# Practical Bioinformatics

Mark Voorhies

6/17/2010

# Review

- Using lists to represent vectors
- Using lists of lists to represent matrices

```
a = []
for i in range(1,4):
    a.append([])
    for j in range(1,4):
        a[-1].append(i**j)
```

```
[[1, 1, 1],
 [2, 4, 8],
 [3, 9, 27]]
```

# Review

- Type conversion, exceptions, and None

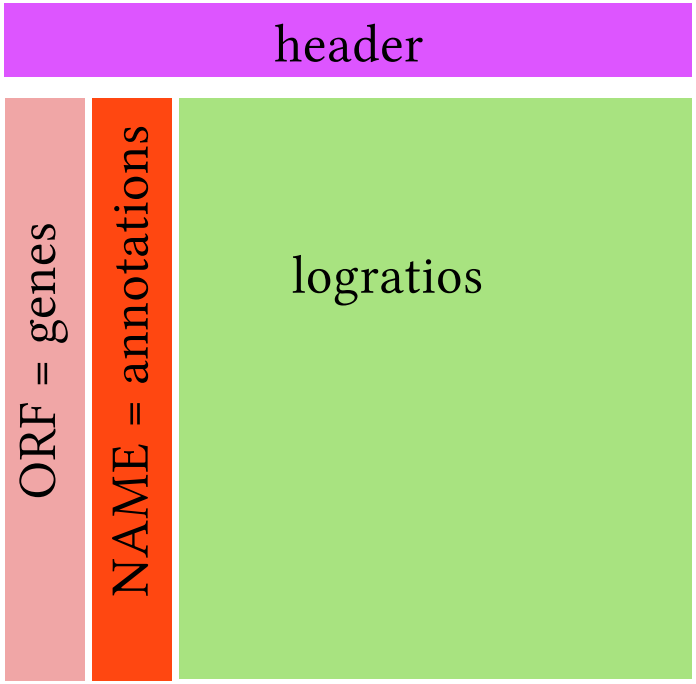
```
try:
    a = float("twenty")
except:
    a = None

if(i < 0):
    raise ValueError
```

# File exercises

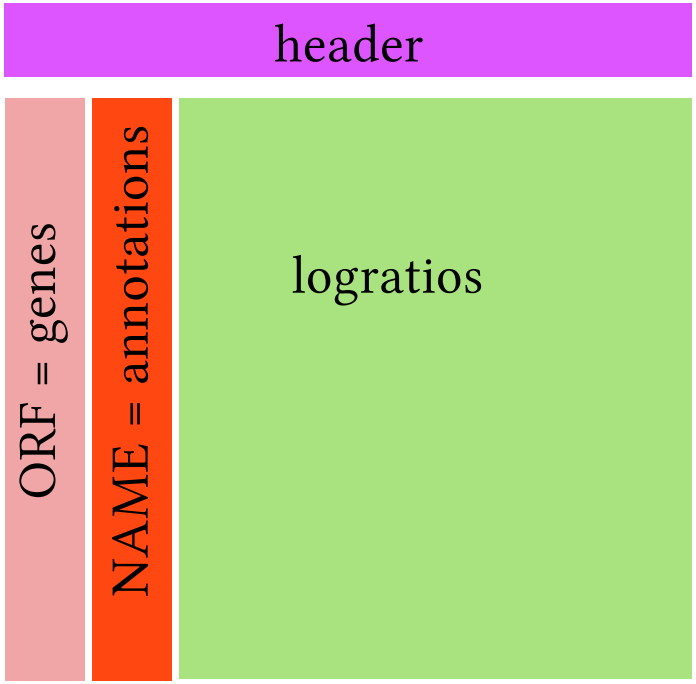
1. Write a function to read a file formatted like supp2data.tdt and return a list of the lines in the file.
2. Change the function to return a list of lists of fields from the file (*i.e.*, the equivalent of the table that you would see in a spreadsheet).
3. Change the function to return [genes, annotations, ratios] where:
   - genes is a list corresponding to the first column of the file
   - annotations is a list corresponding to the second column of the file
   - ratios is a list of lists corresponding to the matrix of ratios

   and the header line is omitted.
4. Update your function to return the ratio matrix as floating point numbers rather than strings. Empty cells should be set to 0.0 or None (your choice).

# supp2data.tdt

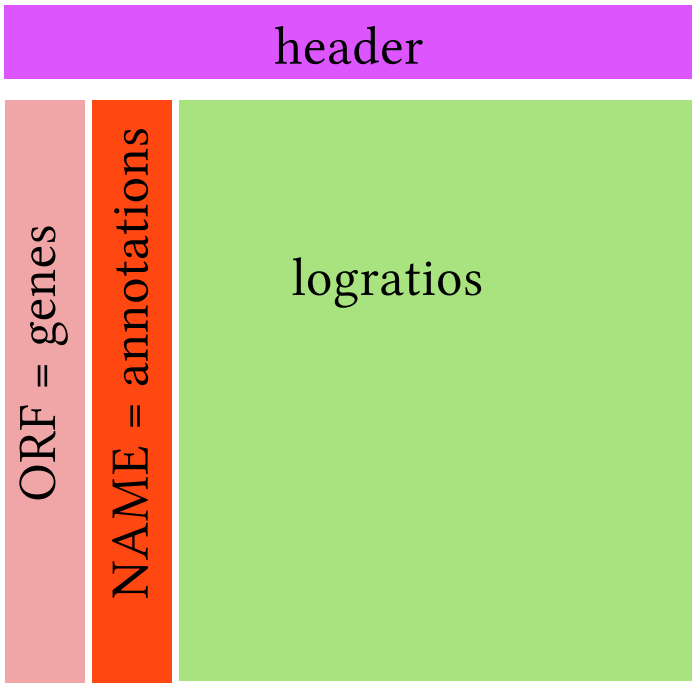


# supp2data.tdt

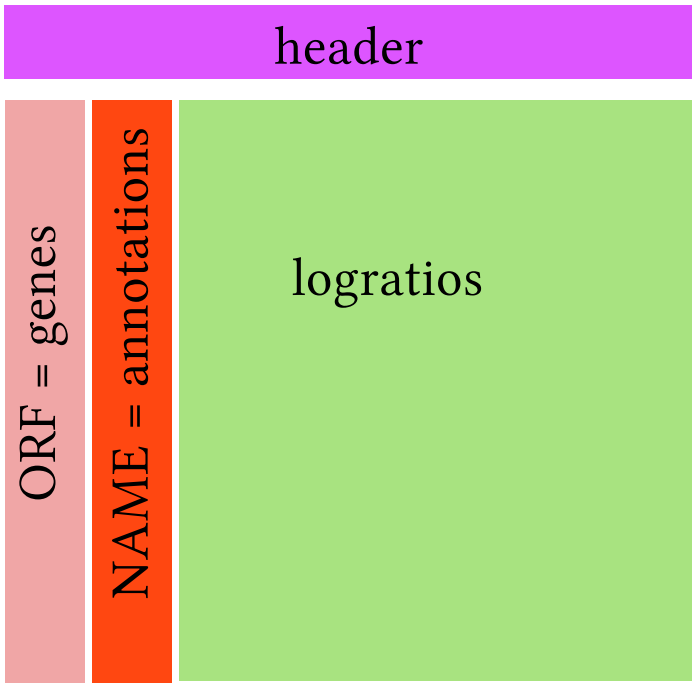



```
[["YBR166C", "YOR357C", "YLR292C", ...],

 ["TYR1 ...", "GRD19 ...", "SEC72 ...", ...],

 [[ 0.33, −0.17,  0.04, −0.07, −0.09, ...],
  [−0.64, −0.38, −0.32, −0.29, −0.22, ...],
  [−0.23,  0.19, −0.36,  0.14, −0.40, ...],
  ...]
]
```

# A file parsing function

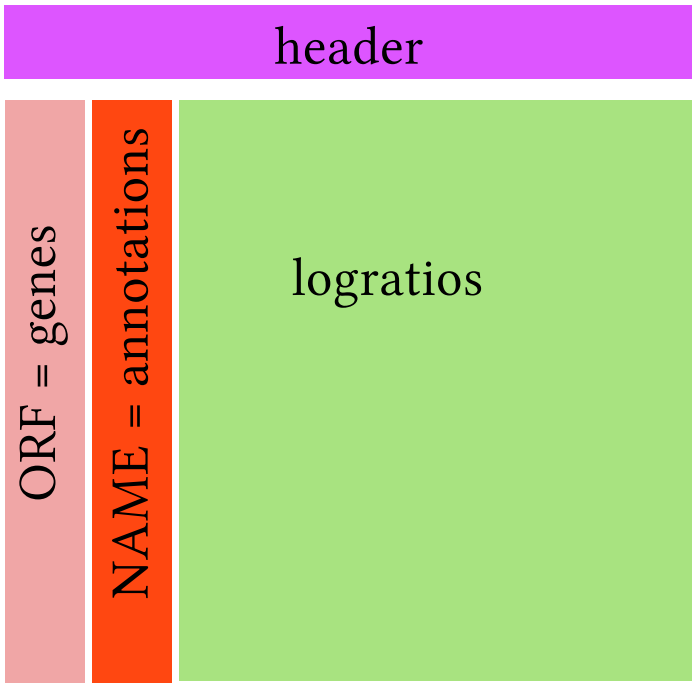


# A file parsing function




```
def parseTdt(filename):
    # Initialize return values
    genes = []
    annotations = []
    ratios = []
    # Open file and burn header line
    fp = open(filename)
    header = fp.readline()
    # Parse each data line
    for line in fp:
        fields = line.split("\t")
        genes.append(fields[0])
        annotations.append(fields[1])
        # Add a new row to the ratios matrix
        ratios.append([])
        for ratio in fields[2:]:
            try:
                value = float(ratio)
            except:
                value = None
            # Add the parsed value
            # to the current row
            ratios[-1].append(value)
    return [genes, annotations, ratios]
```

# A file parsing class




```
class TdtRatios:
    def __init__(self, filename):
        # Initialize member values
        self.genes = []
        self.annotations = []
        self.ratios = []
        # Open file and burn header line
        fp = open(filename)
        self.header = fp.readline()
        # Parse each data line
        for line in fp:
            fields = line.split("\t")
            self.genes.append(fields[0])
            self.annotations.append(fields[1])
            # Add a new row to the ratios matrix
            self.ratios.append([])
            for ratio in fields[2:]:
                try:
                    value = float(ratio)
                except:
                    value = None
                # Add the parsed value
                # to the current row
                self.ratios[-1].append(value)
```

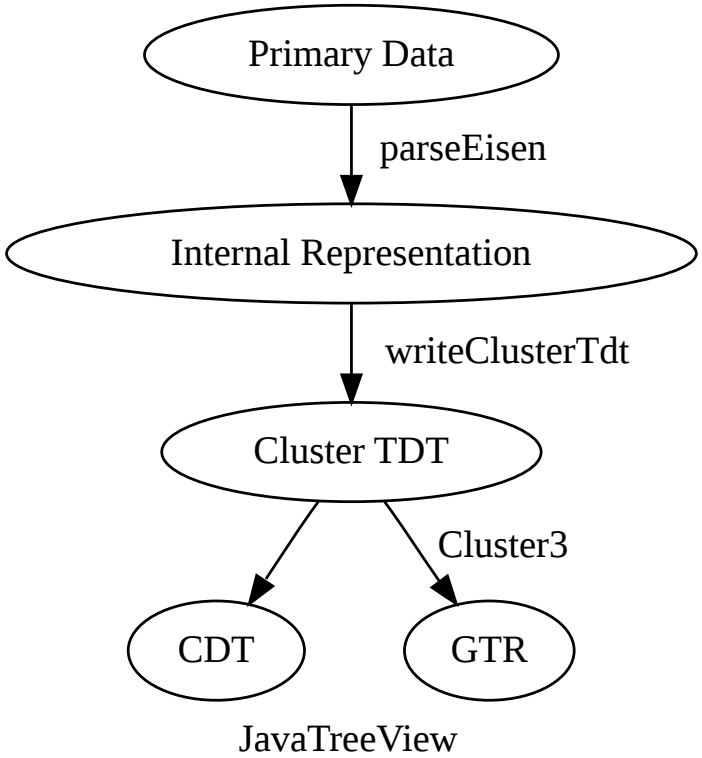
# Distance functions

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

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

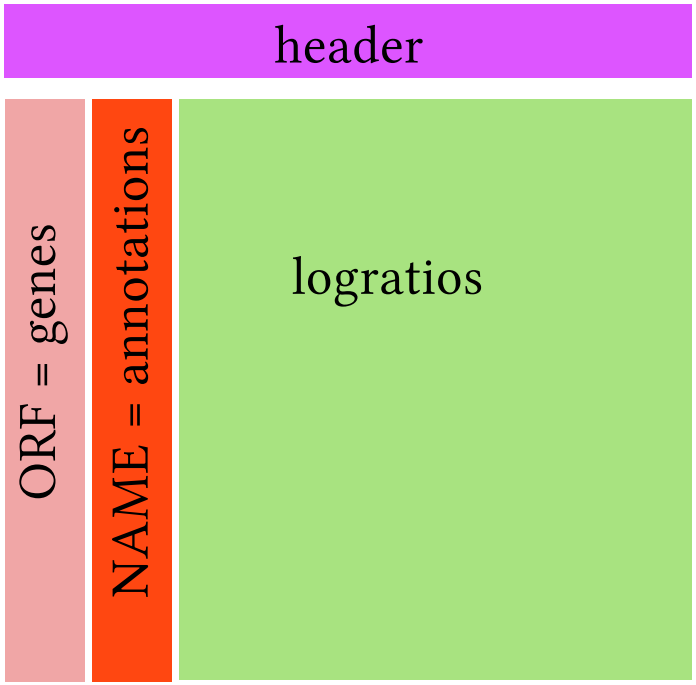
2. Amend the function to calculate the centered Pearson (or another distance metric from the Cluster3 manual)
3. Write a function to calculate all pairwise distances for the yeast expression profiles for a particular distance function.
4. Save the results of your pairwise distance calculation in the CDT format described in the JavaTreeView manual.

# Clustering protocol

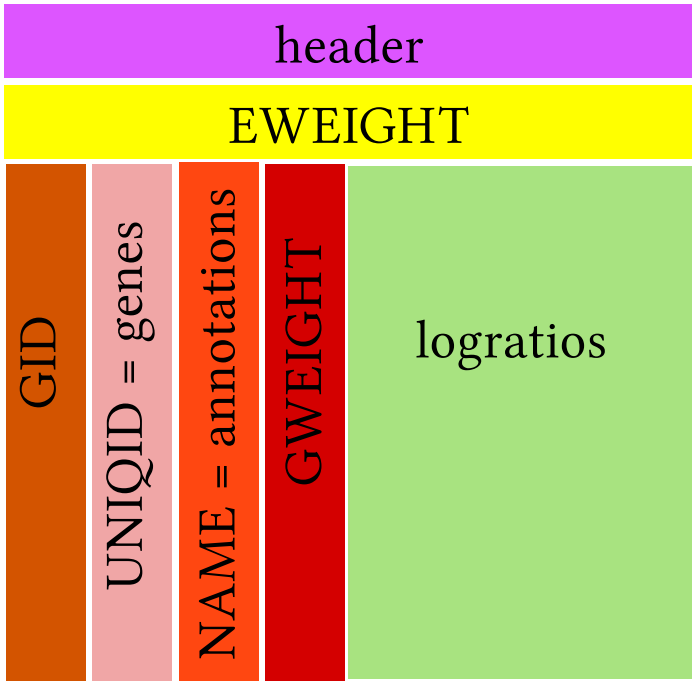


# The CDT file format

supp2data.tdt




Cluster CDT output




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

# Cluster3

- CLUSTER TDT file format (section 2.1)
- Cluster3 GUI
- Loading data
- Hierarchical clustering

# JavaTreeView

- CDT file format (pg. 19)
- Pixel settings
- Annotation settings
- URL presets
- GTR file format (pg. 21)
- Gene tree annotations