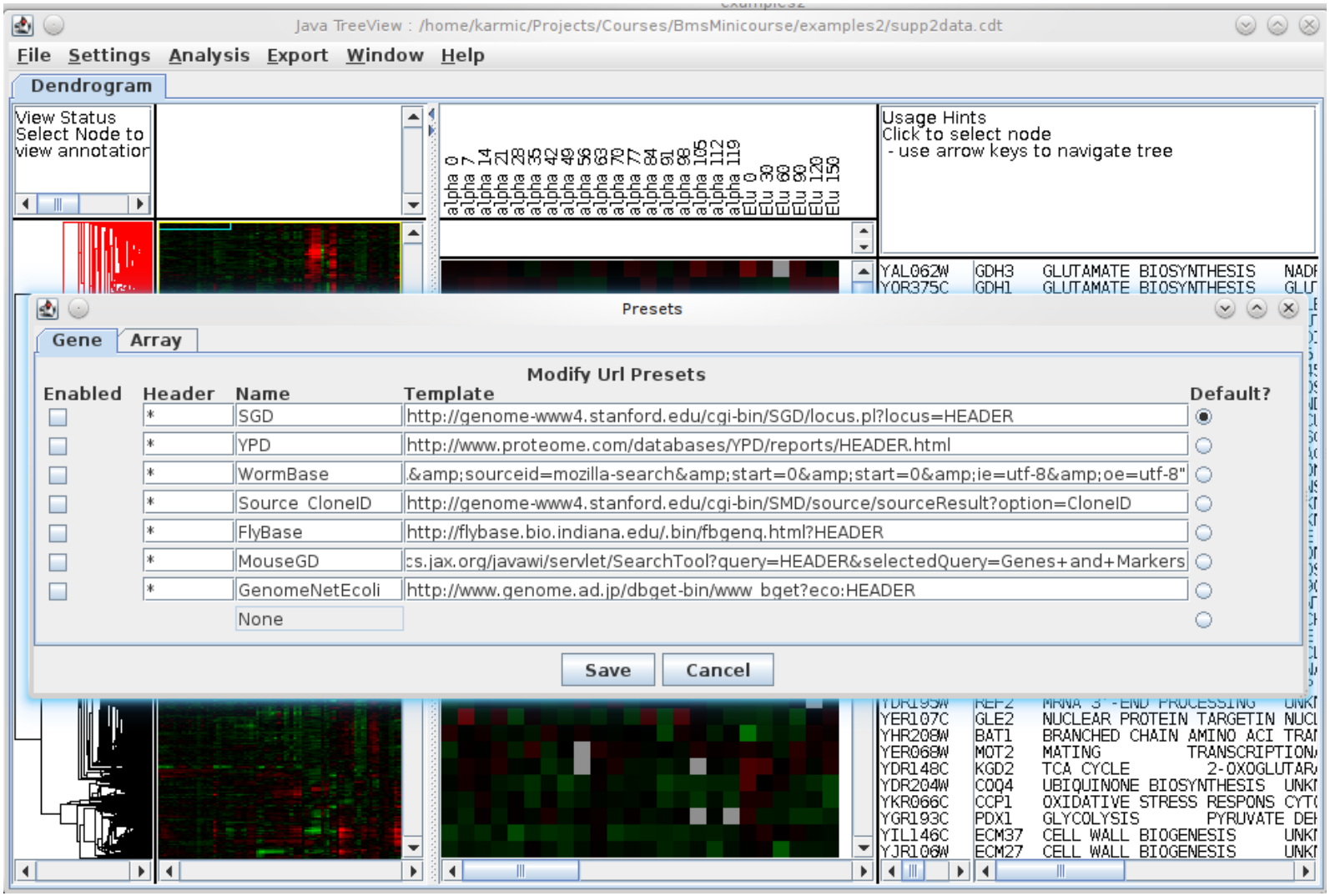
Select annotation columns



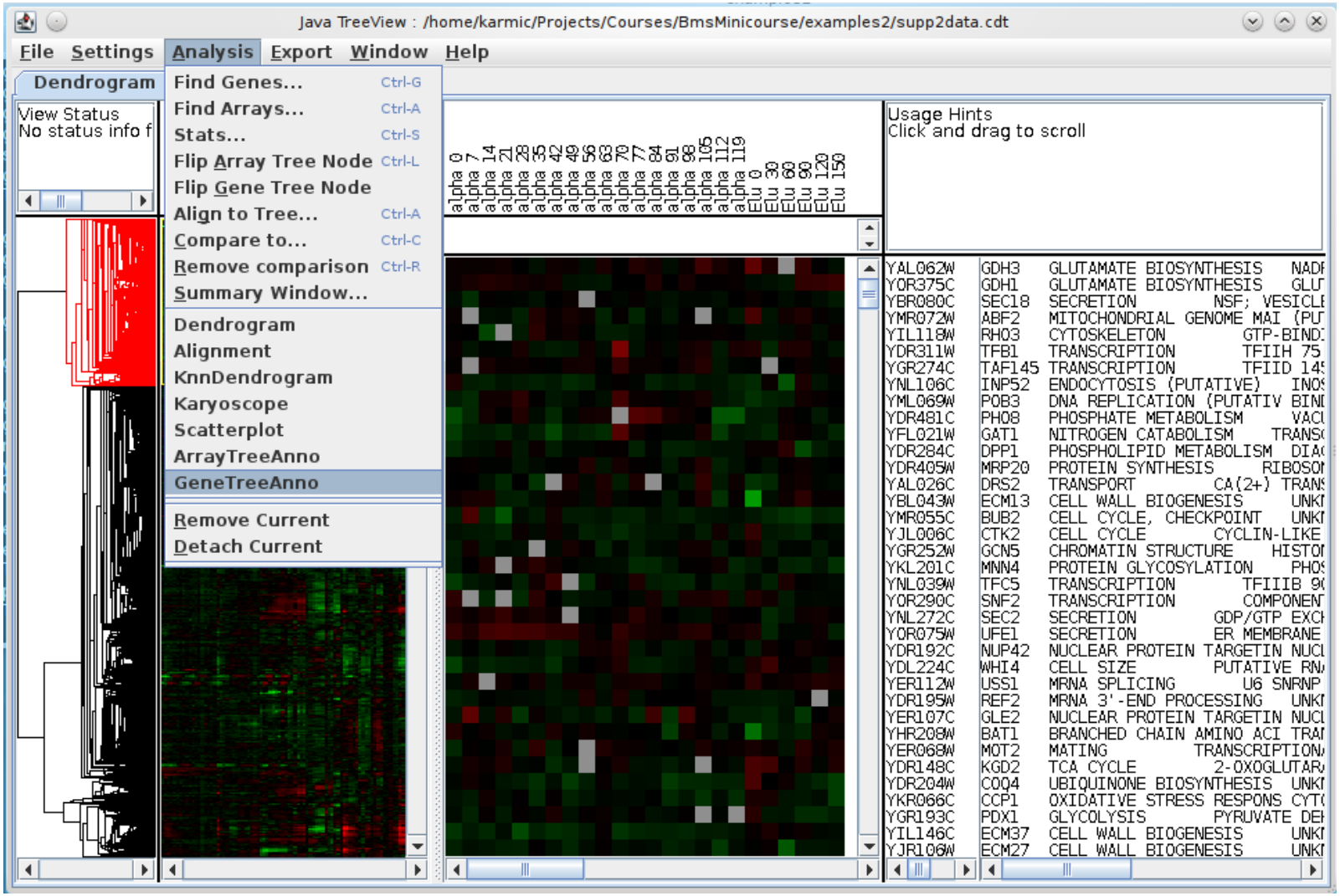
Select URL for gene annotations



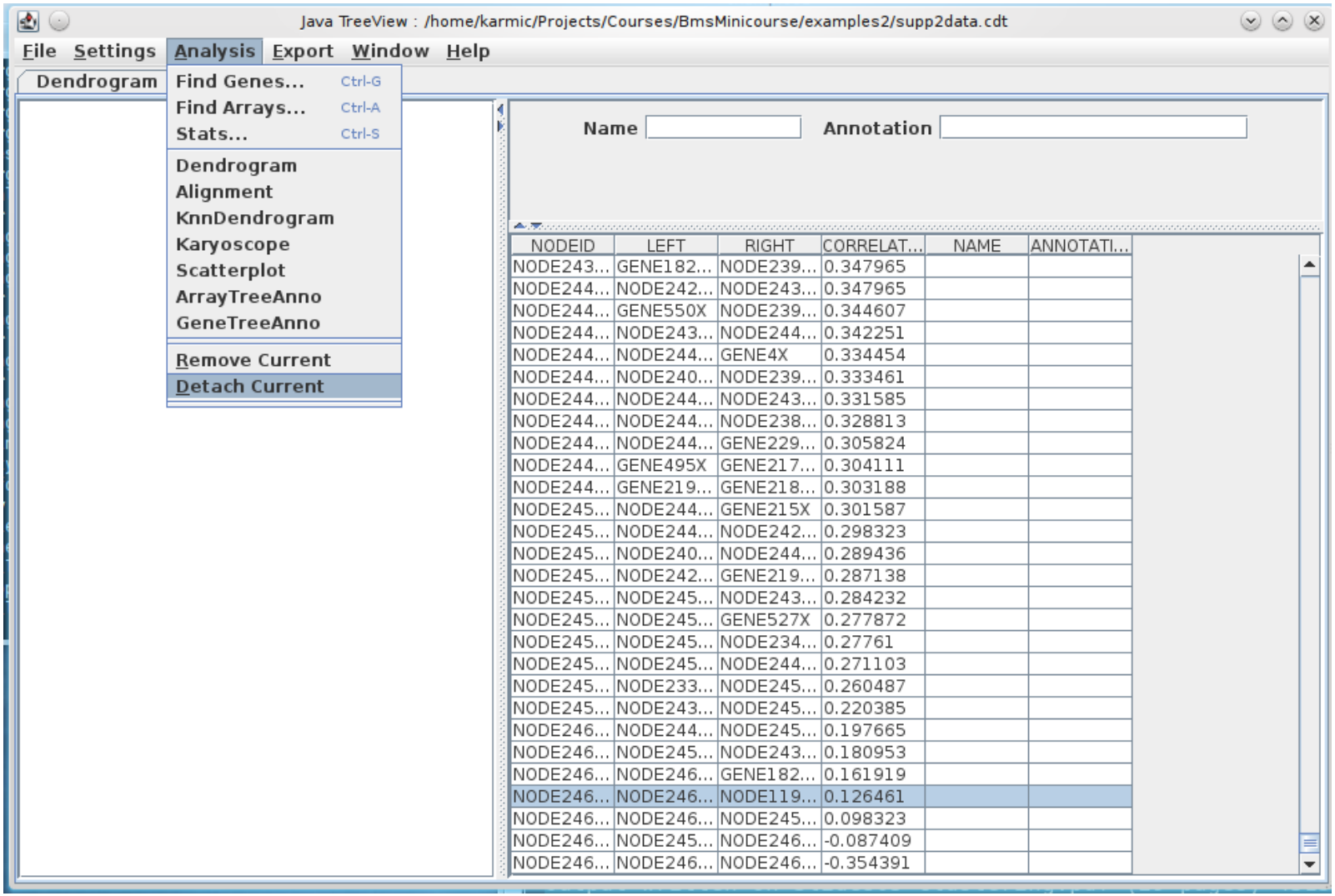
Select URL for gene annotations



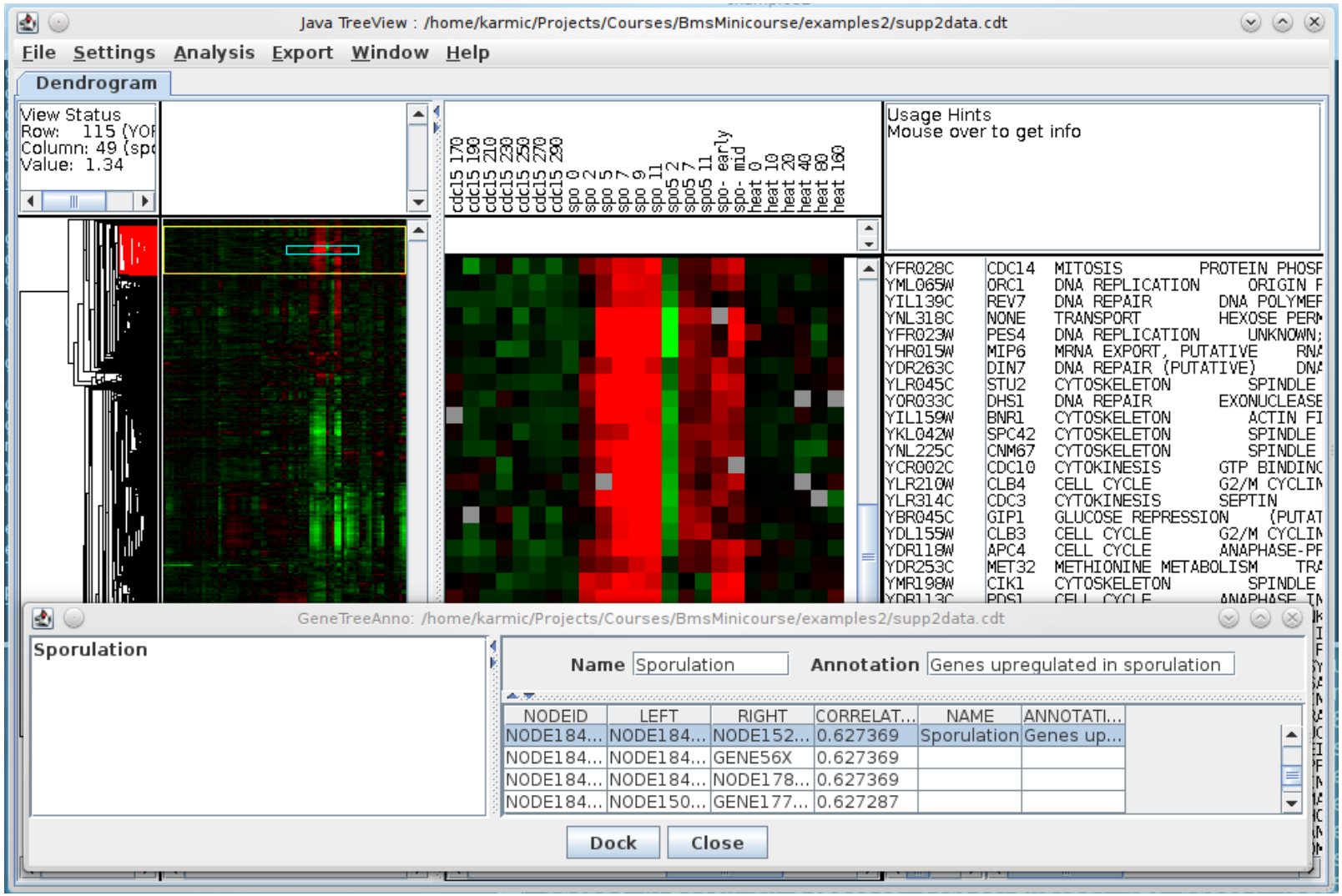
Activate and detach annotation window



Activate and detach annotation window



Activate and detach annotation window



Scripting Cluster

- Running Cluster3 from the command line
  - /Applications/Cluster.app/Contents/MacOS/Cluster
  - /Program Files/Stanford University/Cluster3/Cluster.exe
- Command-line programs are like functions
- "man program" is like "help(function)"
- Use the `subprocess` module to run command-line programs from within Python.

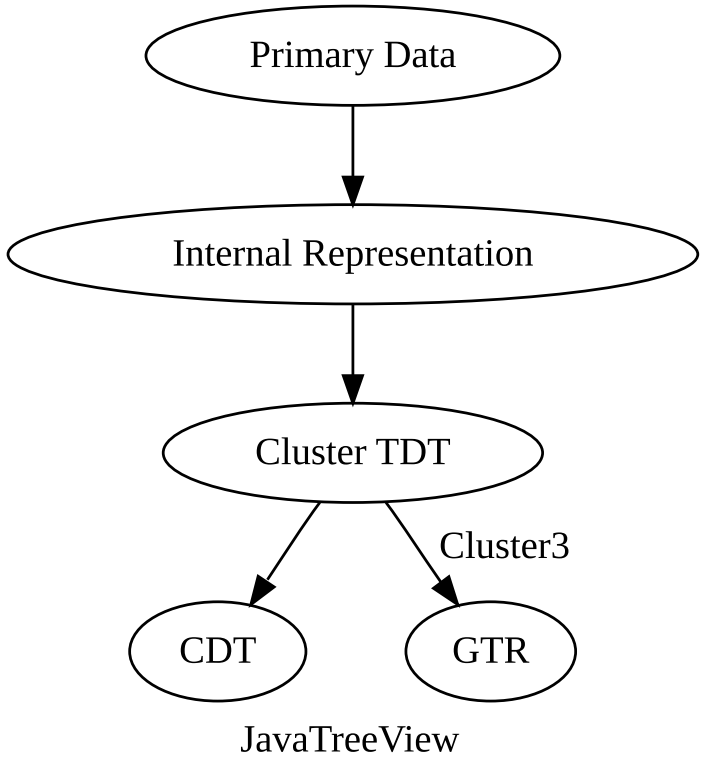
Programs as functions

USAGE: cluster [options]

-f filename	File loading
-u jobname	Allows you to specify a different name for the output files (default is derived from the input file name)
-g [0..8]	Specifies the distance measure for gene clustering 0: No gene clustering 1: Uncentered correlation 2: Pearson correlation 3: Uncentered correlation, absolute value 4: Pearson correlation, absolute value 5: Spearman's rank correlation 6: Kendall's tau 7: Euclidean distance 8: City-block distance (default: 0)
-m [msca]	Specifies which hierarchical clustering method to use m: Pairwise complete-linkage s: Pairwise single-linkage c: Pairwise centroid-linkage a: Pairwise average-linkage (default: m)

Scripting the Protocol

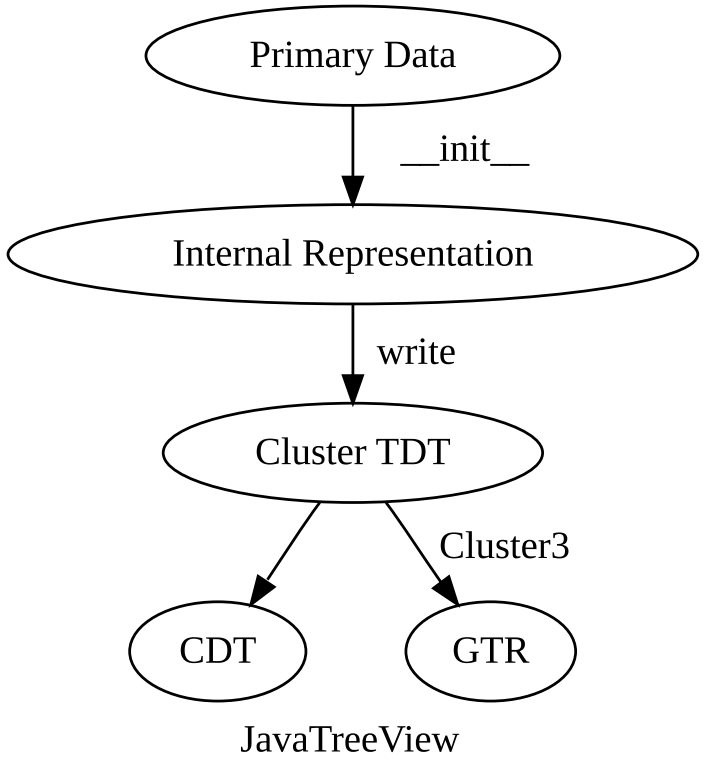
```
from subprocess import check_call
check_call(
    # Which program to run
    ("cluster",
     # Input file
     "-f","supp2data.tdt",
     # Output prefix
     "-u","supp2data.Uncentered.Complete",
     # Clustering method: complete linkage
     "-m","m",
     # Distance function: uncentered Pearson
     "-g","1"))
```



Shuffling Genes to Remove Structure

```
from TdtRatios import TdtRatios
data = TdtRatios("supp2data.tdt")
data.shuffleGenes()
data.write("shuffled.tdt")

from subprocess import check_call
check_call(
    ("cluster",
     "-f","shuffled.tdt",
     "-u","shuffled.Uncentered.Complete",
     "-m","m",
     "-g","1"))
```



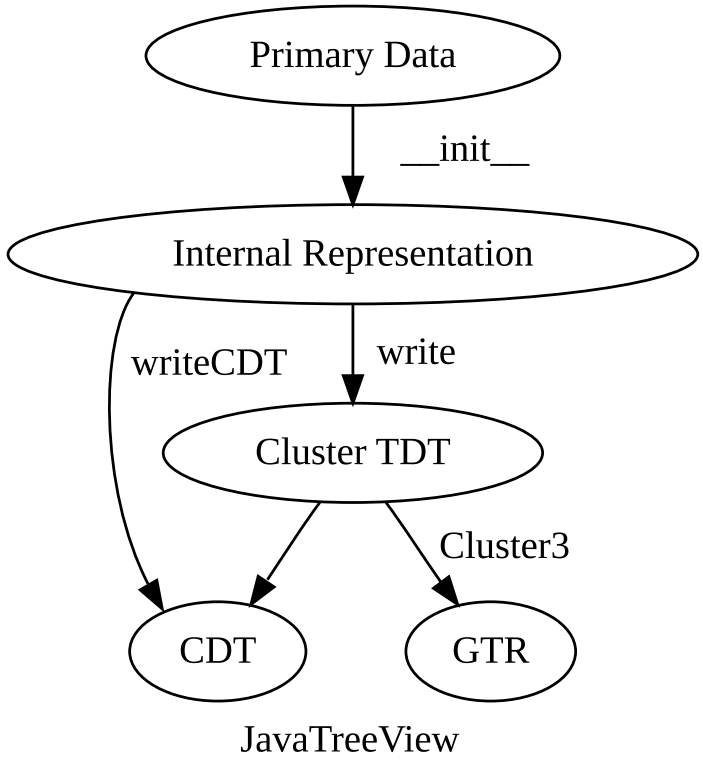
Adding a shuffling function to our class

```
def shuffleGenes(self, seed = None):
    """Shuffle expression matrix by row."""
    import random
    if(seed != None):
        random.seed(seed)
    indices = range(len(self.genes))
    random.shuffle(indices)
    genes = [self.genes[i] for i in indices]
    self.genes = genes
    annotations = [self.annotations[i] for i in indices]
    self.genes = genes
    ratios = [self.ratios[i] for i in indices]
    self.ratios = ratios
```

Checking that the Shuffling Worked

```
from TdtRatios import TdtRatios
data = TdtRatios("supp2data.tdt")
data.shuffleGenes()
data.write("shuffled.tdt")
data.writeCDT("shuffled.cdt")

from subprocess import check_call
check_call(
    ("cluster",
     "-f","shuffled.tdt",
     "-u","shuffled.Uncentered.Complete",
     "-m","m",
     "-g","1"))
```



Generating CDT output

```
def write(self, filename):
    fp = open(filename,"w")
    fp.write(self.header)
    for i in range(len(self.genes)):
        fp.write(self.genes[i])
        fp.write("\t"+self.annotations[i])
        for j in self.ratios[i]:
            if(j == None):
                fp.write("\t"+"")
            else:
                fp.write("\t%f" % j)
        fp.write("\n")
```

```
def writeCDT(self, filename):
    """Write CDT file for JavaTreeView."""
    fp = open(filename,"w")
    cols = self.header.rstrip().split("\t")
    fp.write("\t".join(["GID"]+cols[:2]+
                       ["GWEIGHT"]+cols[2:])
             +"\n")
    fp.write("\t".join(["EWEIGHT"]+[""]*3+
                       ["1.0"]*
                       len(self.ratios))
             +"\n")
    for i in range(len(self.genes)):
        fp.write("GENE%4dX" % (i+1))
        fp.write("\t"+self.genes[i])
        fp.write("\t"+self.annotations[i])
        fp.write("\t1.0")
        for j in self.ratios[i]:
            if(j == None):
                fp.write("\t"+"")
            else:
                fp.write("\t%f" % j)
        fp.write("\n")
```

Clustering exercises

1. Cluster supp2data.tdt and explore the results in JavaTreeView. Can you identify the clusters from figure 2 of the Eisen paper. Click on gene names to open the corresponding SGD annotations in your web browser. Are the current annotations consistent with those in supp2data.tdt? Are they consistent with the clustering pattern?
2. Write you pairwise distance matrix to a CDT file (in this case, the rows and columns are *both* genes) and visualize it in JavaTreeView.
3. Add functions to TdtRatios to reproduce the shuffling controls in figure 3 of the Eisen paper (removing correlations among genes and/or arrays).
4. Modify the clustering protocol script to run Cluster3 multiple times on the same input, varying distance metric and/or clustering method. Be sure to give the output files distinct names.