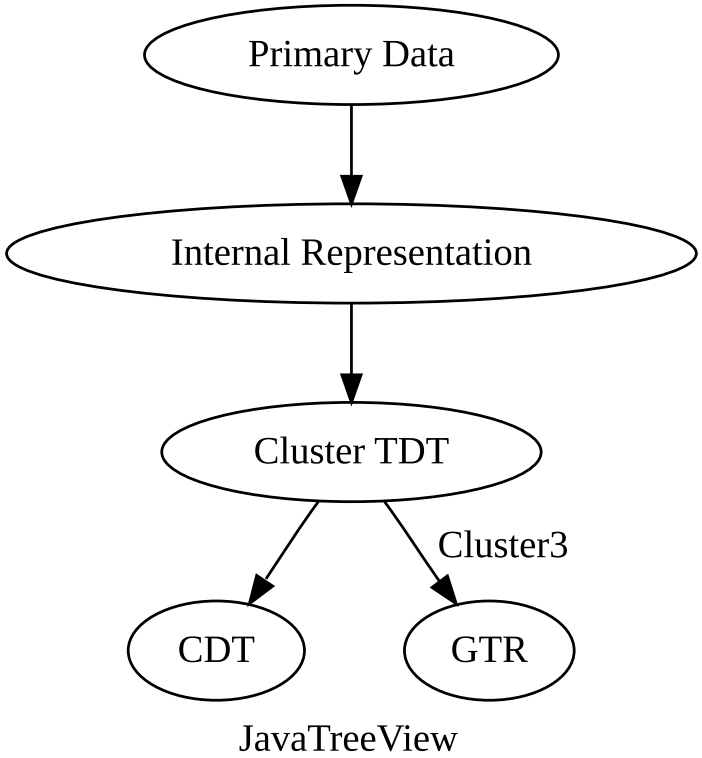
Practical Bioinformatics

Mark Voorhies

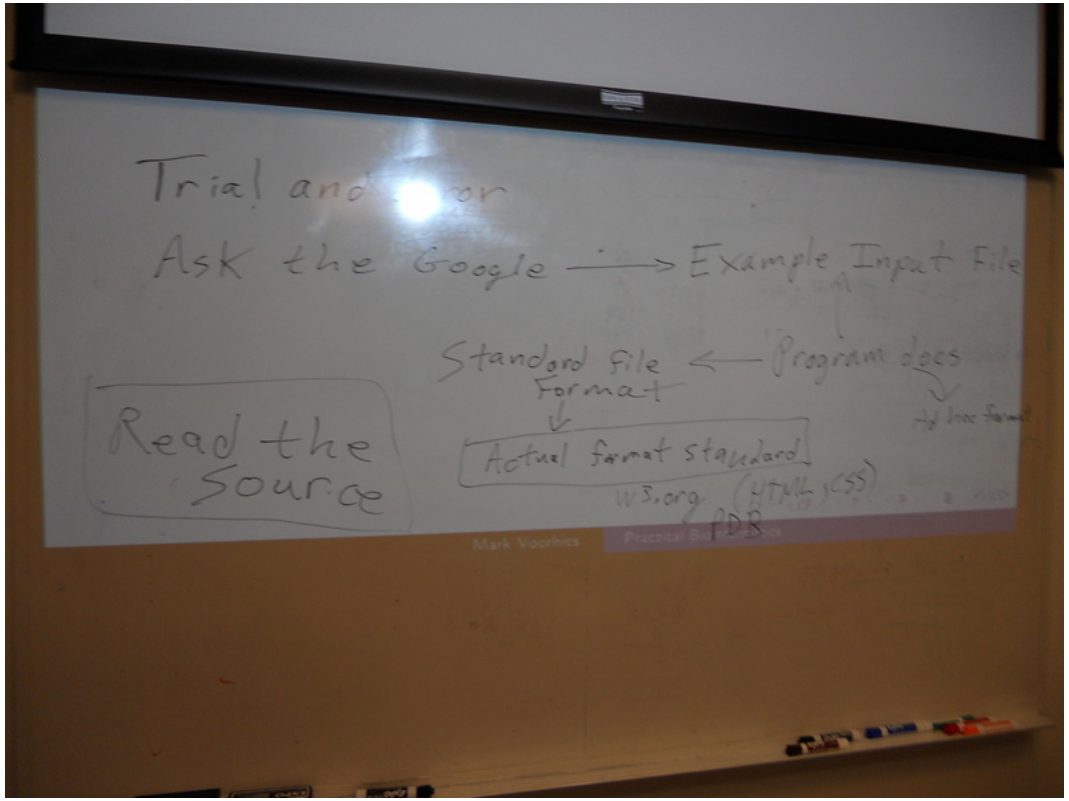
4/25/2011

Clustering protocol



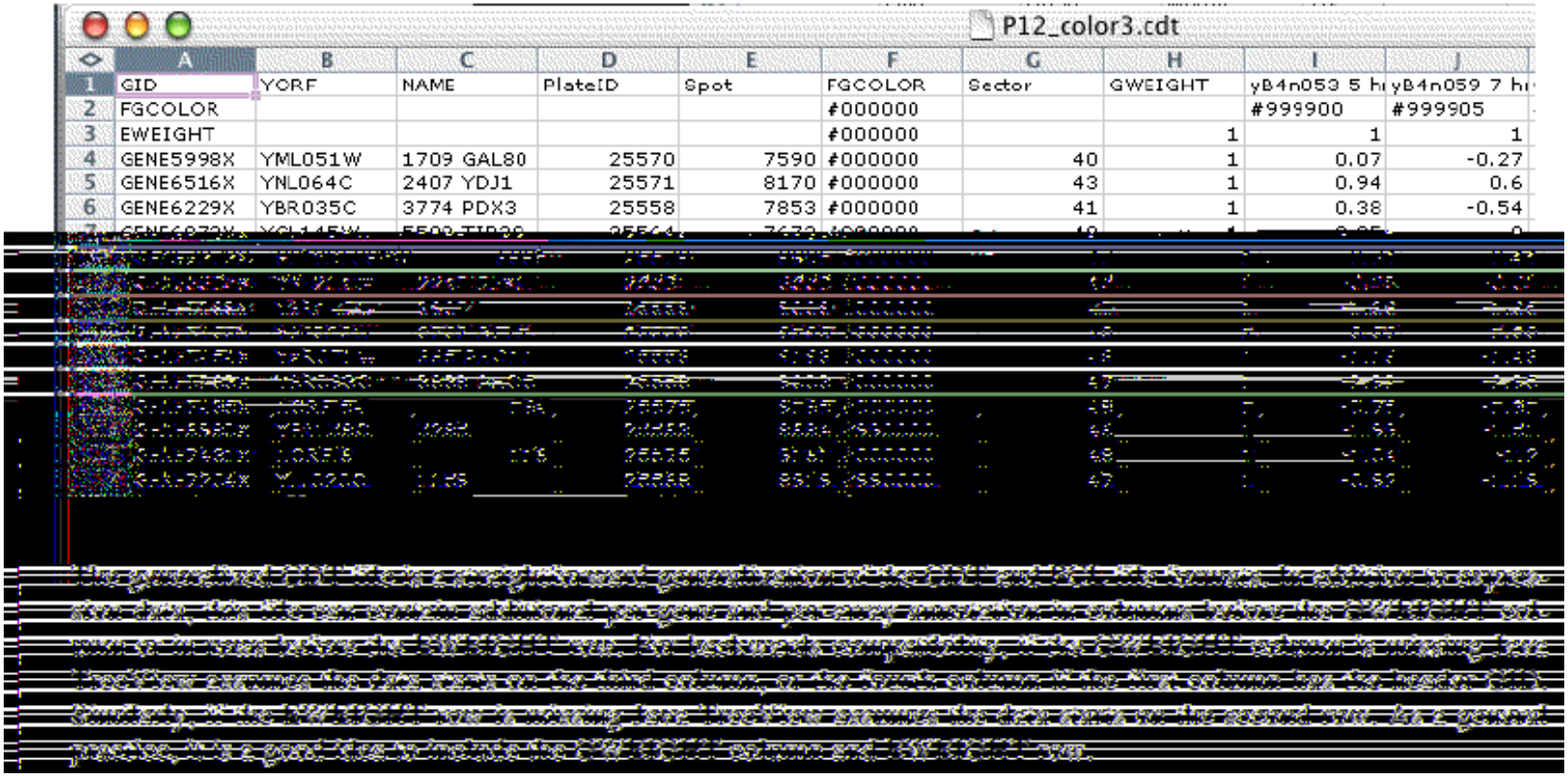
How can we discover how we should format a file?

Whiteboard Image



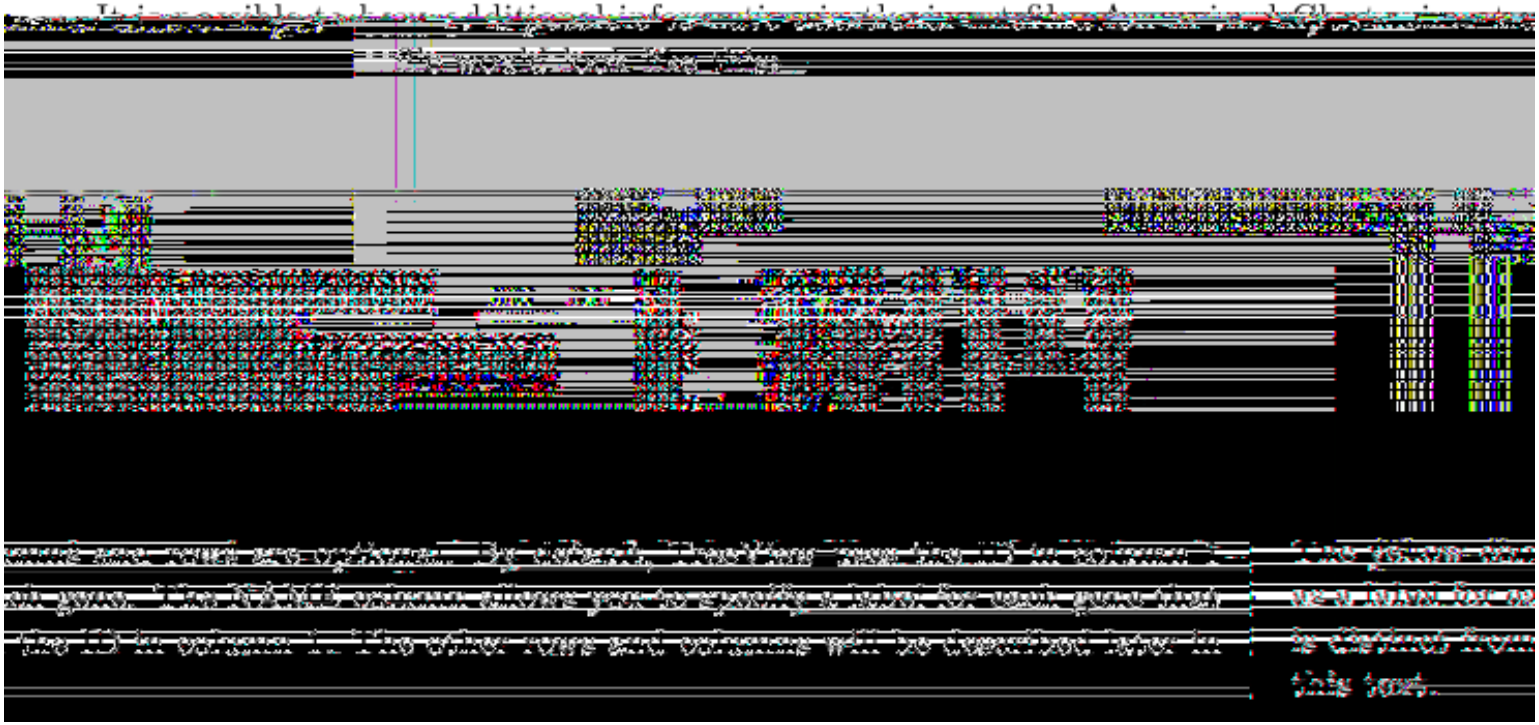
RTFM: JavaTreeView

Figure 2.16. Screenshot of Generalized CDT file in Excel



	A	B	C	D	E	F	G	H	I	J
1	GID	YORF	NAME	PlateID	Spot	FGCOLOR	Sector	GWEIGHT	yB4n053 5 h	yB4n059 7 h
2	FGCOLOR					#000000			#999900	#999905
3	EWEIGHT					#000000		1	1	1
4	GENE5998X	YML051W	1709 GAL80	25570	7590	#000000	40	1	0.07	-0.27
5	GENE6516X	YNL064C	2407 YDJ1	25571	8170	#000000	43	1	0.94	0.6
6	GENE6229X	YBR035C	3774 PDX3	25558	7853	#000000	41	1	0.38	-0.54

RTFM: Cluster3



How does Cluster3 handle newlines?

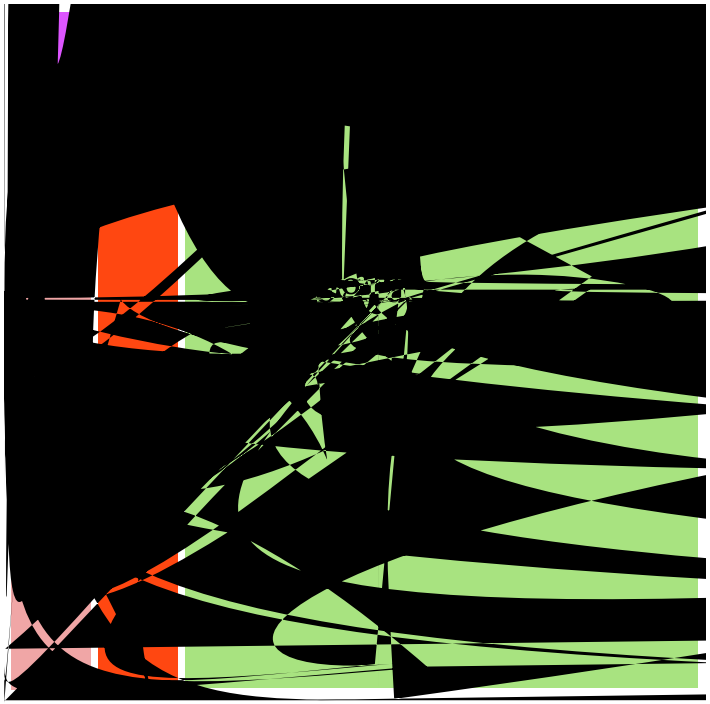
```
static char* GetLine(FILE* inputfile)
/* The function GetLine reads one line from the inputfile, and returns it as a
 * null-terminated string. If inputfile is at EOF, a null pointer is returned.
 * The calling routine should free the char* returned by GetLine.
 */
{ int c;
  int n = 0;
  int size = 1023;
  char* line = malloc((size+1)*sizeof(char));
  while ((c = getc(inputfile))!=EOF && c!='\r' && c!='\n')
  { if (n == size)
    { size *= 2;
      line = realloc(line,(size+1)*sizeof(char));
    }
    line[n] = (char)c;
    n++;
  }
  if (c=='\r')
  { c = getc(inputfile);
    if (c!='\n' && c!=EOF) ungetc(c,inputfile);
  }
  if (n==0 && c==EOF)
  { free(line);
    return 0;
  }
  line[n] = '\0';
  line = realloc(line,(n+1)*sizeof(char));
  return line;
}
```

How does Cluster3 handle delimiters?

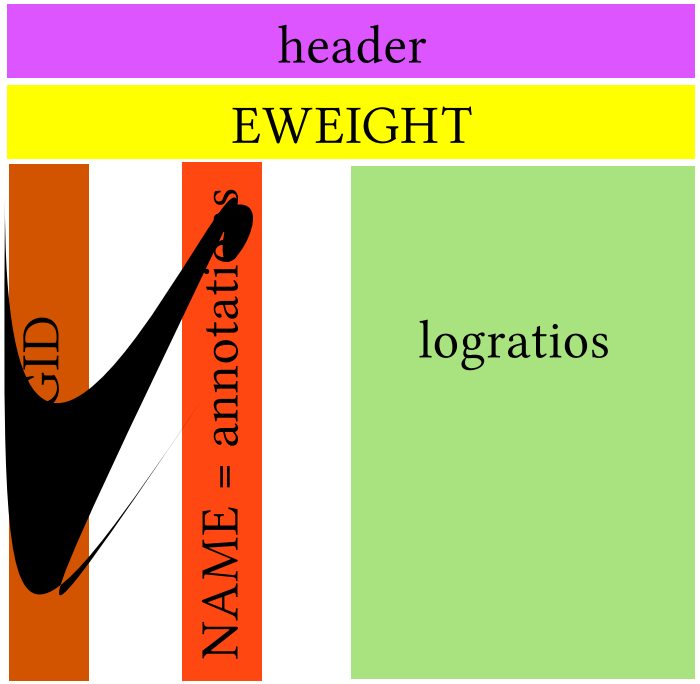
```
static char* tokenize(char* s)
{
  char* p = s;
  while (1)
  {
    if (*p=='\0') return NULL;
    if (*p=='\t')
    {
      *p = '\0';
      return p+1;
    }
    p++;
  }
  /* Never get here */
  return NULL;
}
```

The CDT file format

Minimal CLUSTER input



Cluster3 CDT output



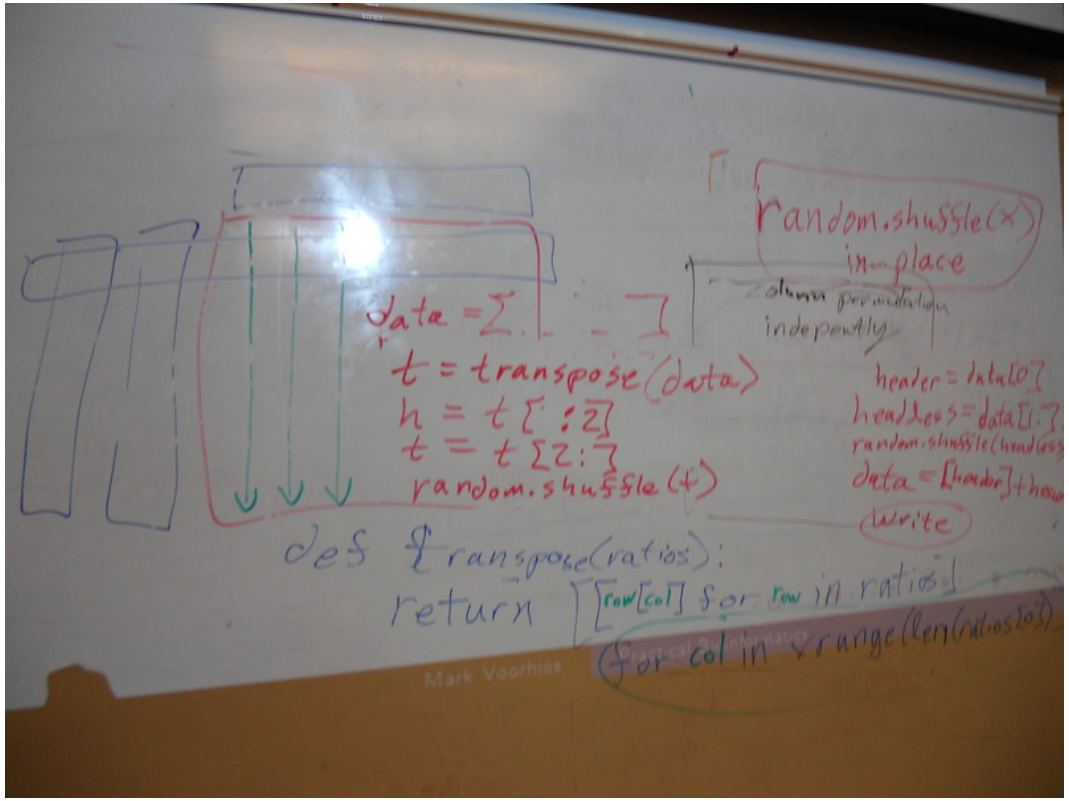
- Tab delimited (\t)
- UNIX newlines (\n)
- Missing values $\rightarrow$ empty cells

Be liberal in what you accept, strict in what you emit

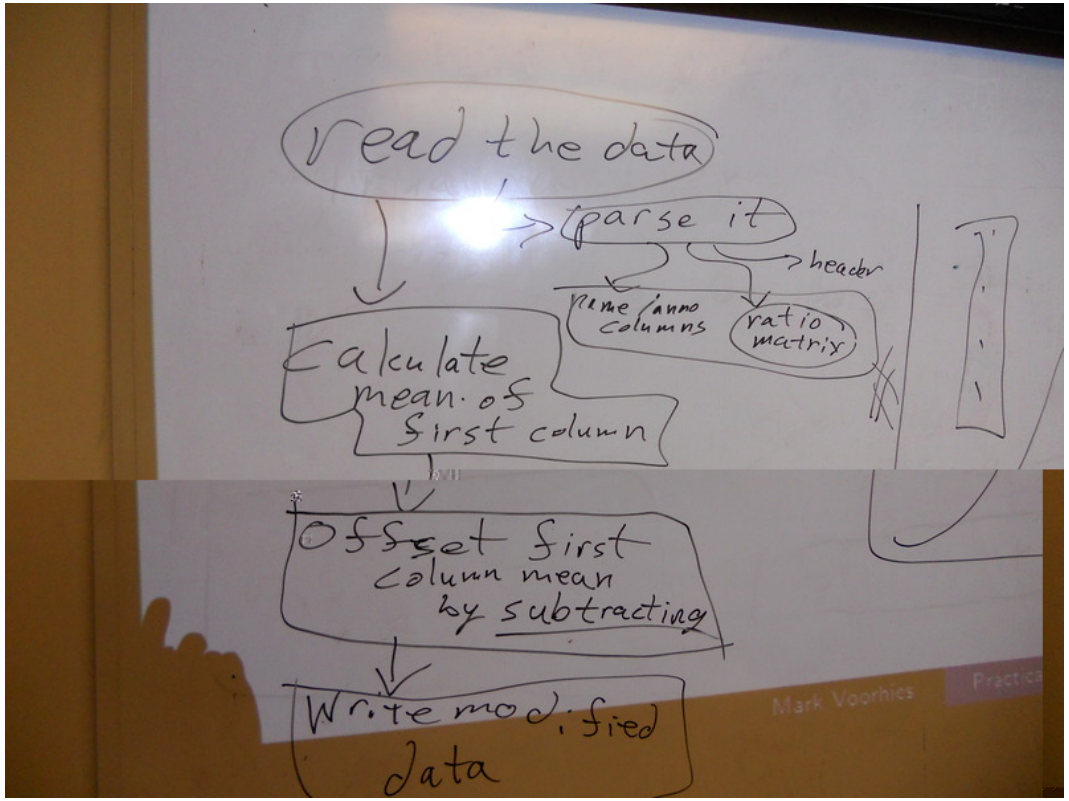
```
def writerow(out, row):
    out.write(
        # Use tab delimiters
        "\t".join(
        # Normalize all whitespace (including \t, \r, and \n) and "
        #   to "normal" spaces.
        [re.sub(r'[\s"]'," ",str(i)) for i in row])
        # Use UNIX-style newlines
        +"\n")

def writecdt(filename, matrix,
             row_names, row_annotations, columns):
    out = open(filename,"w")
    writerow(out, ["UNIQID","NAME"]+columns)
    for (uniqid,name,row) in zip(row_names,row_annotations,matrix):
        writerow(out, [uniqid,name]+row)
    out.close()
```

Strategies for Shuffling

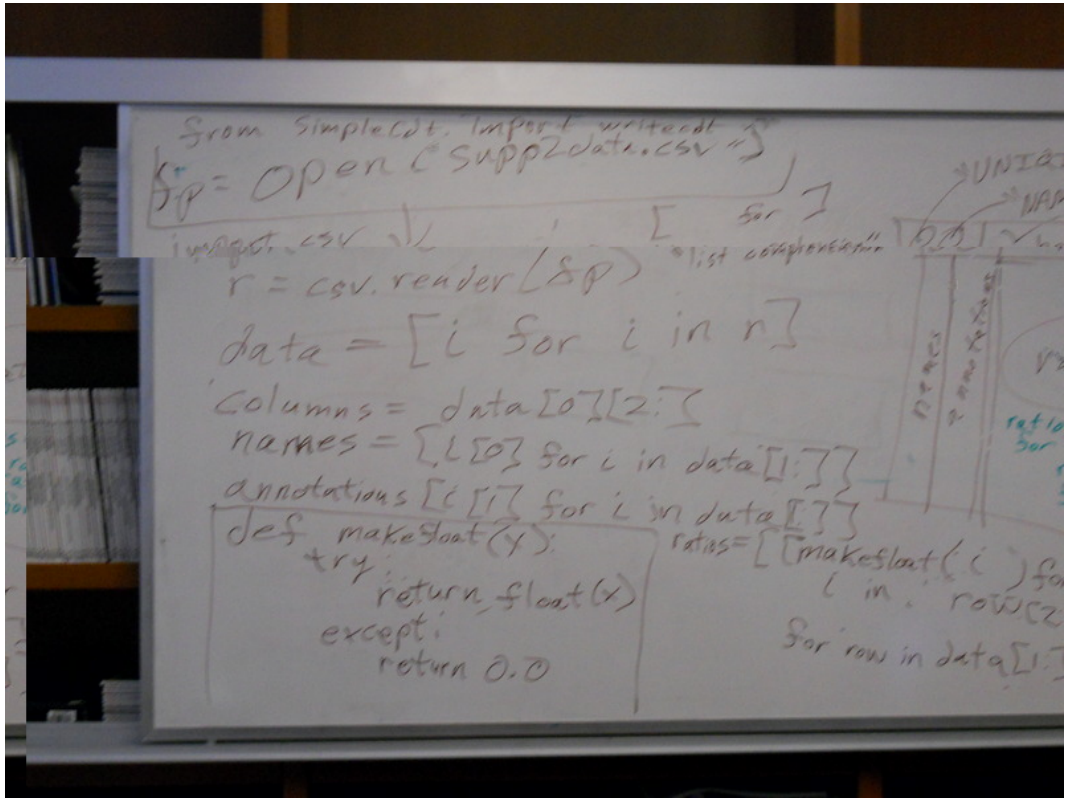
Whiteboard Image



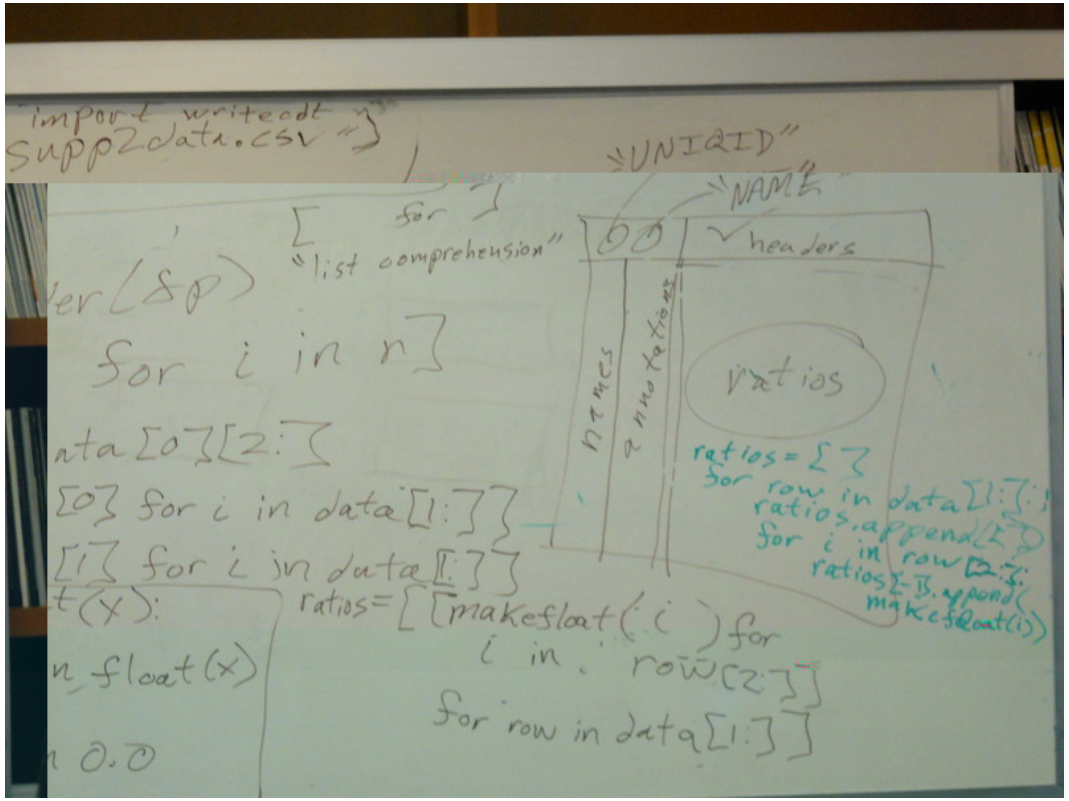
Whiteboard Image



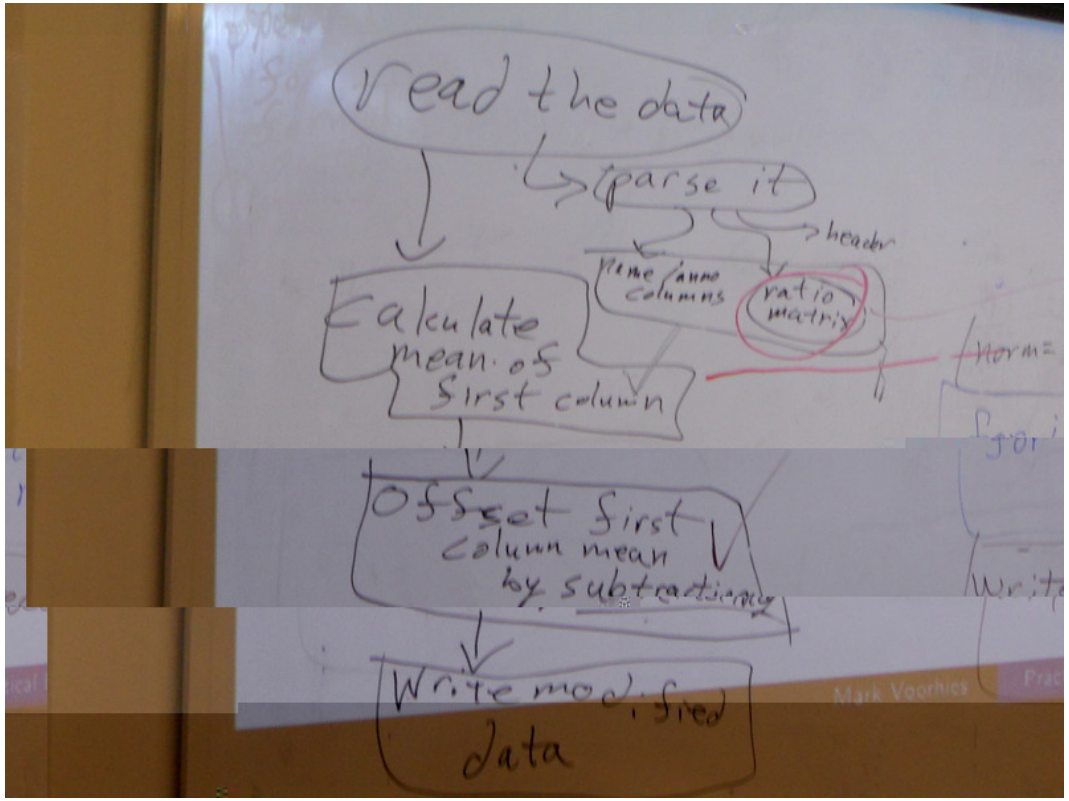
Whiteboard Image



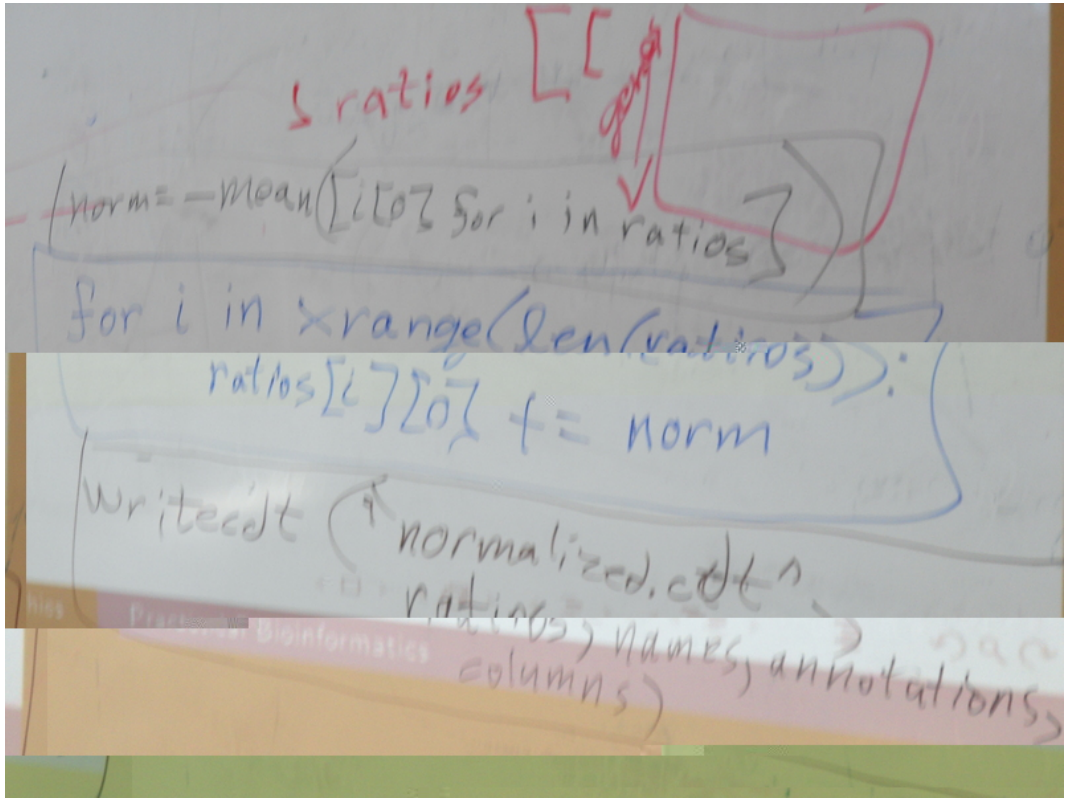
Whiteboard Image



Whiteboard Image

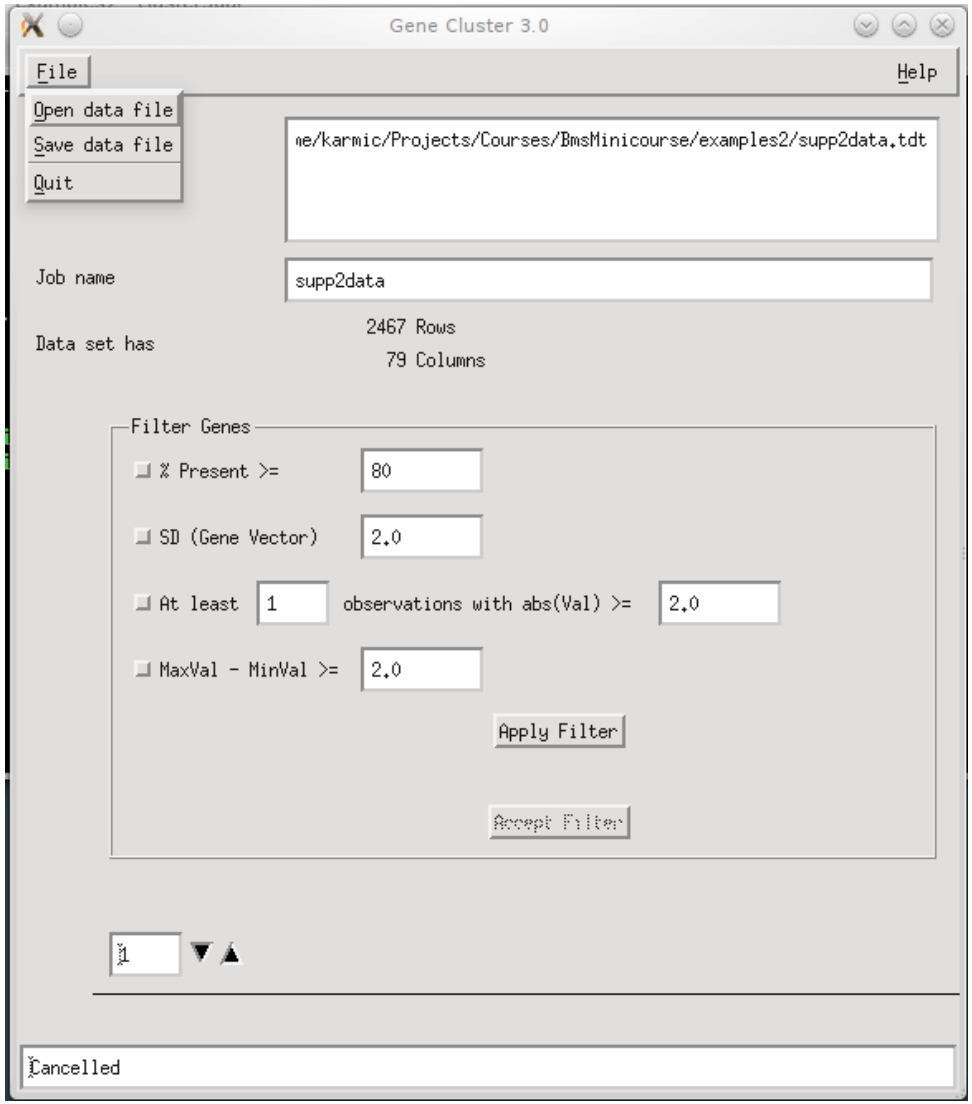


Whiteboard Image

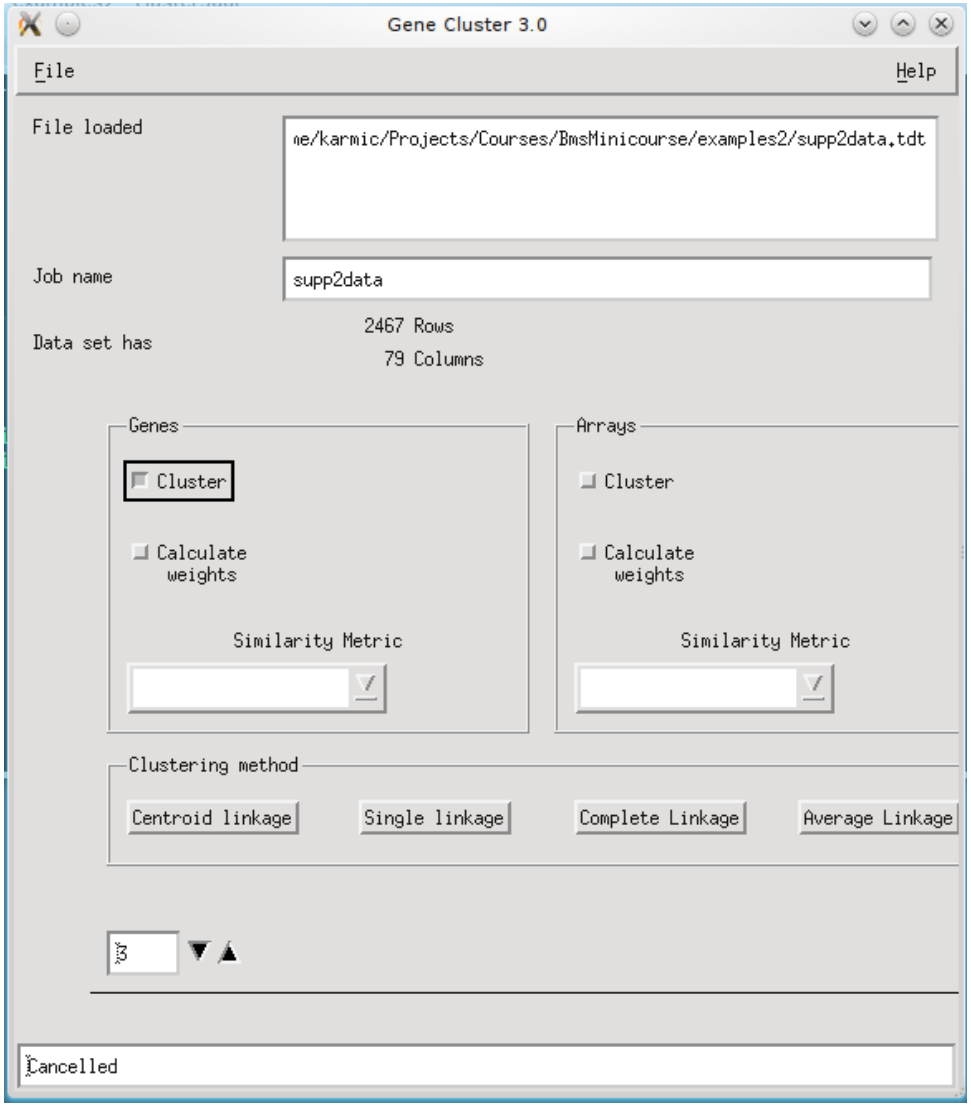


Using the Cluster3 GUI

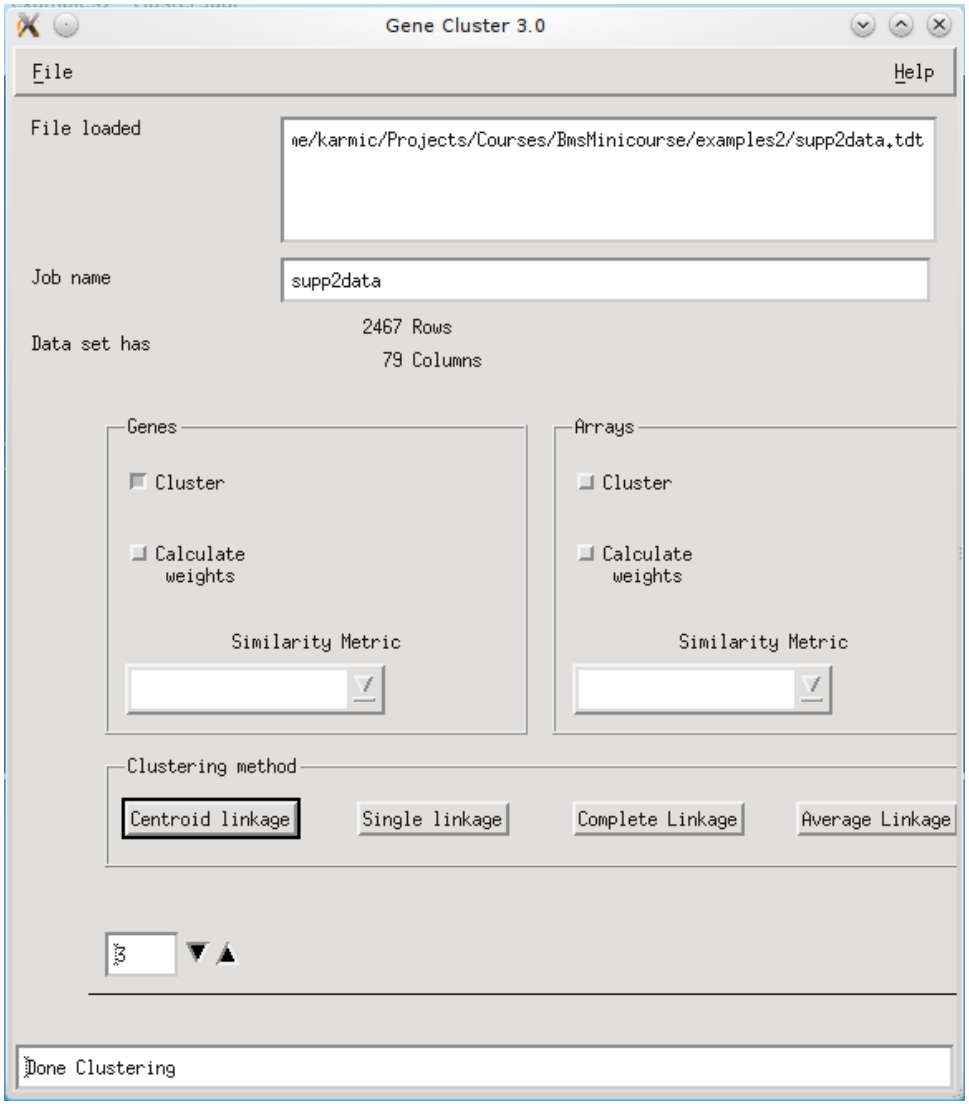
Load your data



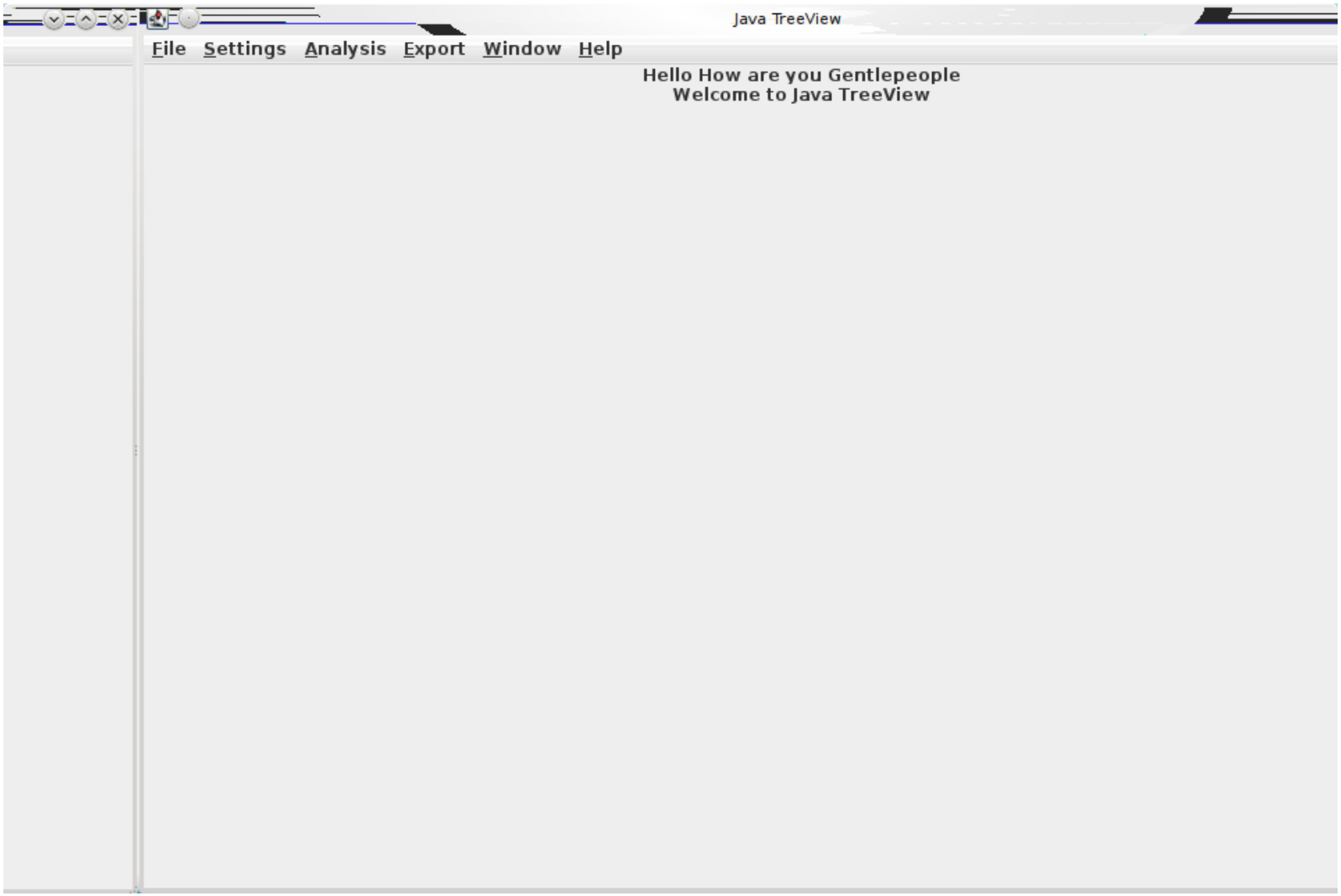
Choose distance function



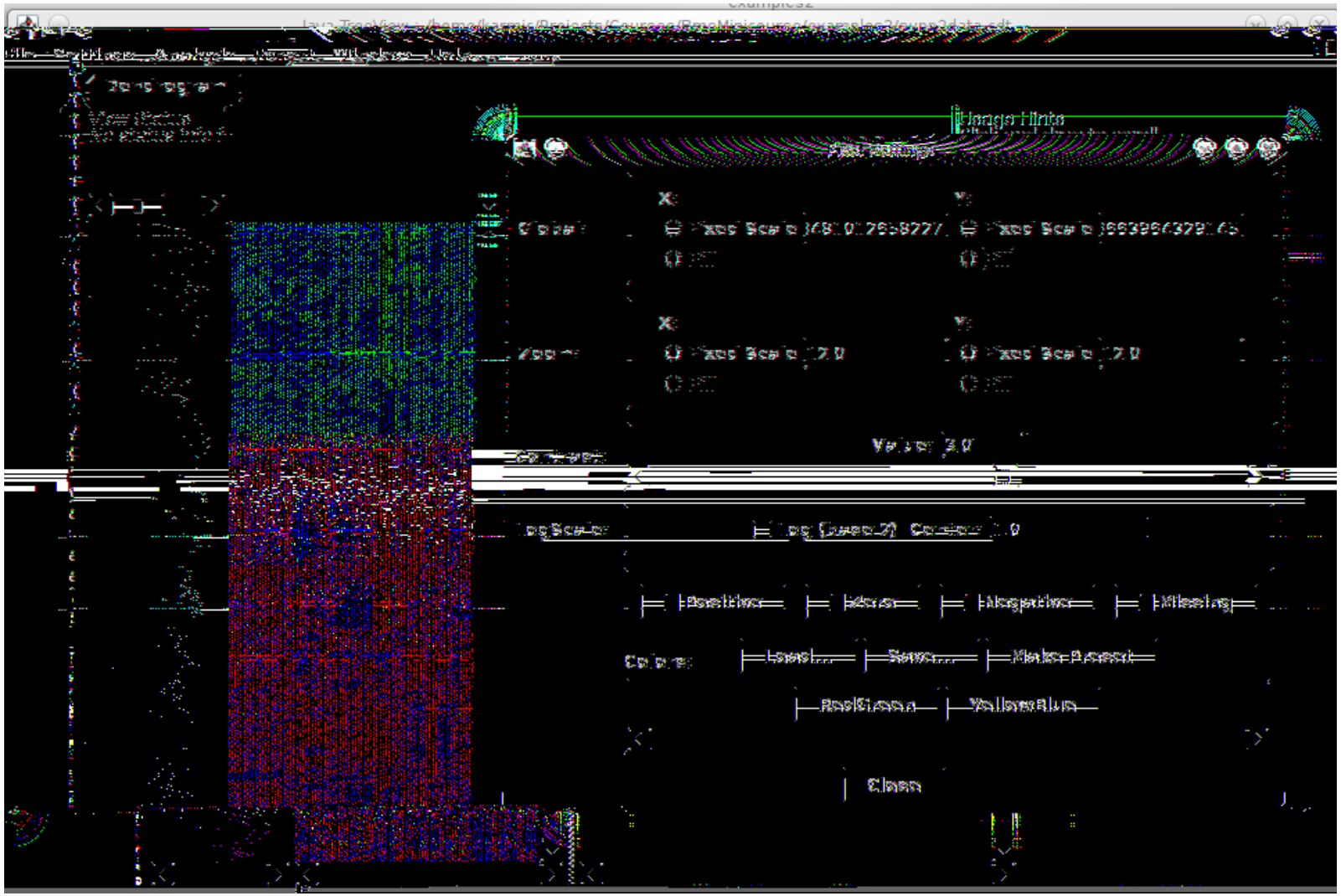
Choose linking method



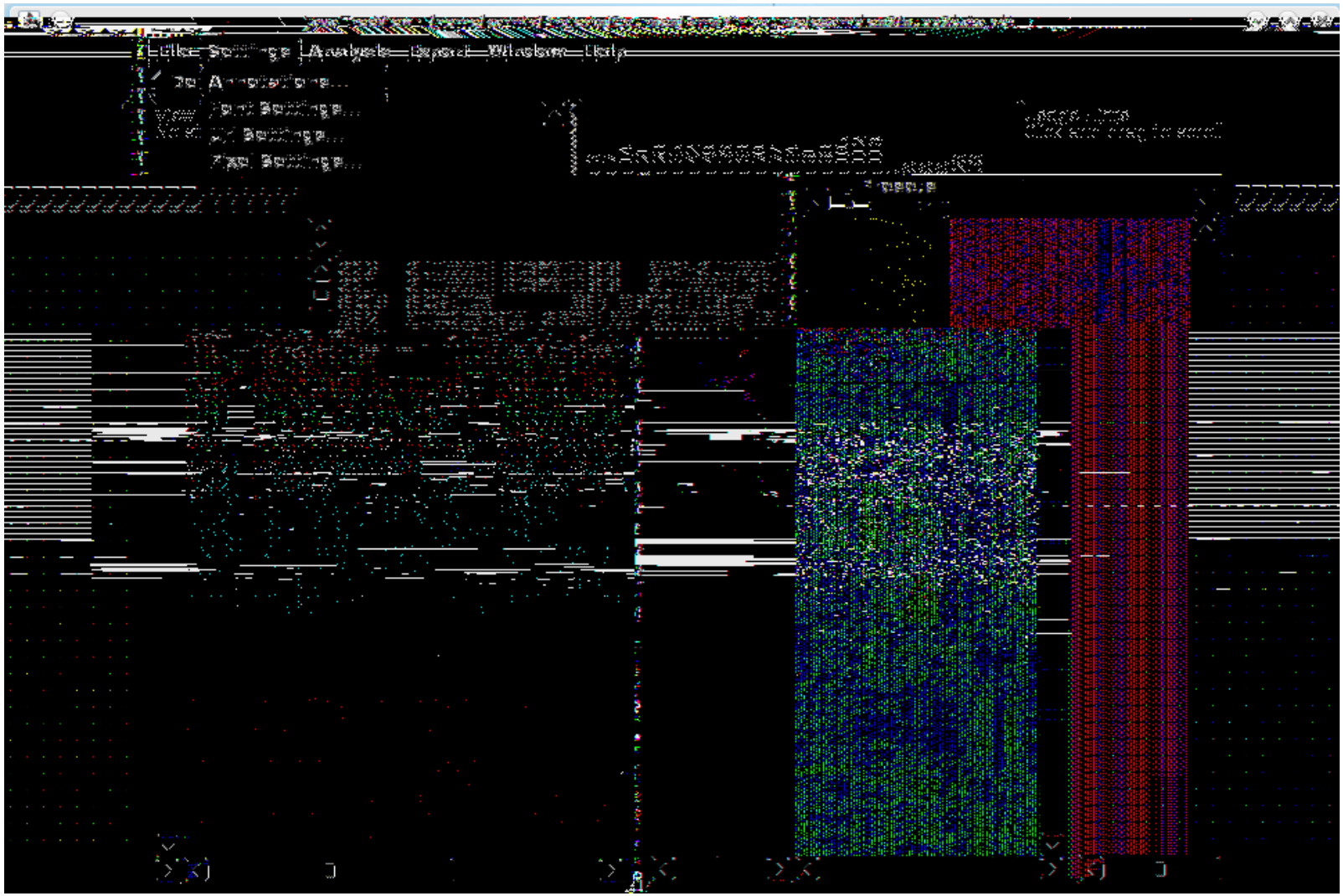
Using JavaTreeView



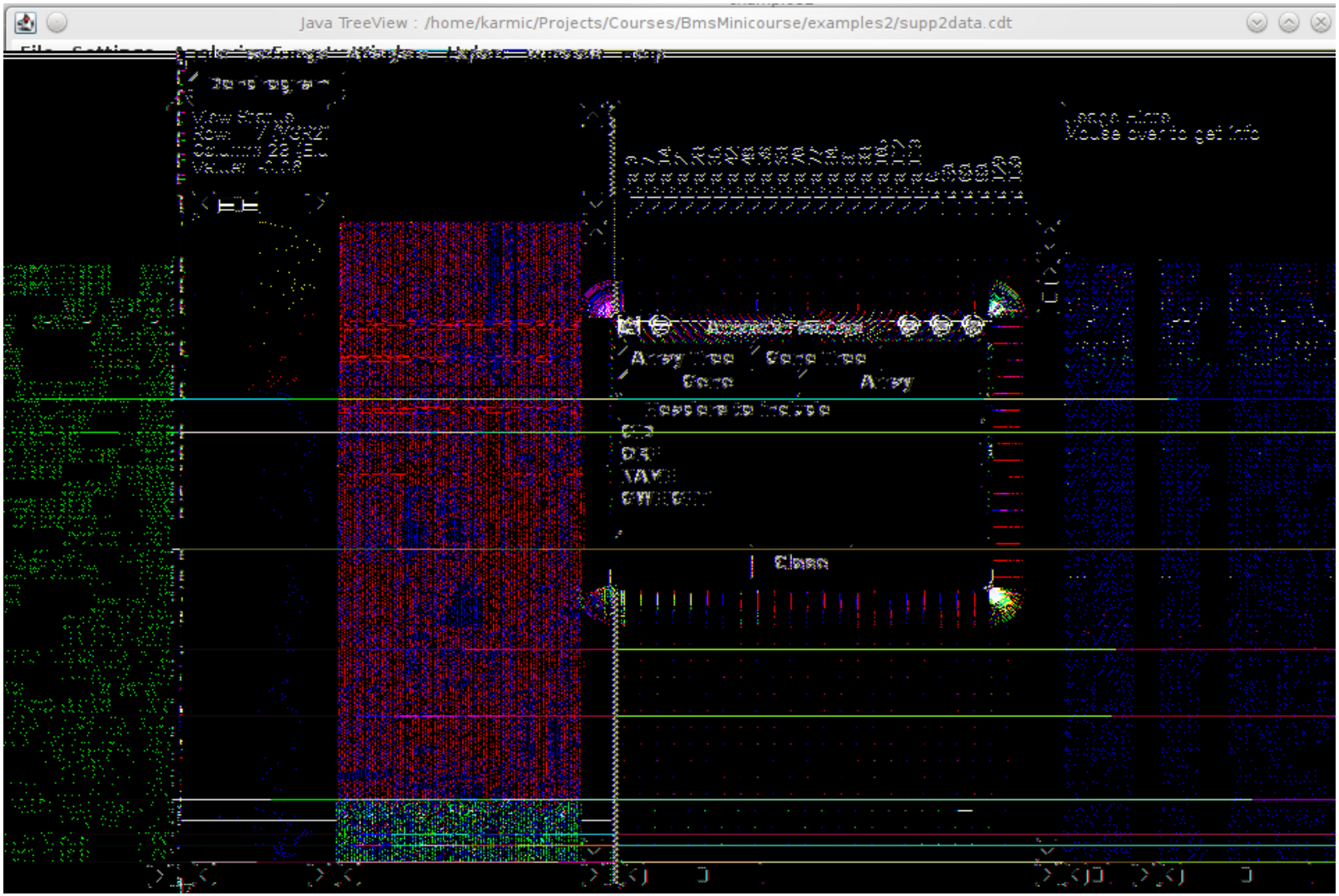
Adjust pixel settings for global view



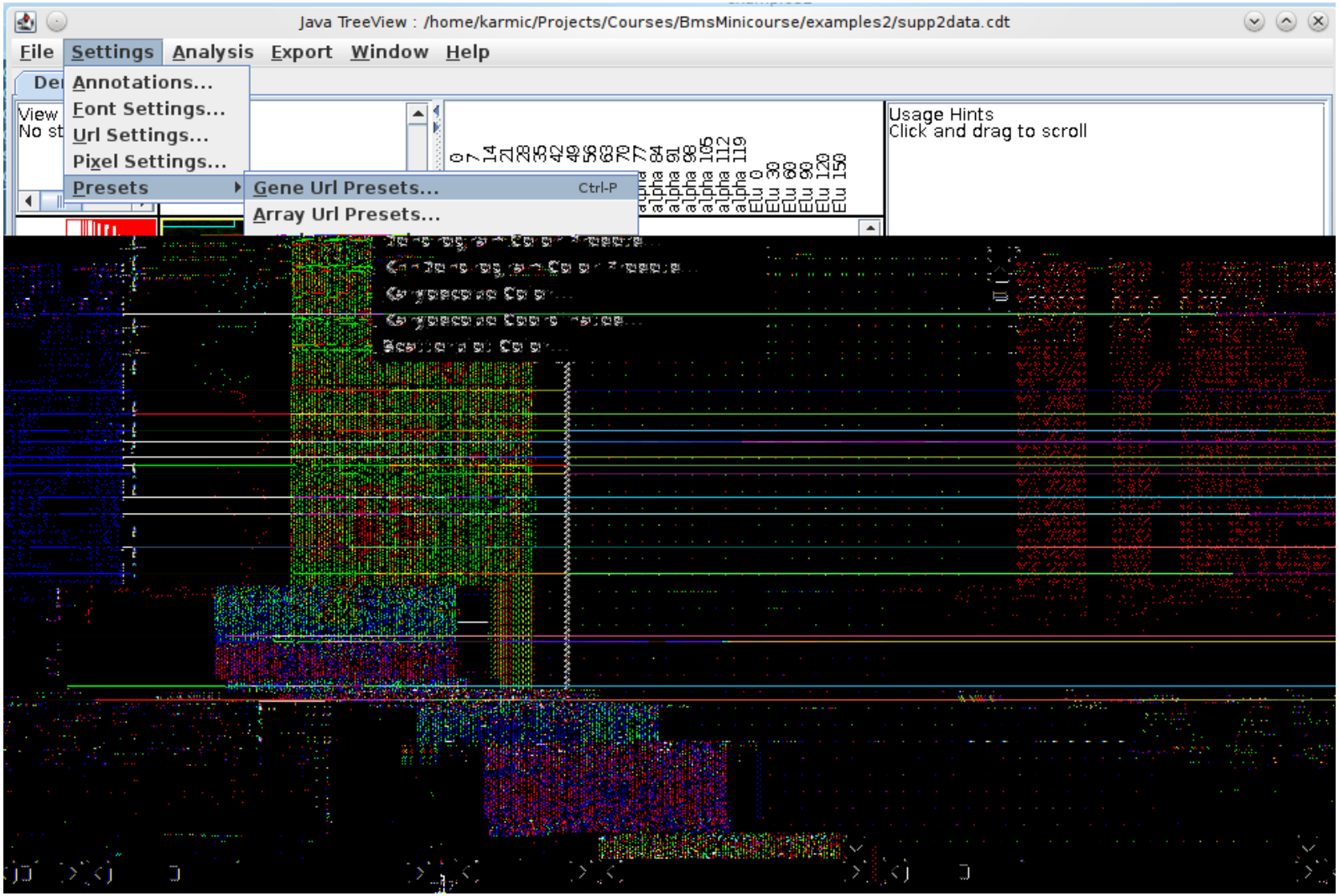
Select annotation columns



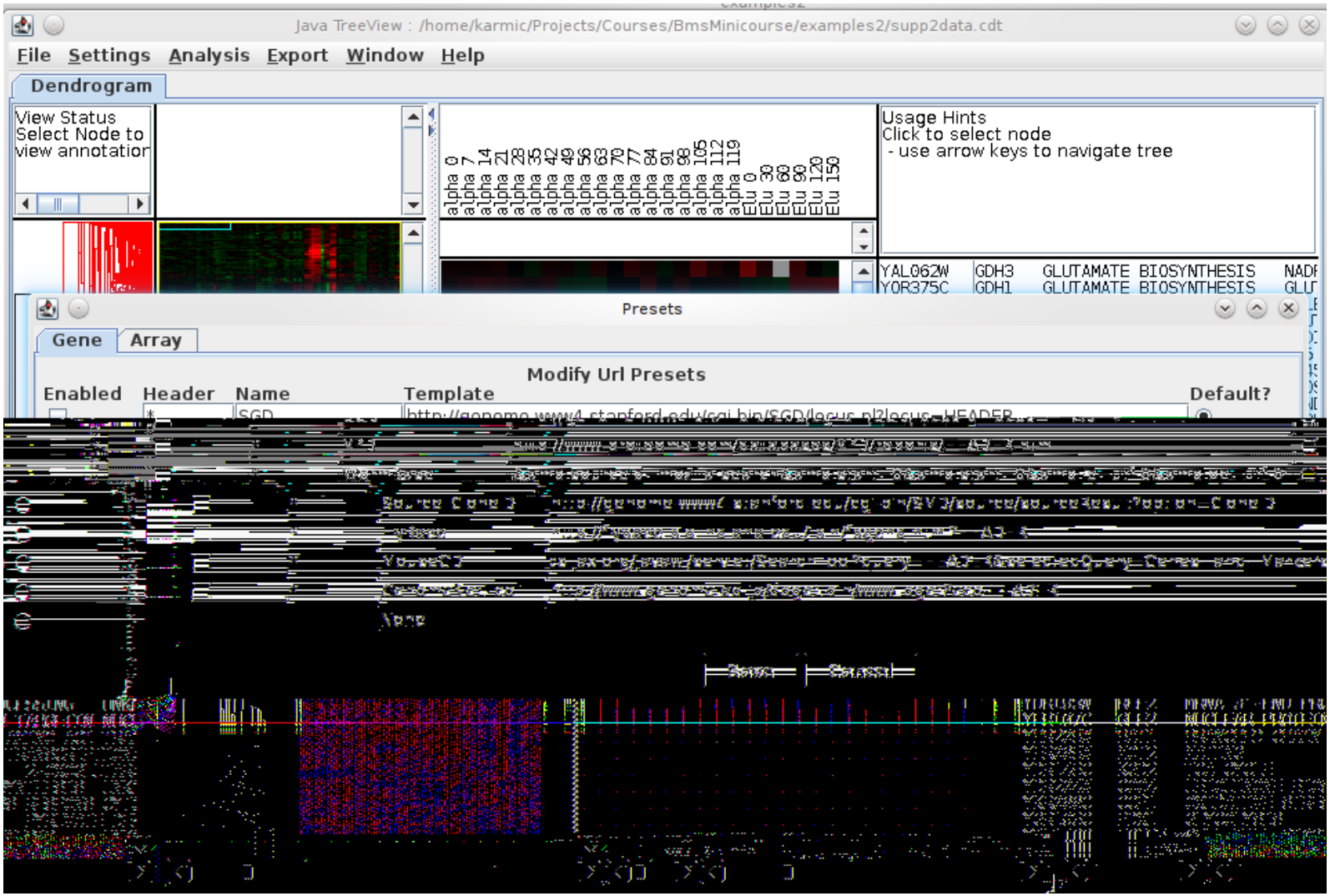
Select annotation columns



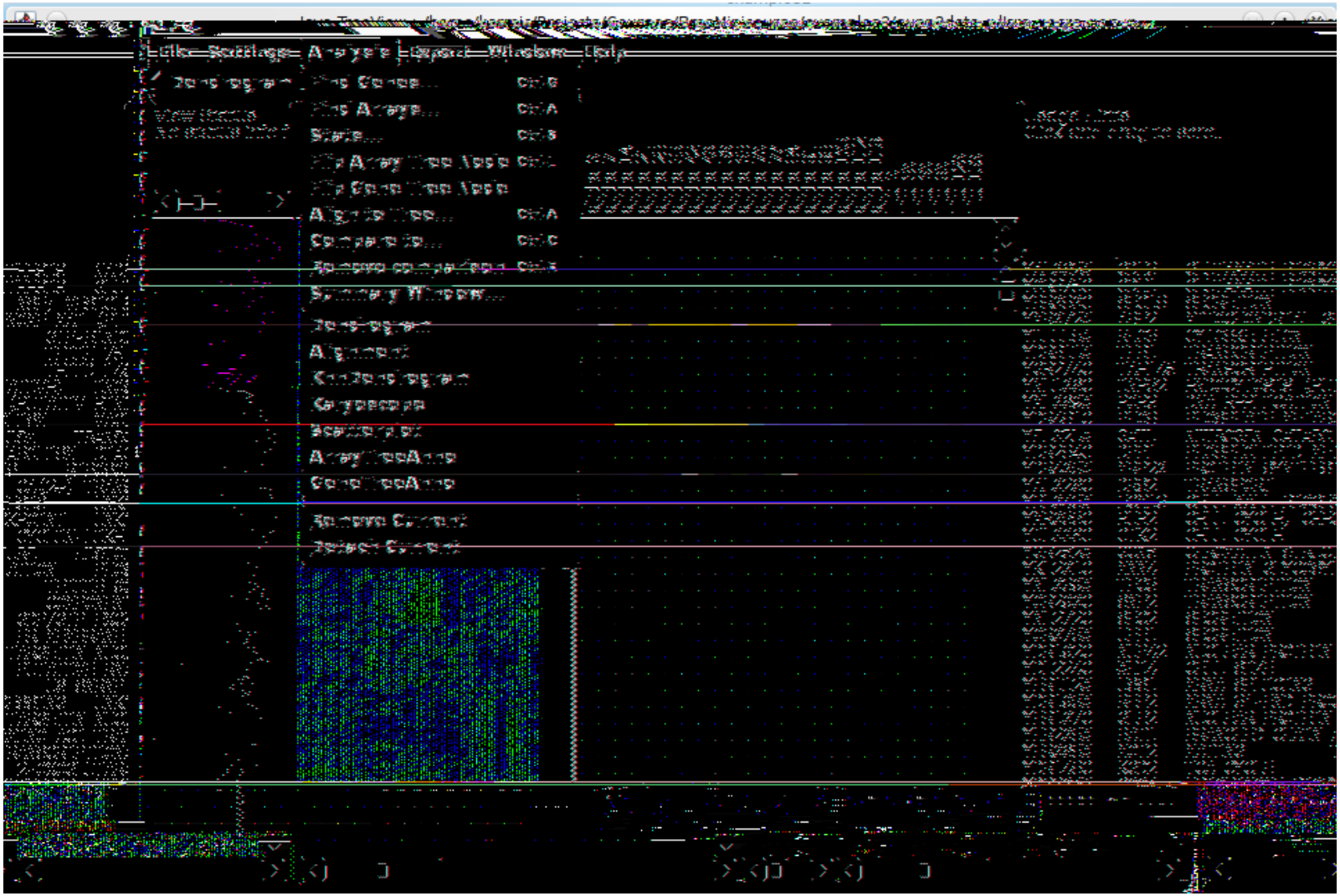
Select URL for gene annotations



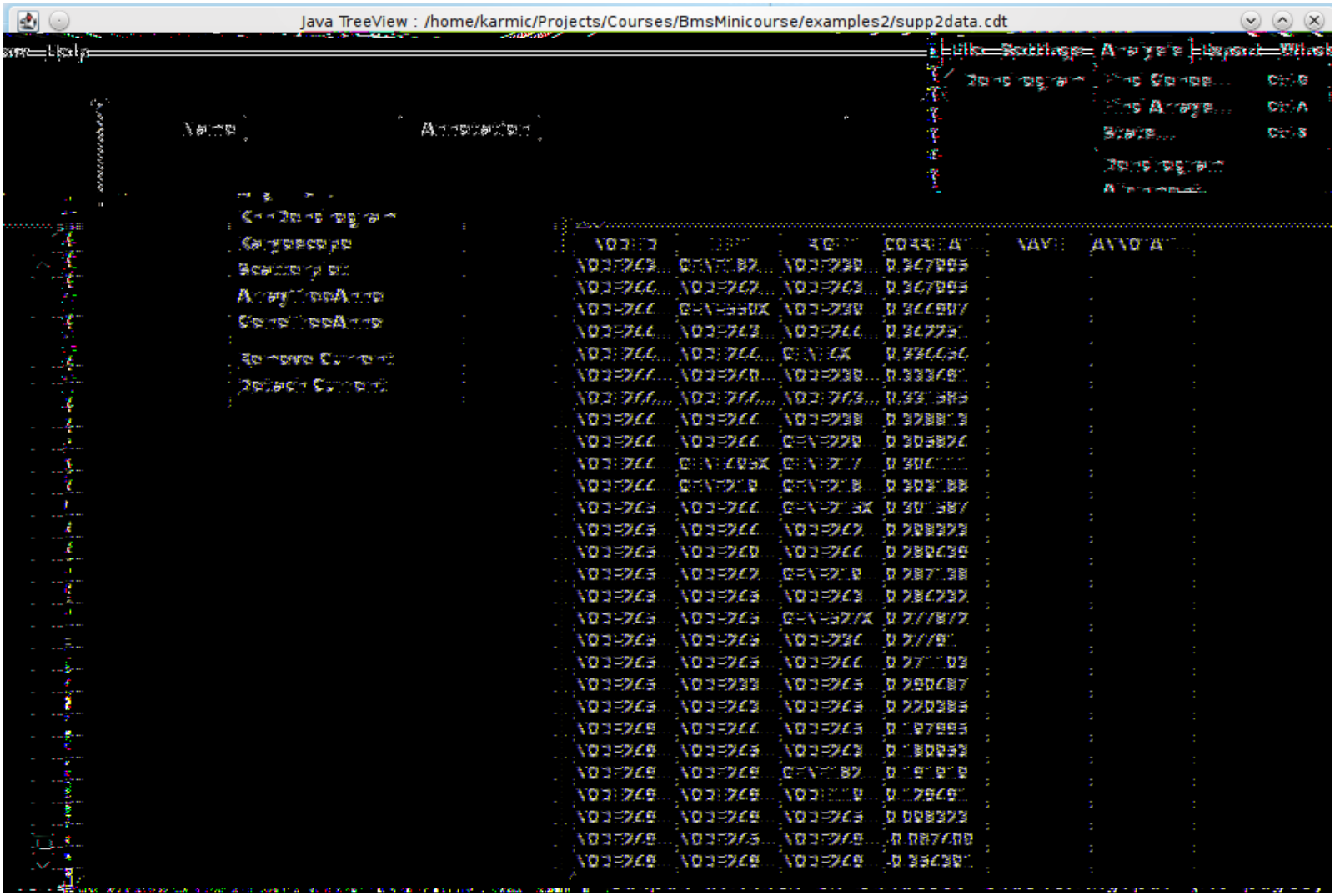
Select URL for gene annotations



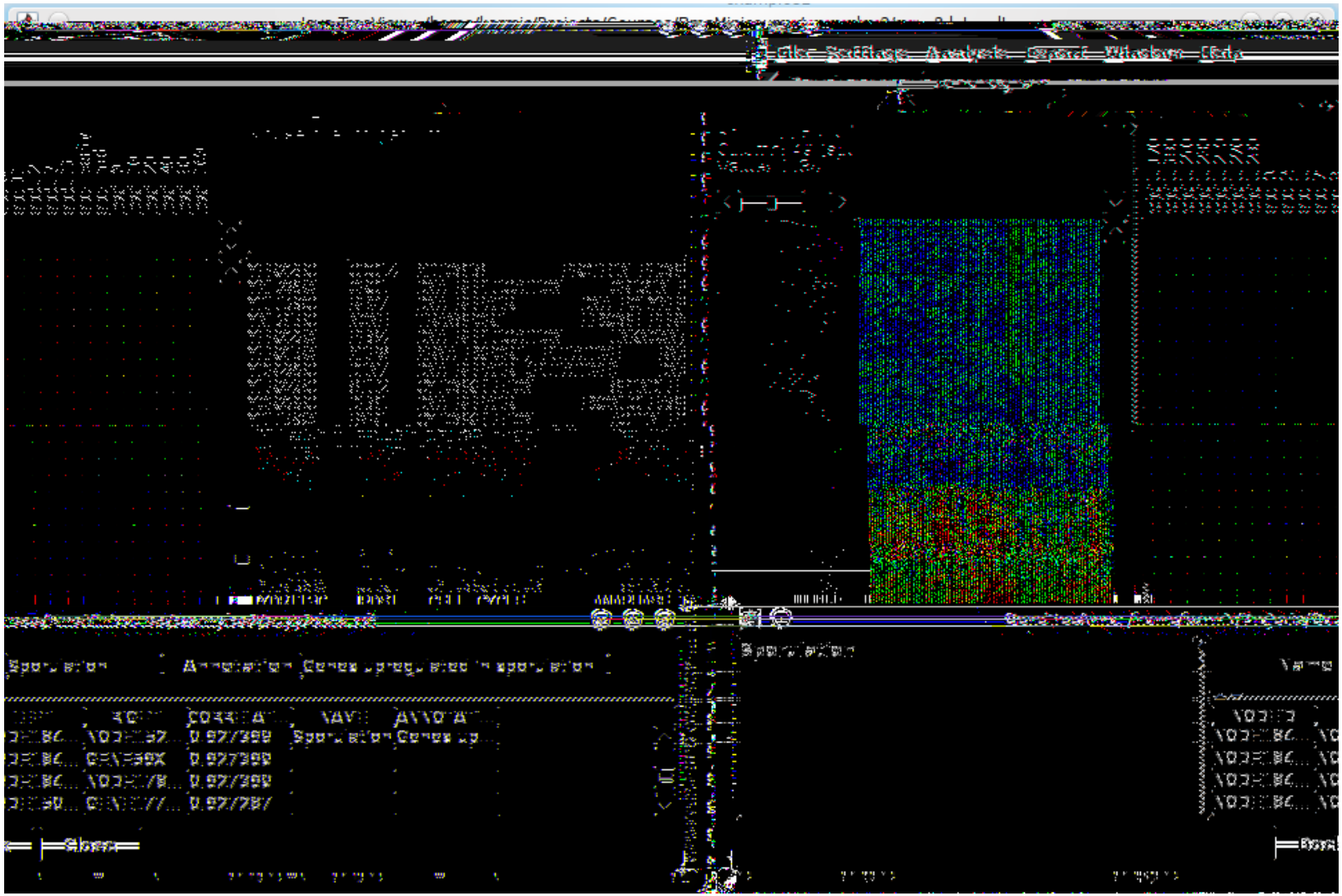
Activate and detach annotation window



Activate and detach annotation window



Activate and detach annotation window



Calculating the Distance Matrix

Pearson distances

Pearson similarity

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

Pearson distances

Pearson similarity

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

Pearson distance

$$d_{uncentered}(x,y) = 1 - s(x,y) \quad (2)$$

Pearson distances

Pearson similarity

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

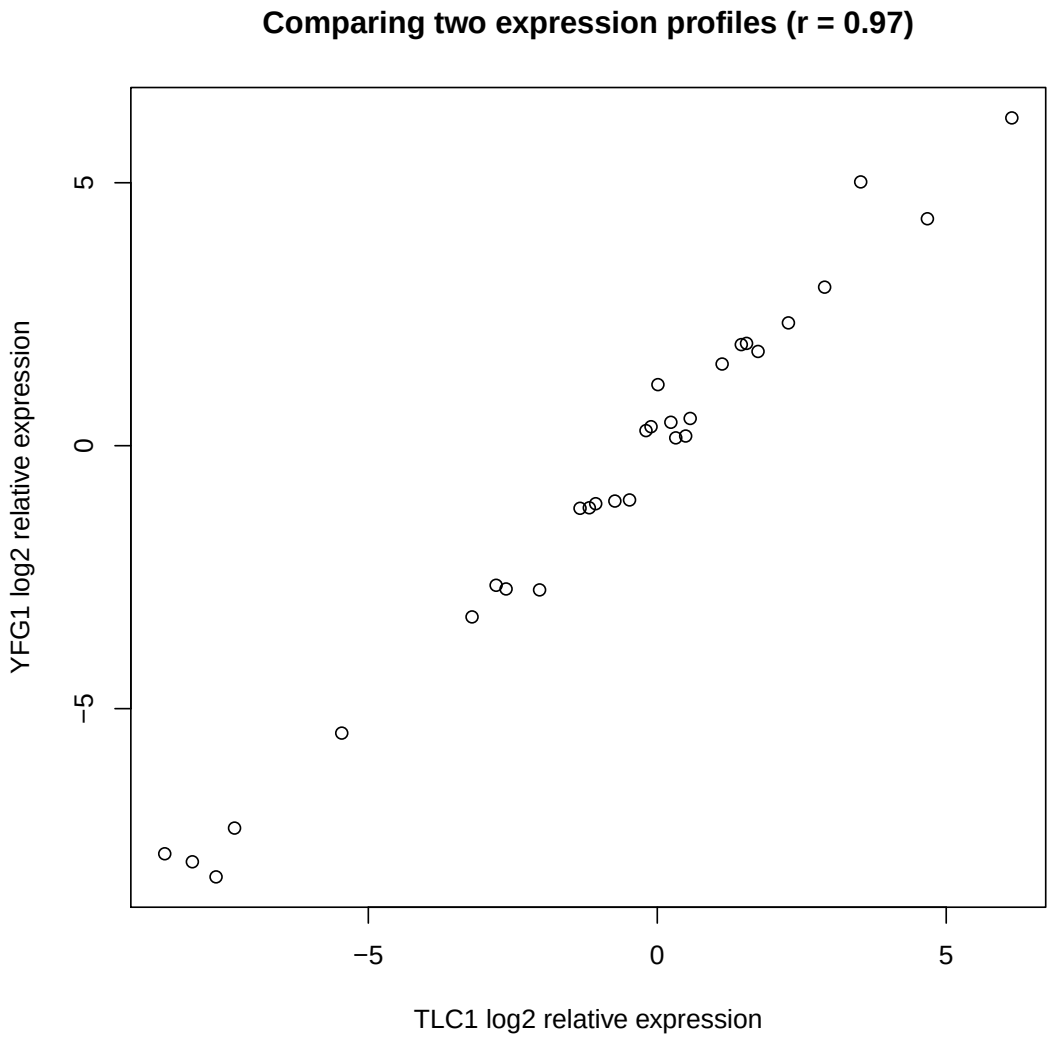
Pearson distance

$$d_{uncentered}(x,y) = 1 - s(x,y) \tag{2}$$

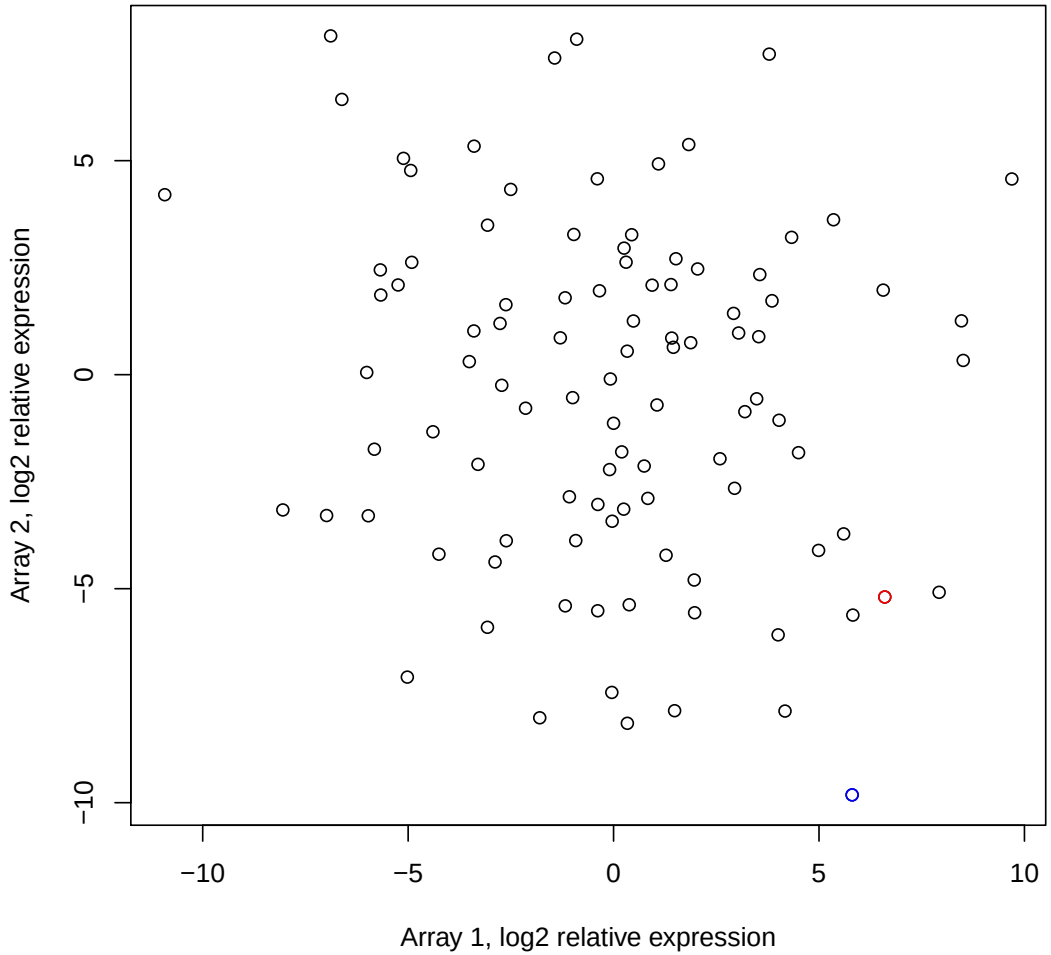
Euclidean distance

$$\frac{\sum_i^N (x_i - y_i)^2}{N} \tag{3}$$

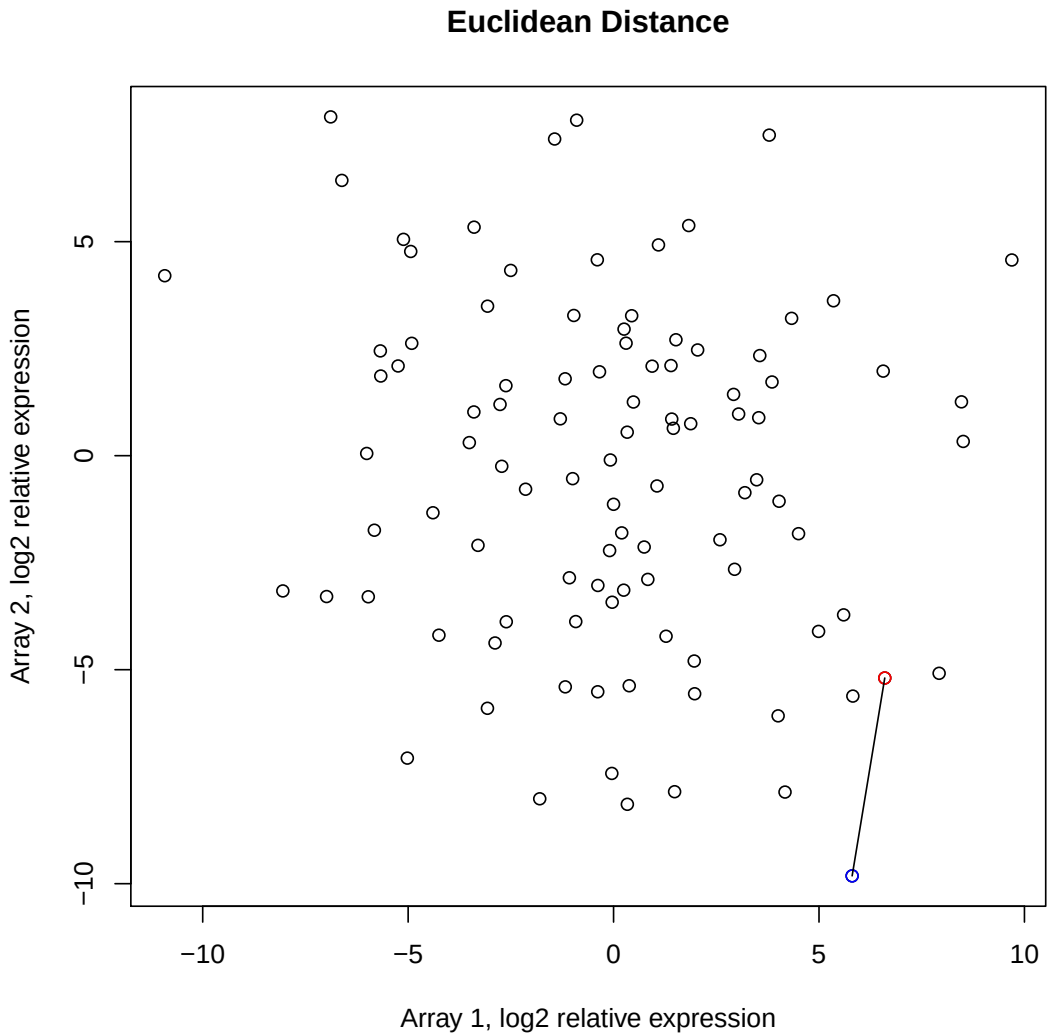
Comparing all measurements for two genes



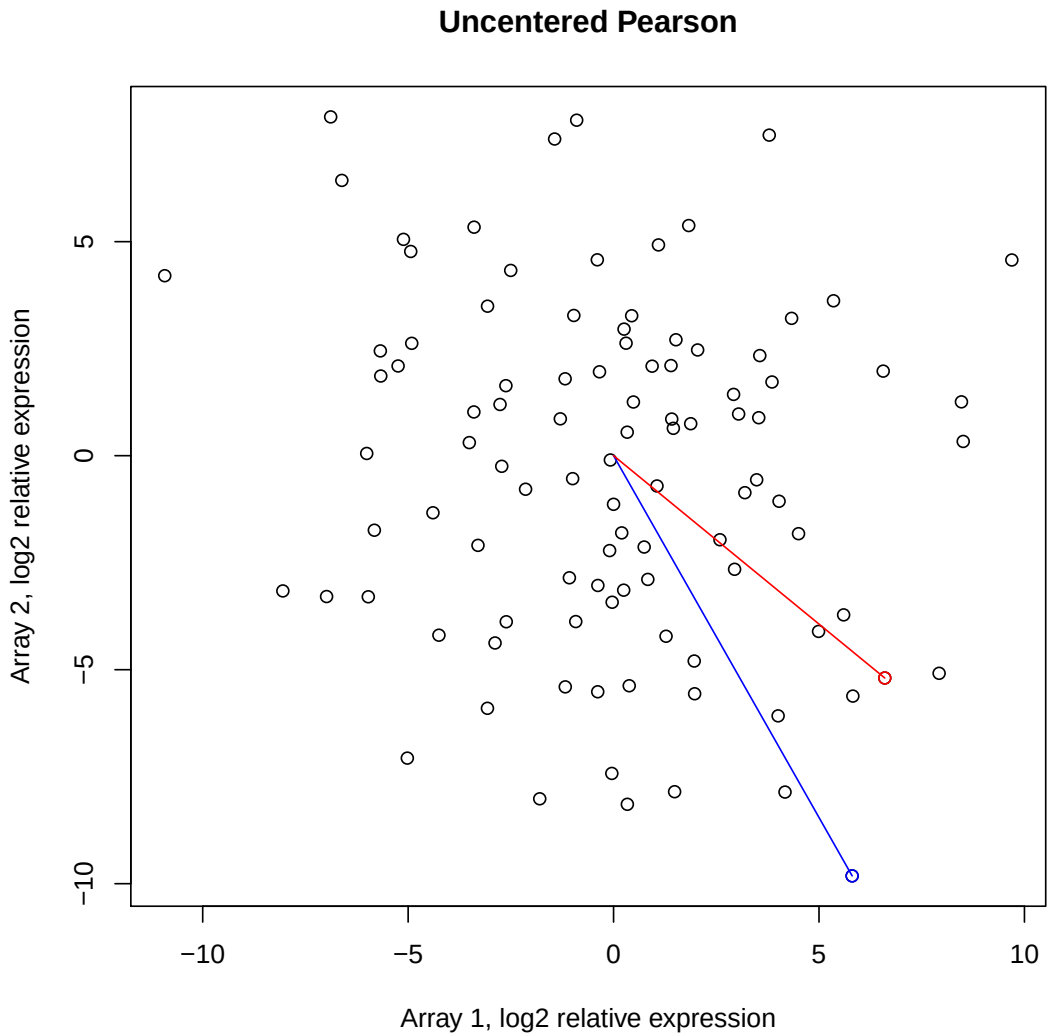
Comparing all genes for two measurements



Comparing all genes for two measurements



Comparing all genes for two measurements



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;
```