Practical Bioinformatics

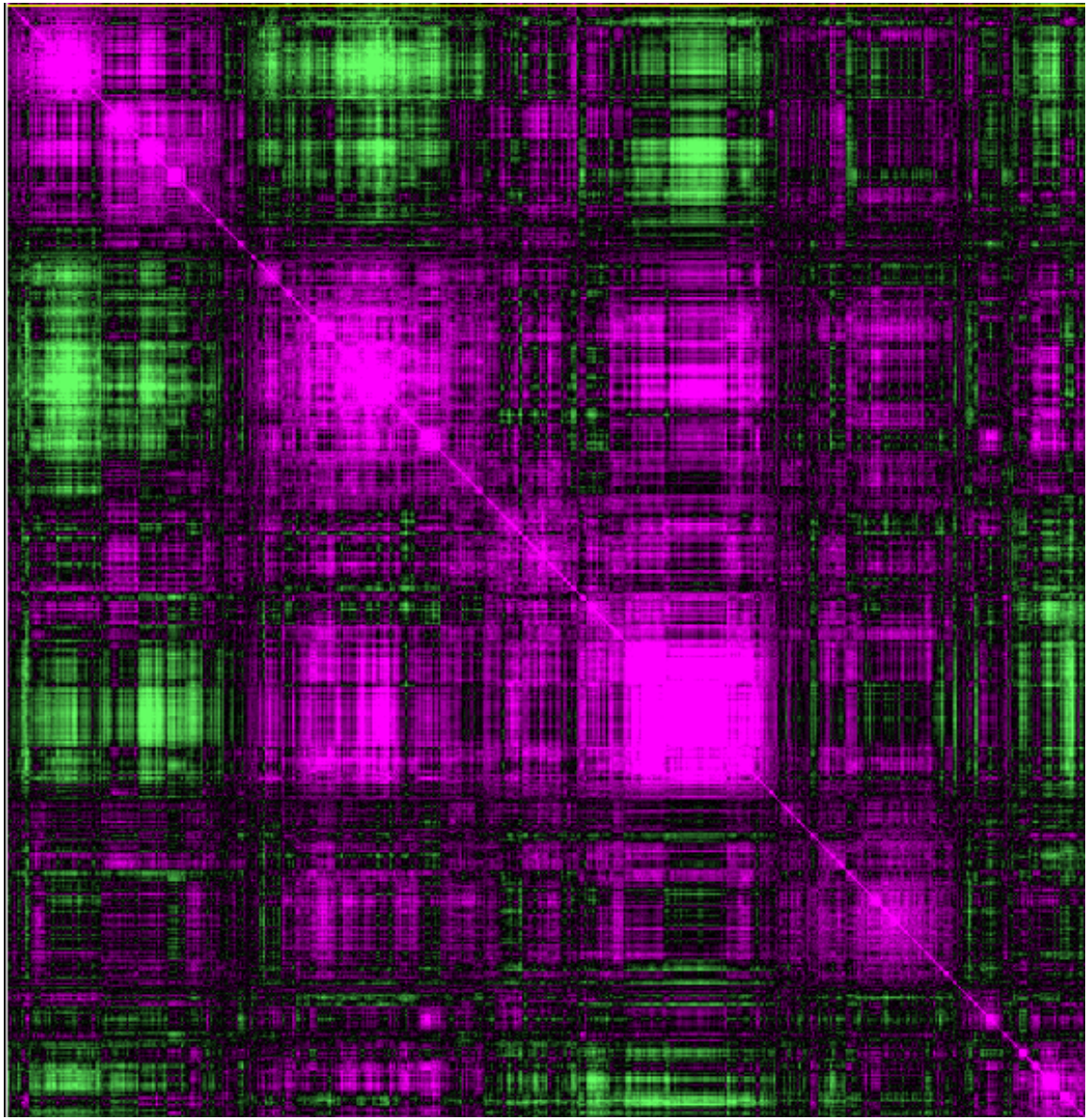
Mark Voorhies

5/15/2015

Gotchas

- Indentation matters

Clustering exercises – Visualizing the distance matrix



Loading and re-loading your functions

```
# Use import the first time you load a module
#   (And keep using import until it loads
#    successfully)
import my_module

my_module.my_function(42)

# Once a module has been loaded, use reload to
#   force python to read your new code
reload(my_module)
```

Setting Canopy's working/import directory

OS X

- Open a terminal
- cd path/to/working/directory
- env PYTHONPATH="$PYTHONPATH:$PWD" canopy

Windows (or OS X)

- Start canopy
- %cd path/to/working/directory
- import sys, os
- sys.path.append(os.getcwd())

Pearson distances

Pearson similarity

$$s(x,y) = \frac{1}{N}\sum_{i}^{N}\left(\frac{x_i - x_{offset}}{\phi_x}\right)\left(\frac{y_i - y_{offset}}{\phi_y}\right)$$

$$\phi_G = \sqrt{\sum_{i}^{N}\frac{(G_i - G_{offset})^2}{N}}$$

Pearson distances

Pearson similarity

$$s(x,y) = \sum_{i}^{N} \left(\frac{x_i - x_{offset}}{\phi_x} \right) \left(\frac{y_i - y_{offset}}{\phi_y} \right)$$

$$\phi_G = \sqrt{\sum_{i}^{N} (G_i - G_{offset})^2}$$

Pearson distances

Pearson similarity

$$s(x,y) = \sum_{i}^{N} \left(\frac{x_i - x_{offset}}{\sqrt{\sum_i^N (x_i - x_{offset})^2}} \right) \left(\frac{y_i - y_{offset}}{\sqrt{\sum_i^N (y_i - y_{offset})^2}} \right)$$

Pearson distances

Pearson similarity

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

Pearson distances

Pearson similarity

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

Pearson distance

$$d_{uncentered}(x,y) = 1 - s(x,y)$$

Pearson distances

Pearson similarity

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

Pearson distance

$$d_{uncentered}(x,y) = 1 - s(x,y)$$

Euclidean distance

$$\frac{\sum_i^N (x_i - y_i)^2}{N}$$

Clustering exercises – Negative controls

Write functions to reproduce the shuffling controls in figure 3 of the Eisen paper (removing correlations among genes and/or arrays).

Clustering exercises – Negative controls

Write functions to reproduce the shuffling controls in figure 3 of the Eisen paper (removing correlations among genes and/or arrays).

```
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.geneName[i] for i in indices]
    self.geneName = genes
    annotations = [self.geneAnn[i] for i in indices]
    self.geneAnn = genes
    num = [self.num[i] for i in indices]
    self.num = num
```

Clustering exercises – Negative controls

Write functions to reproduce the shuffling controls in figure 3 of the Eisen paper (removing correlations among genes and/or arrays).

Clustering exercises – Negative controls

Write functions to reproduce the shuffling controls in figure 3 of the Eisen paper (removing correlations among genes and/or arrays).

```
def shuffleRows(self, seed = None):
    """Permute ratio values within rows."""
    import random
    if(seed != None):
        random.seed(seed)
    for i in self.num:
        random.shuffle(i)
```

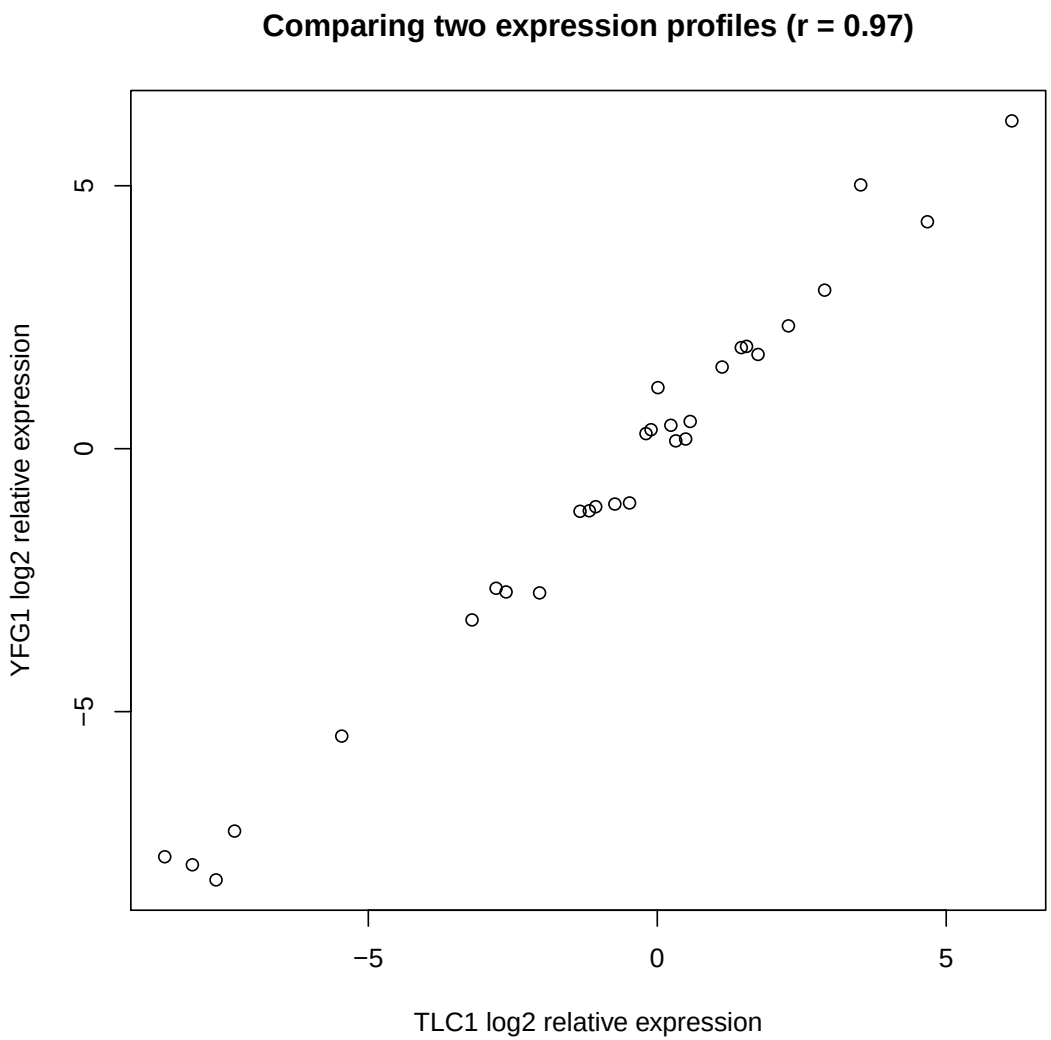
Clustering exercises – Negative controls

Write functions to reproduce the shuffling controls in figure 3 of the Eisen paper (removing correlations among genes and/or arrays).

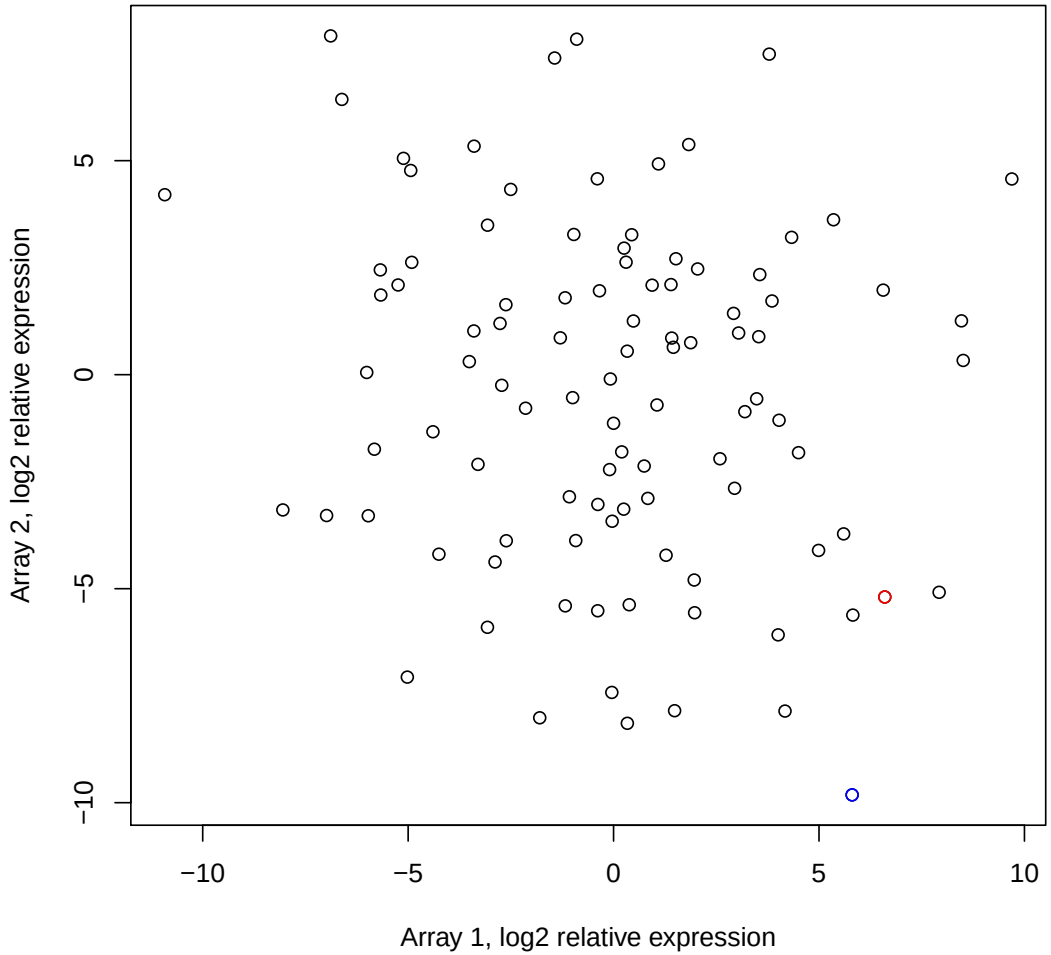
```
def shuffleRows(self, seed = None):
    """Permute ratio values within rows."""
    import random
    if(seed != None):
        random.seed(seed)
    for i in self.num:
        random.shuffle(i)


def shuffleCols(self, seed = None):
    """Permute ratio values within columns."""
    import random
    if(seed != None):
        random.seed(seed)
    # Transpose the expression matrix
    cols = []
    for col in xrange(len(self.num[0])):
        cols.append([row[col] for row in self.num])
    # Shuffle
    for i in cols:
        random.shuffle(i)
    # Transpose back to original orientation
    self.num = []
    for row in xrange(len(cols)):
        self.num.append([col[row] for col in row])
```

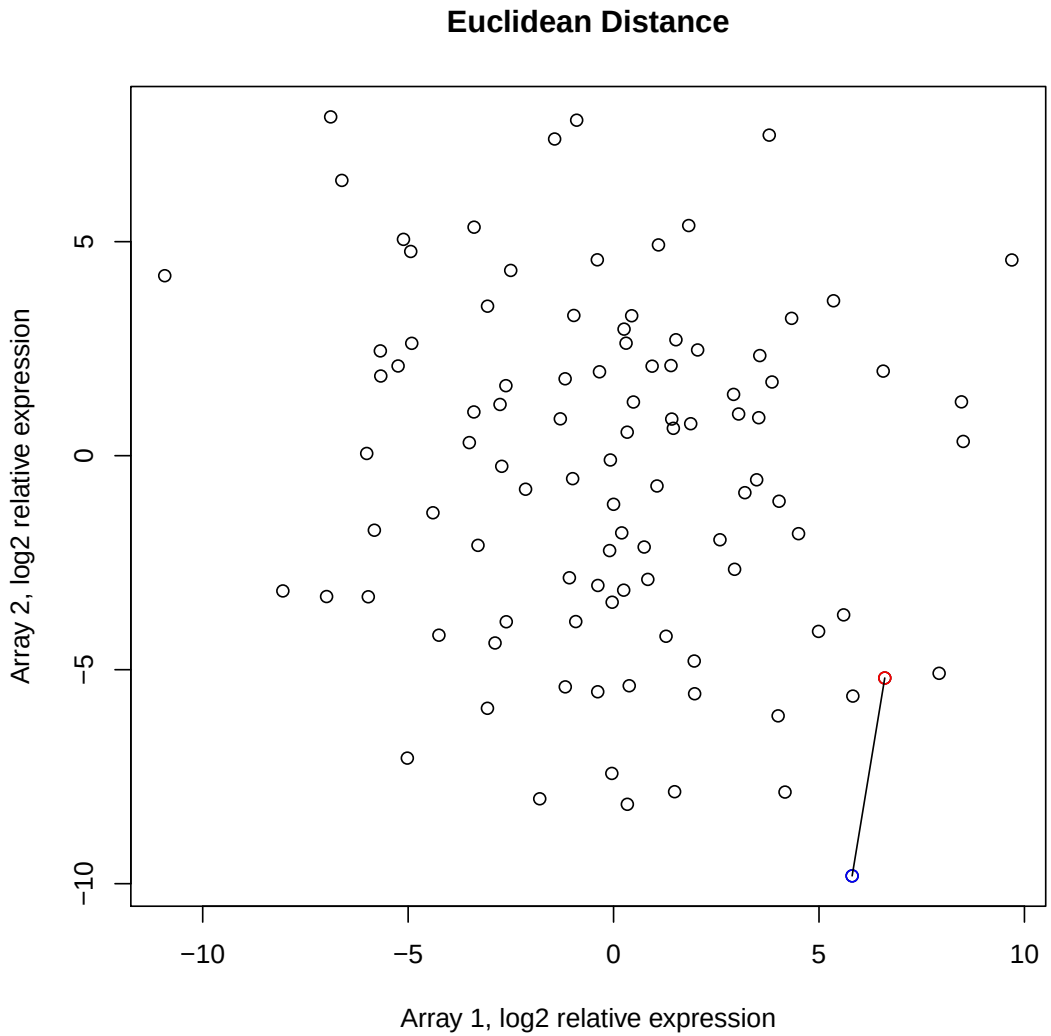
Comparing all measurements for two genes



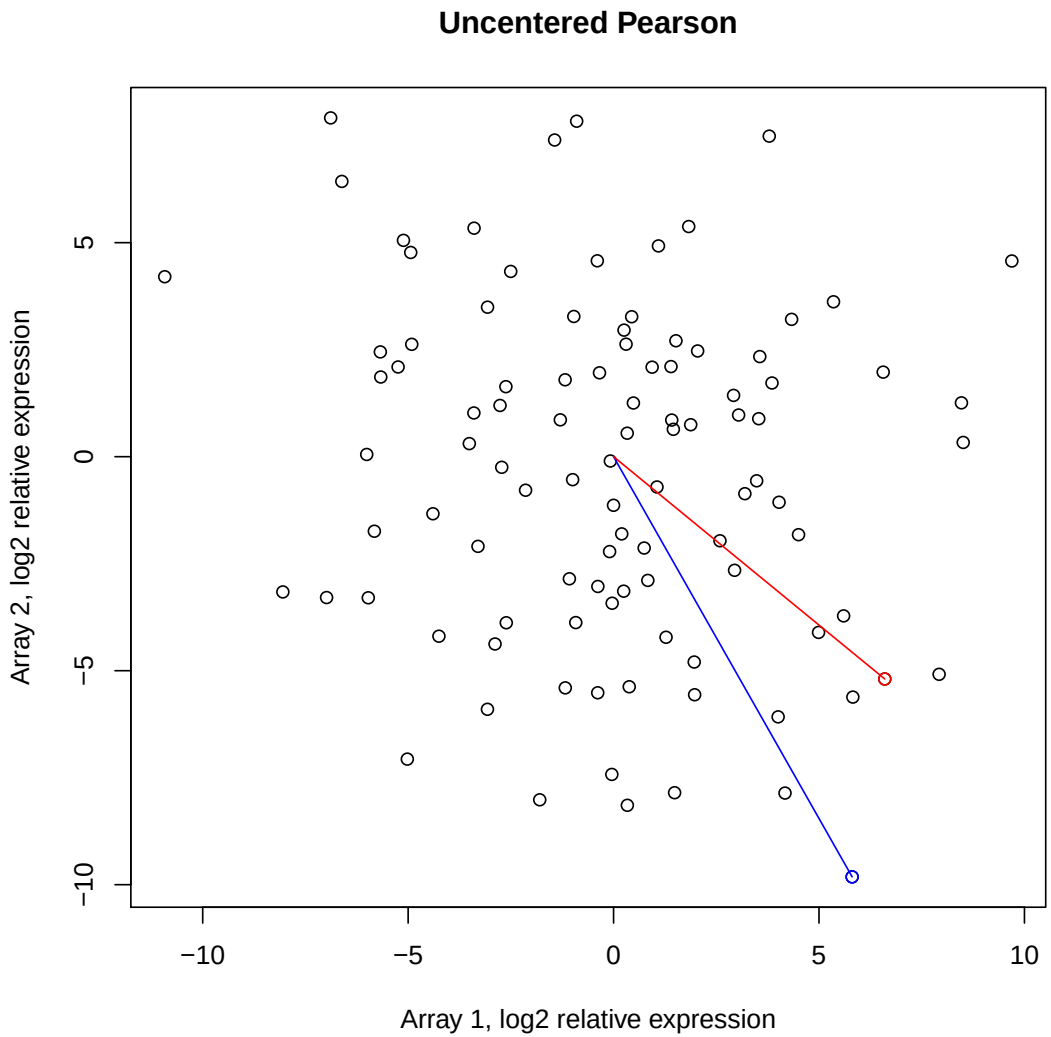
Comparing all genes for two measurements



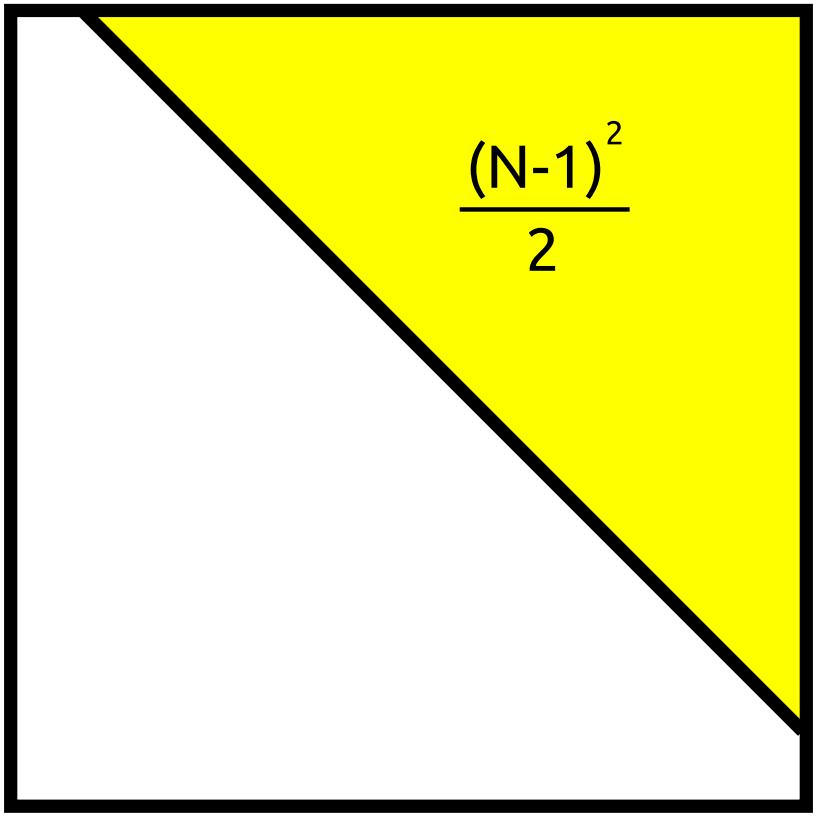
Comparing all genes for two measurements



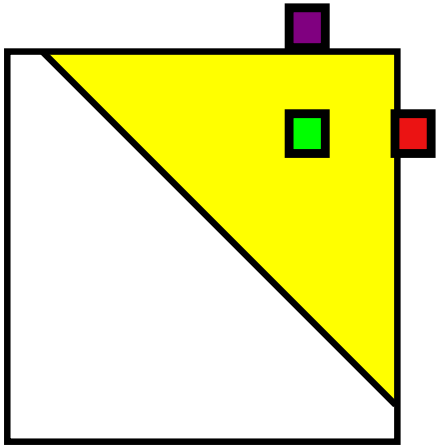
Comparing all genes for two measurements

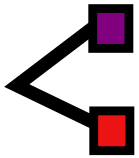


Measure all pairwise distances under distance metric

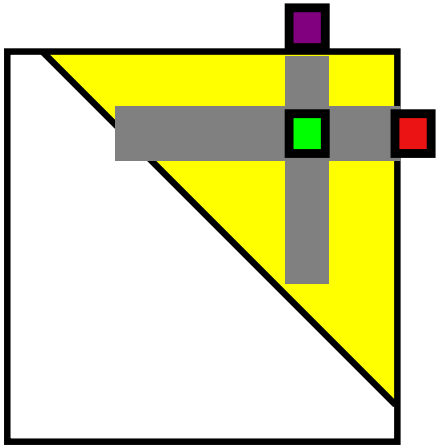


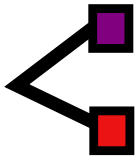
Hierarchical Clustering



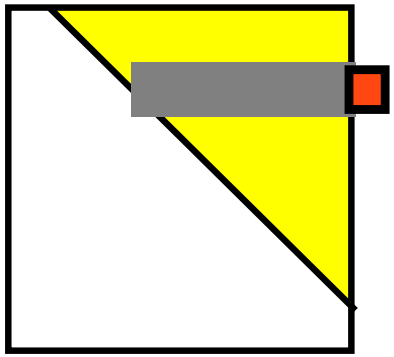


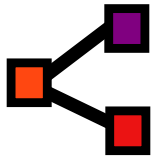
Hierarchical Clustering



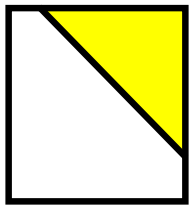


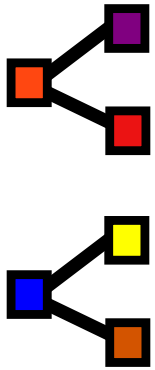
Hierarchical Clustering



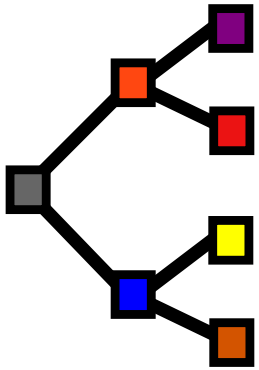


Hierarchical Clustering





Hierarchical Clustering



Scripting Cluster

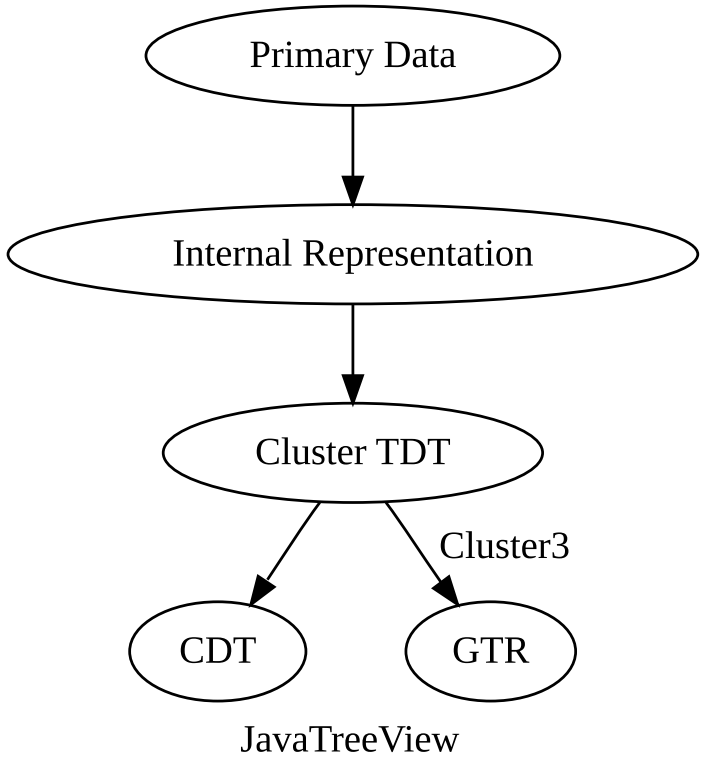
- Running Cluster3 from the command line
  - /Applications/Cluster.app/Contents/MacOS/Cluster
  - /Program Files/Stanford University/Cluster3/Cluster.com
- Command-line programs are like functions
- "man program" is like "help(function)"
- Use the `subprocess` module to run command-line programs from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

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)

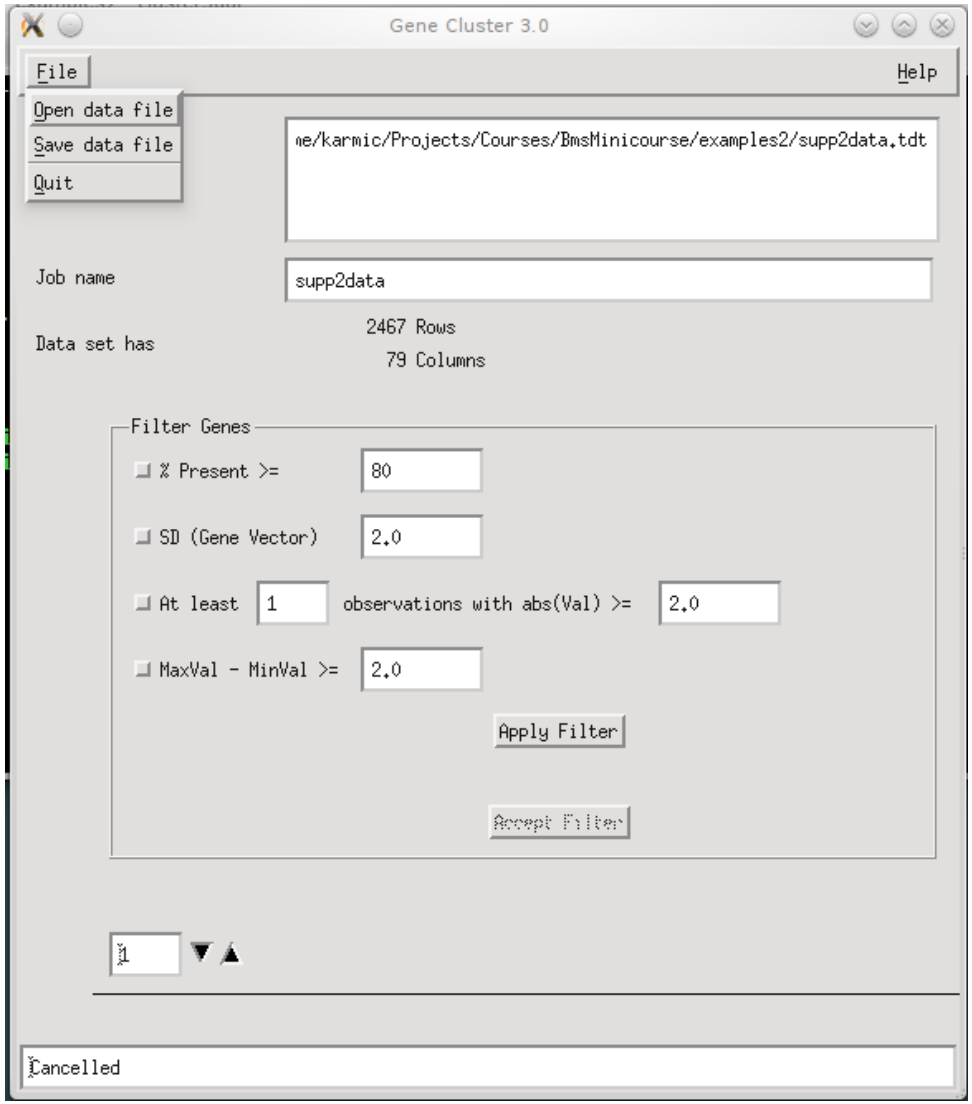
Scripting the Protocol

```
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"))
```

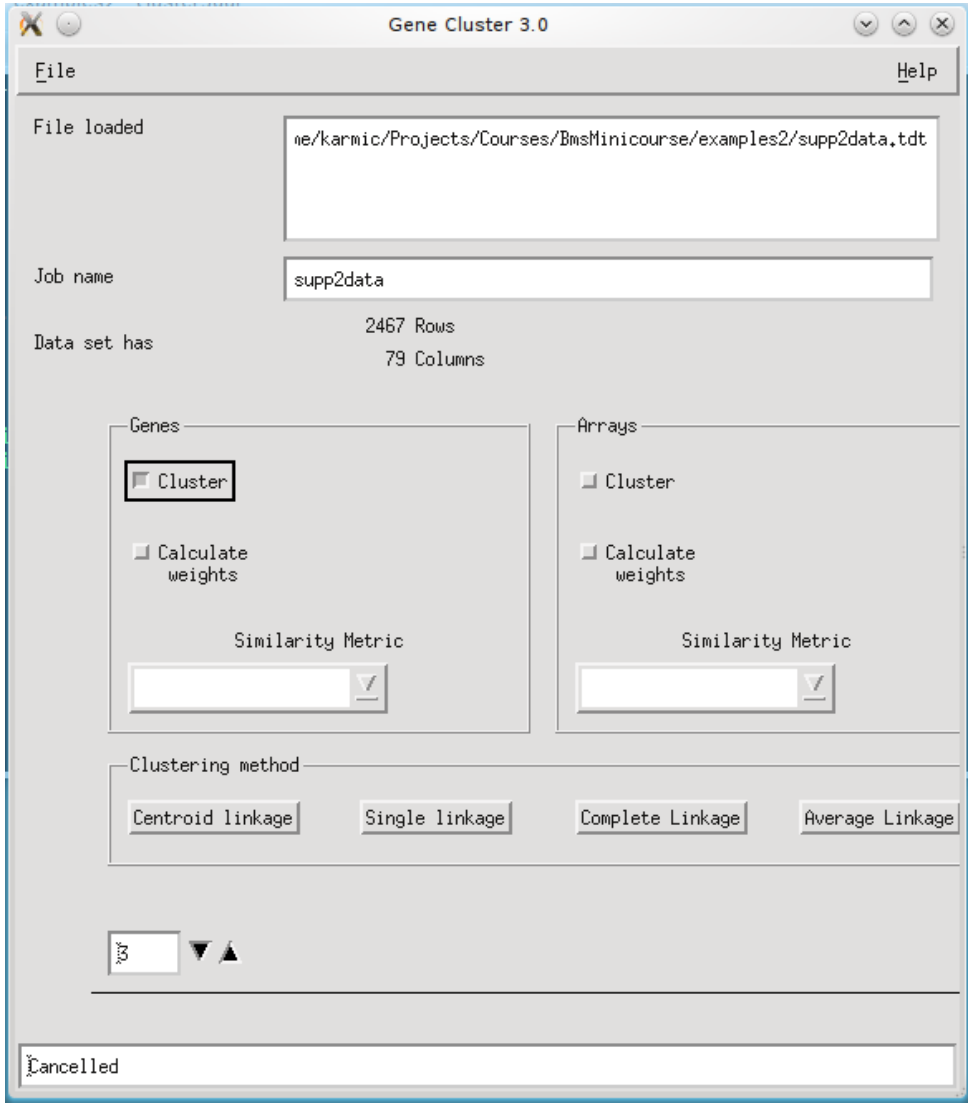


Using the Cluster3 GUI

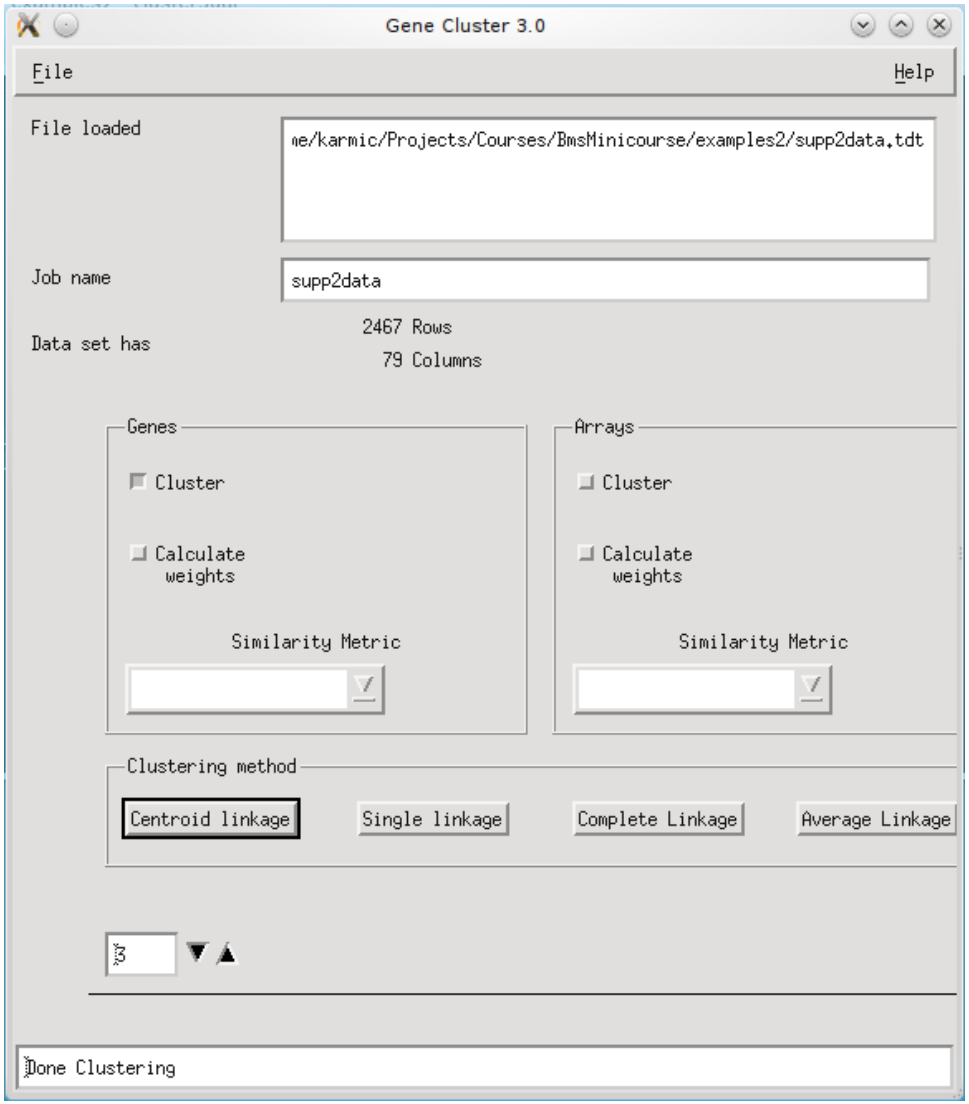
Load your data



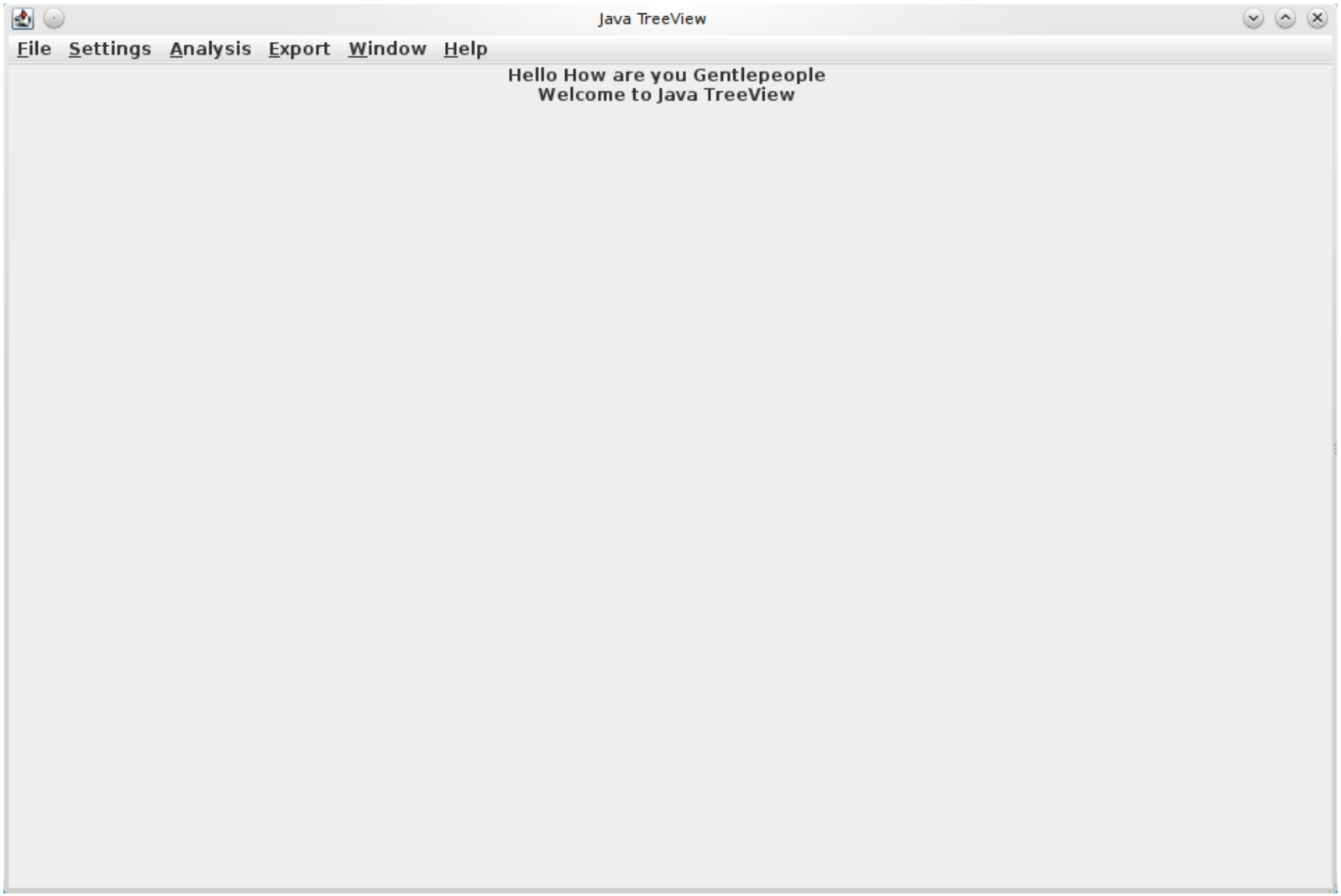
Choose distance function



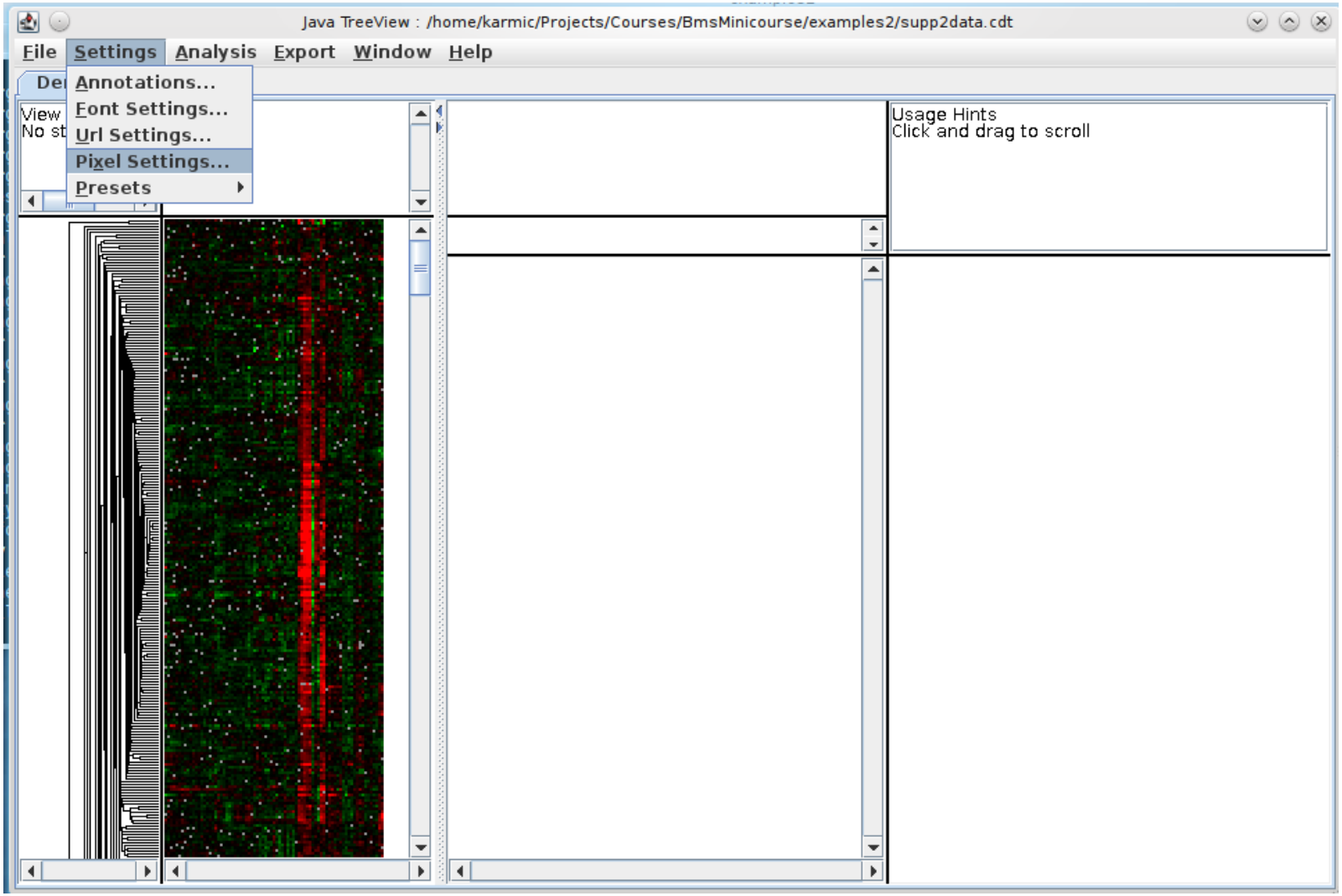
Choose linking method



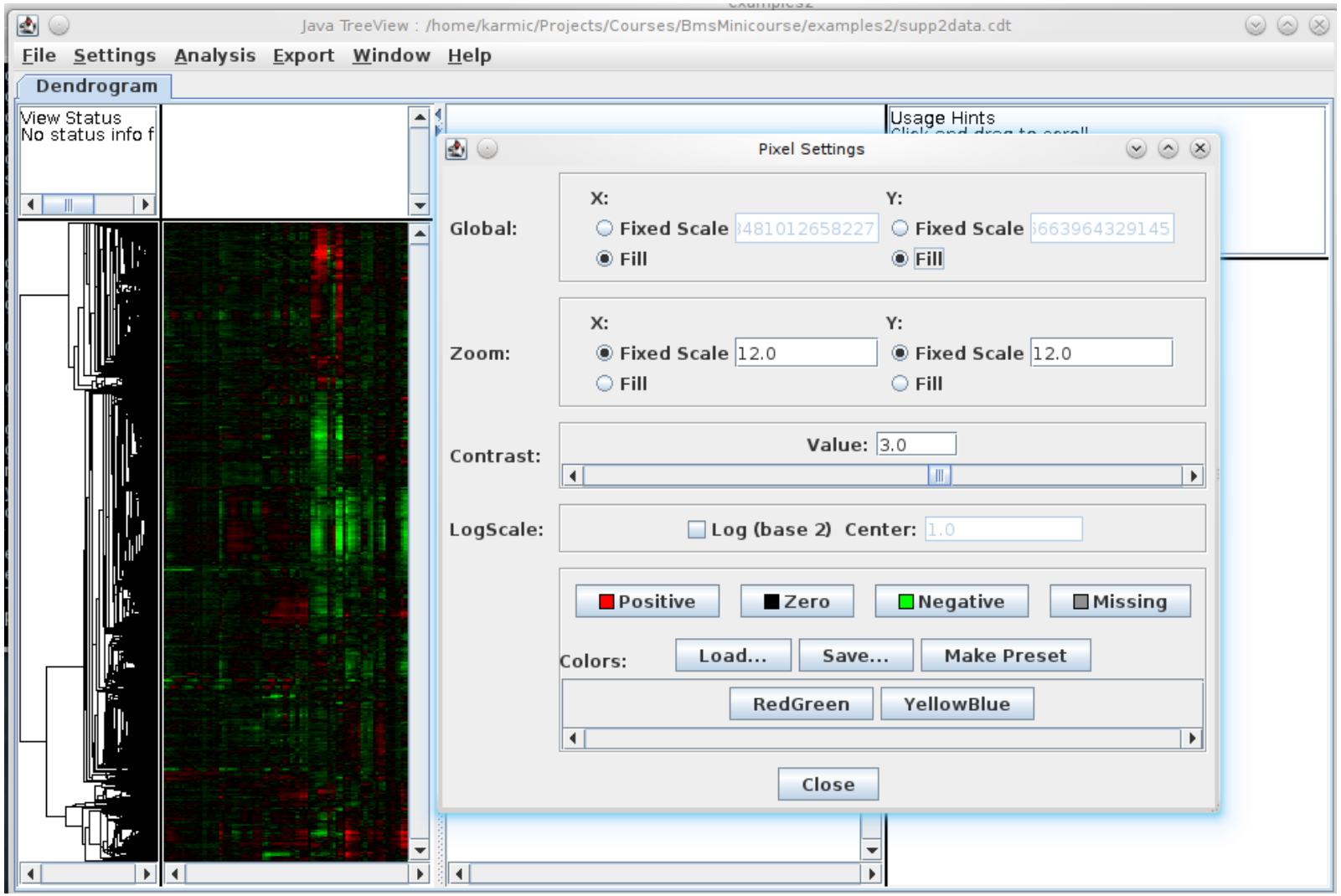
Using JavaTreeView



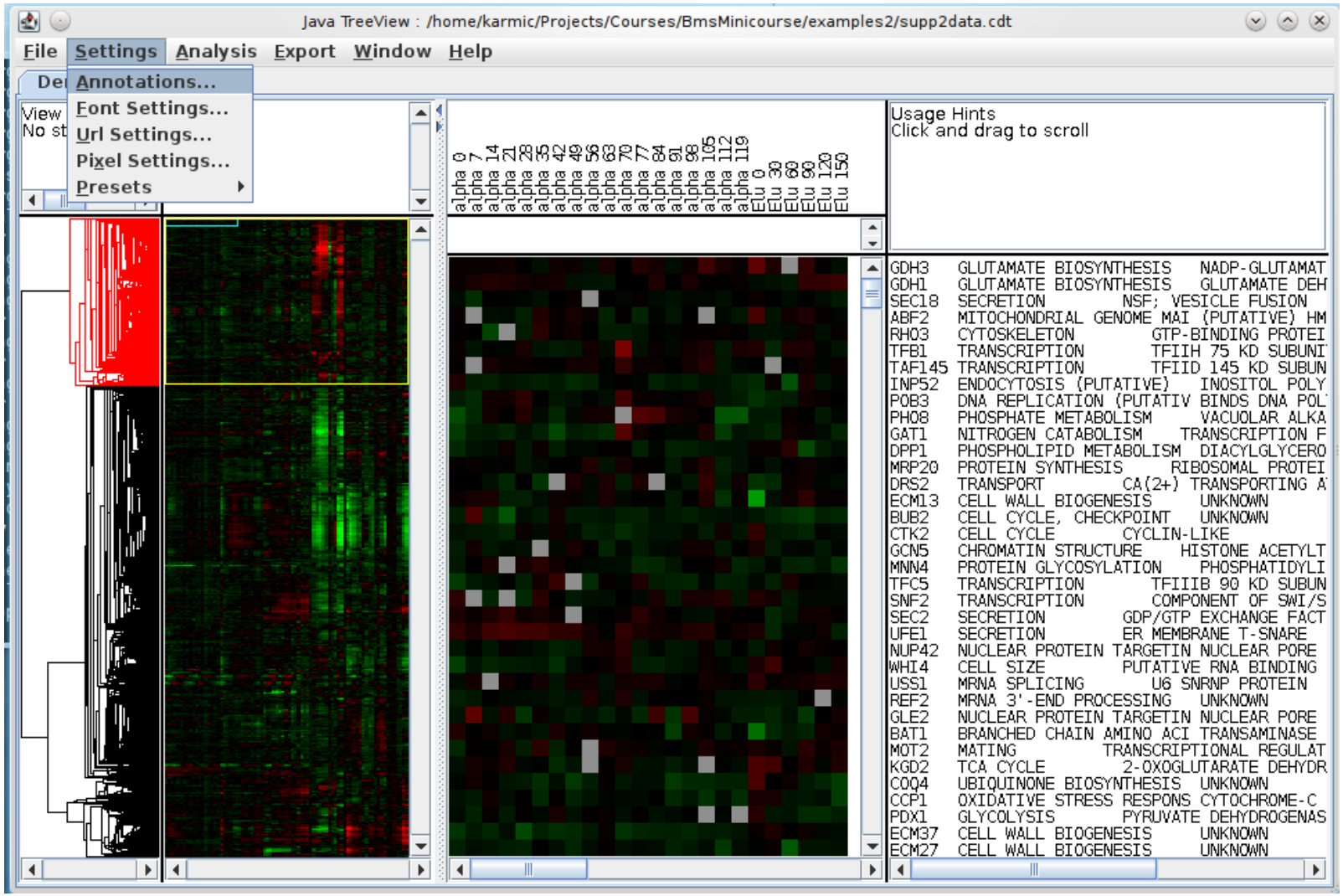
Adjust pixel settings for global view



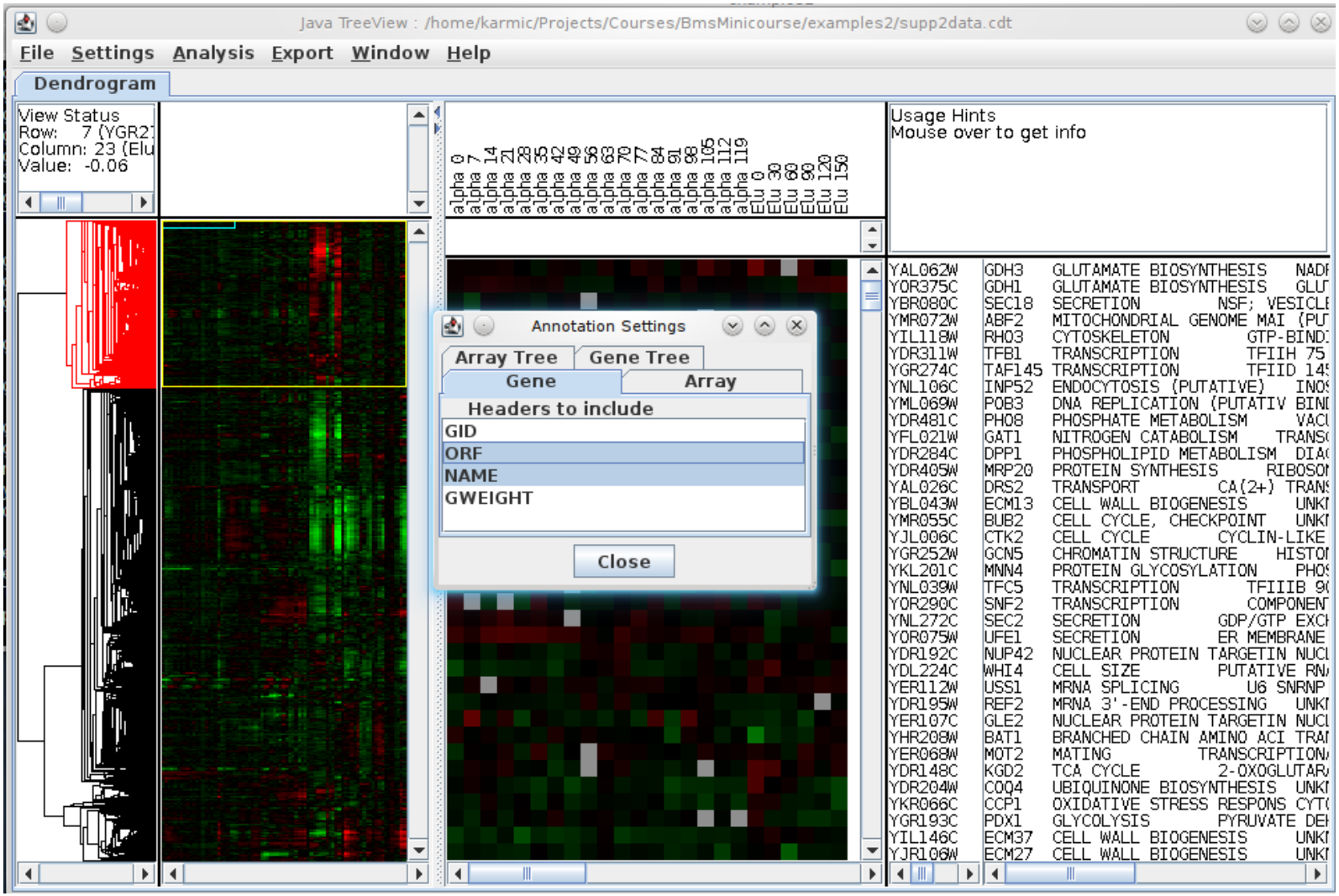
Adjust pixel settings for global view



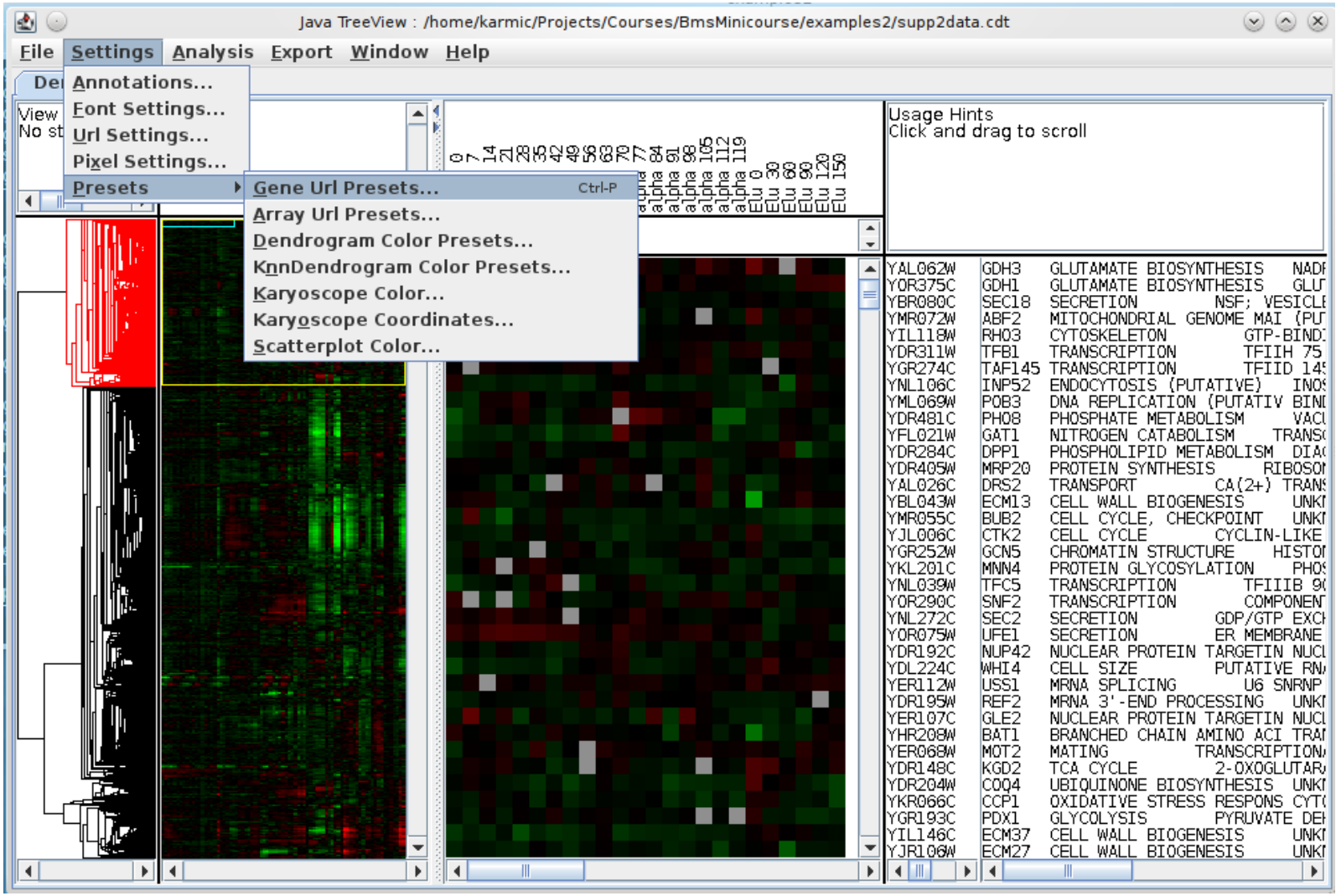
Select annotation columns



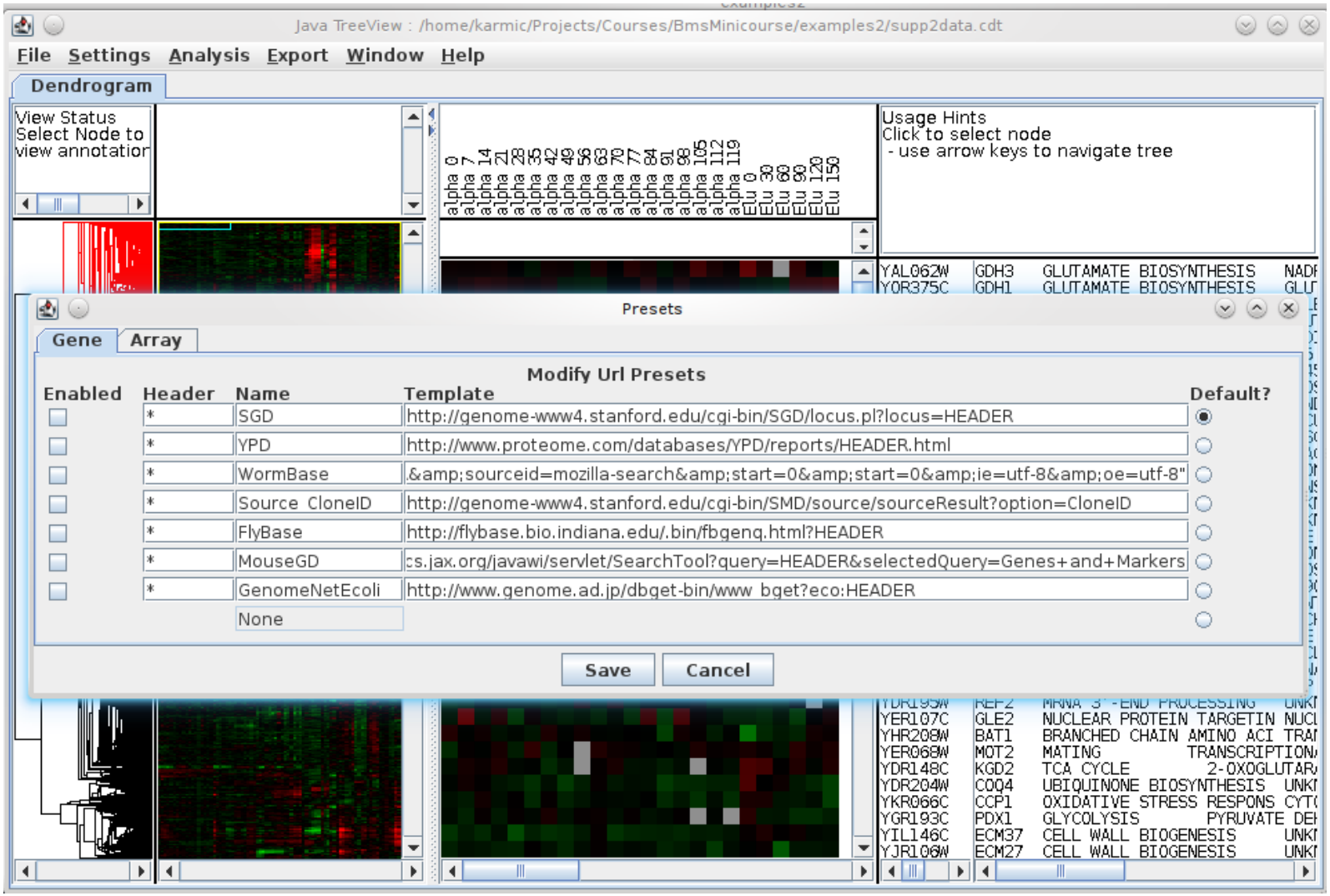
Select annotation columns



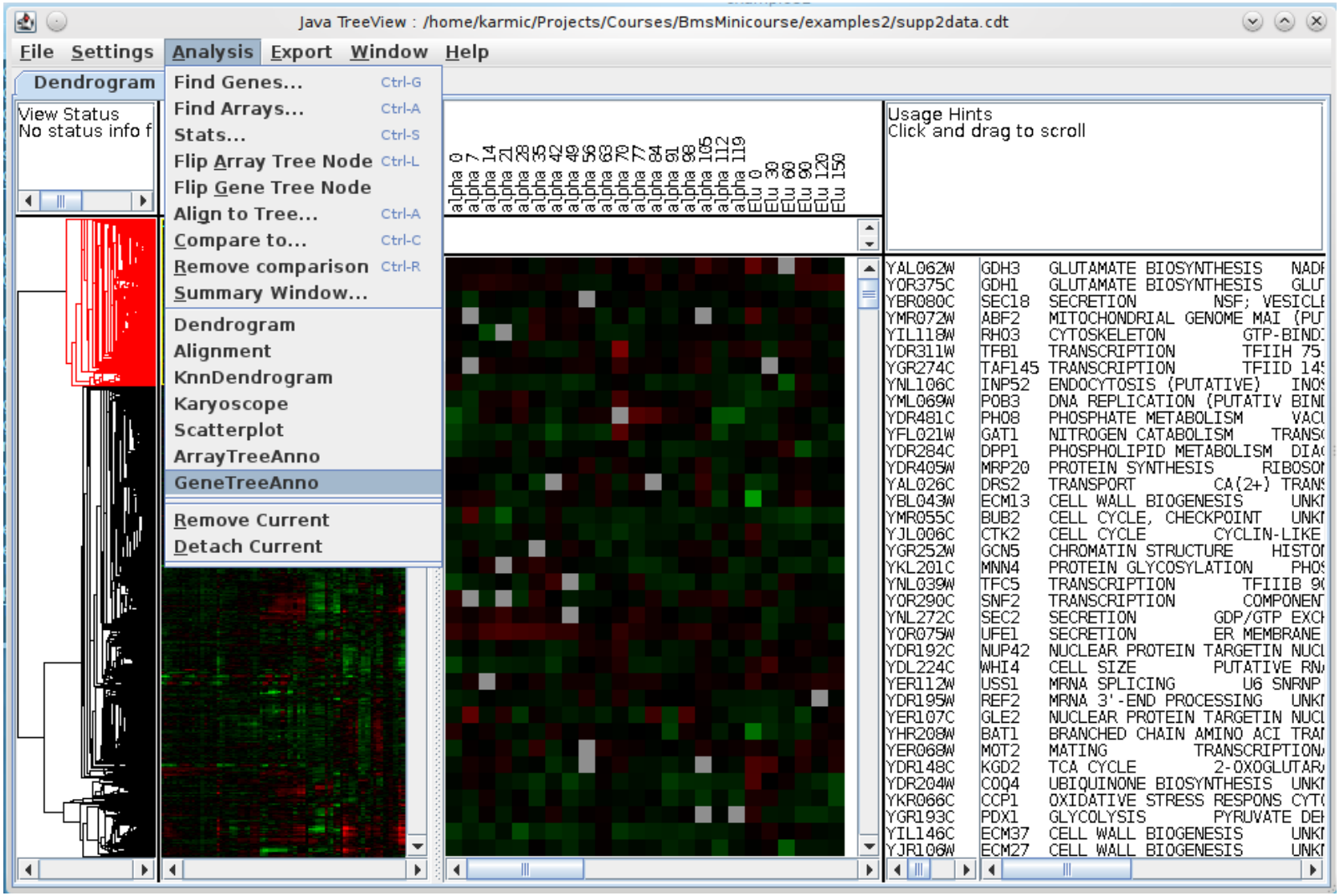
Select URL for gene annotations



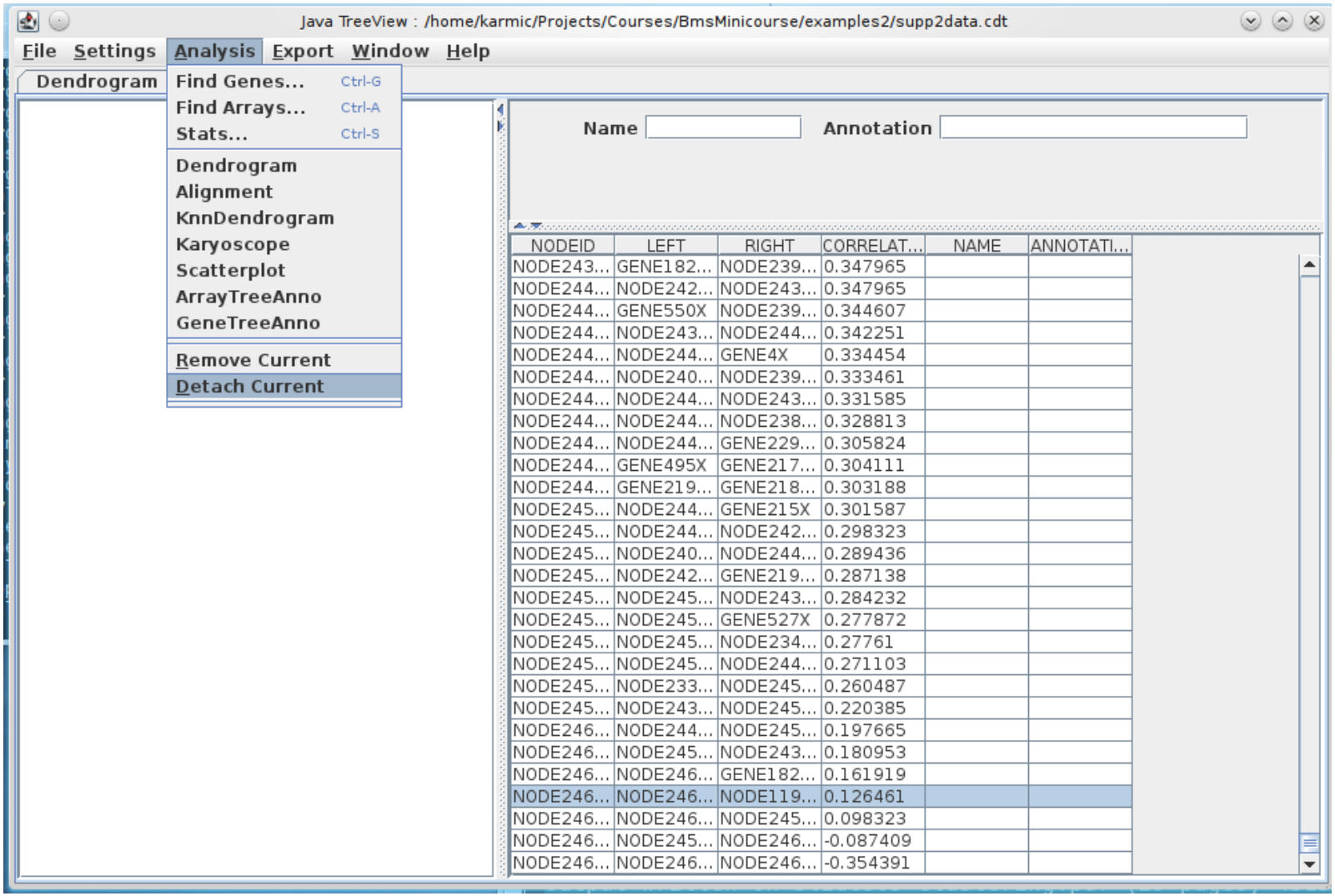
Select URL for gene annotations



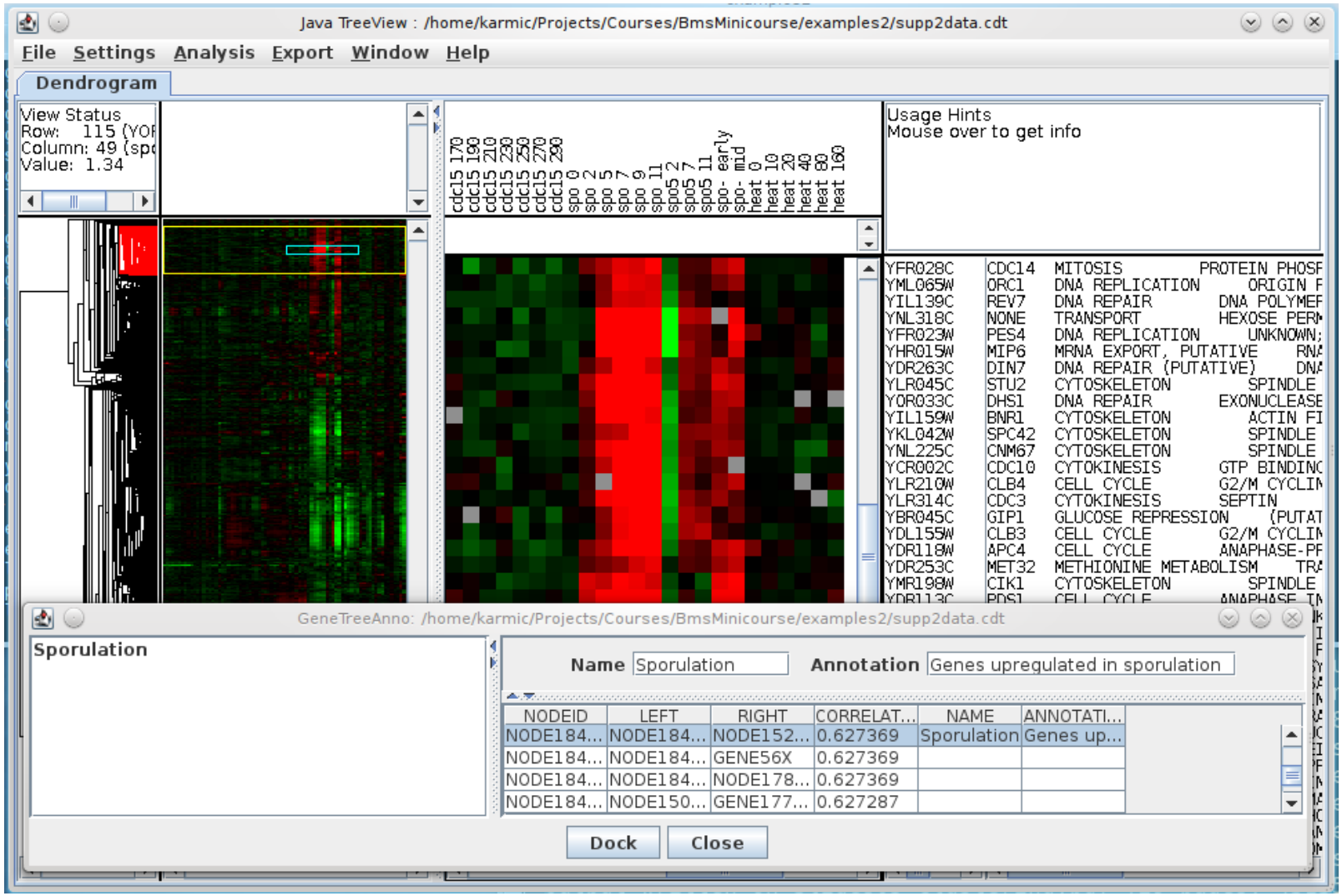
Activate and detach annotation window



Activate and detach annotation window



Activate and detach annotation window



Clustering exercises – Scripting Cluster

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.

Clustering exercises – Scripting Cluster

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.

```
metrics = ("None",
           "Uncentered",
           "Pearson",
           "UncenteredAbs",
           "PearsonAbs",
           "Spearman",
           "Kendall",
           "Euclidean",
           "City")
linkage = (("Complete","m"),
           ("Single","s"),
           ("Centroid","c"),
           ("Average","a"))

# Loop over all 32 possible methods
print "Starting hierarchical clustering runs..."
from subprocess import check_call
for metric in xrange(1,len(metrics)):
    print "    ",metrics[metric],"..."
    for (linkname, link) in linkage:
        print "        ",linkname
        check_call(("cluster","-f","shuffled.txt",
                    "-u",".".join(("shuffled",
                                   metrics[metric],
                                   linkname)),
                    "-m",link,"-g",str(metric)))
```

Homework

1. If you haven't done so already, read the PNAS paper
2. Explore the figure 2 data with Cluster3 and JavaTreeView.
   - Can you find/reproduce the clusters described in the paper?
   - Are the annotations consistent with the current annotations in SGD?
   - Are there other patterns that you can find in the data?
   - What follow-up experiments are prompted by this analysis?