Practical Bioinformatics

Mark Voorhies

4/20/2011

Review

- Corrected slides for days 1 and 2 are on the website

Review

- Corrected slides for days 1 and 2 are on the website
- Each byte in a text file can be translated as a human-readable character.

Review

- Corrected slides for days 1 and 2 are on the website
- Each byte in a text file can be translated as a human-readable character. The different operating systems differ only in how they mark the end of a line:
  - UNIX: \n (linefeed)
  - MacOS: \r (carriage return)
  - Windows: \r \n (CRLF)

Review

- Corrected slides for days 1 and 2 are on the website
- Each byte in a text file can be translated as a human-readable character. The different operating systems differ only in how they mark the end of a line:
  - UNIX: \n (linefeed)
  - MacOS: \r (carriage return)
  - Windows: \r \n (CRLF)

  Here's how to fix the \r problem on OS X:

  tr '\r' '\n' < macfile.txt > unixfile.txt

Review

- Corrected slides for days 1 and 2 are on the website
- Each byte in a text file can be translated as a human-readable character. The different operating systems differ only in how they mark the end of a line:
  - UNIX: \n (linefeed)
  - MacOS: \r (carriage return)
  - Windows: \r \n (CRLF)

  Here's how to fix the \r problem on OS X:

  tr '\r' '\n' < macfile.txt > unixfile.txt
- Loading and re-loading your modules

```
# 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)
```

Gotchas

- If the same value will be referred to many times, save it in a variable

```
def stdev(x):
    m = mean(x)
    s = 0.0
    for i in x:
        s += (i - m)**2
    from math import sqrt
    return sqrt(s/(len(x) - 1))
```

Gotchas

- If the same value will be referred to many times, save it in a variable

```
def stdev(x):
    m = mean(x)
    s = 0.0
    for i in x:
        s += (i - m)**2
    from math import sqrt
    return sqrt(s/(len(x) - 1))
```

- Use list comprehensions for vector operations

```
v = [3,2,4,5,1]
a = v**2                  # no
a = [i**2 for i in v] # yes
```

Gotchas

- If the same value will be referred to many times, save it in a variable

```
def stdev(x):
    m = mean(x)
    s = 0.0
    for i in x:
        s += (i - m)**2
    from math import sqrt
    return sqrt(s/(len(x) - 1))
```

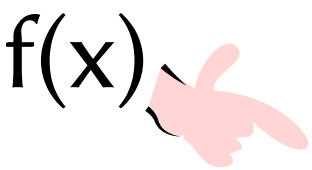
- Use list comprehensions for vector operations

```
v = [3,2,4,5,1]
a = v**2                  # no
a = [i**2 for i in v] # yes
```

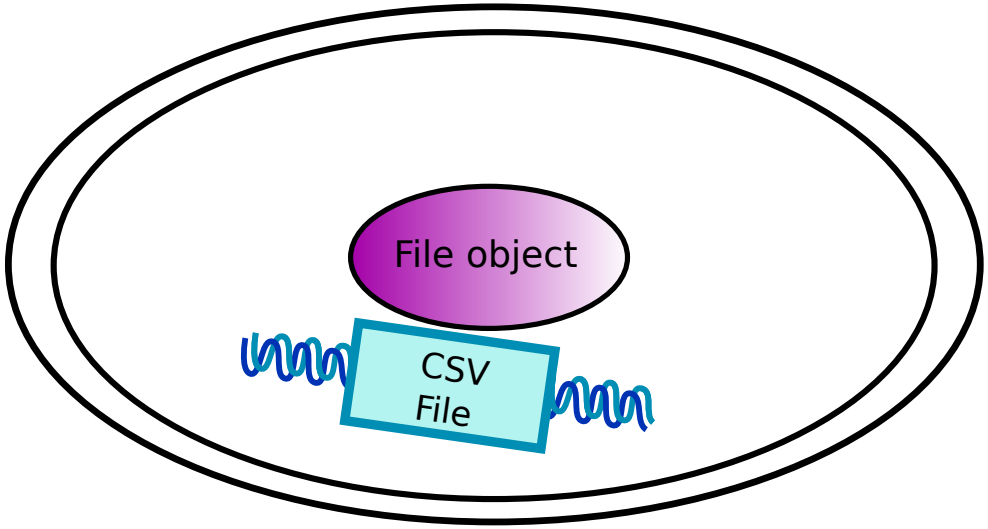
$$\frac{v}{\sqrt{\sum_{i=1}^{|v|} v_i^2}}$$

```
norm = sqrt(sum(i**2 for i in v))
unit = [i/norm for i in v]
```

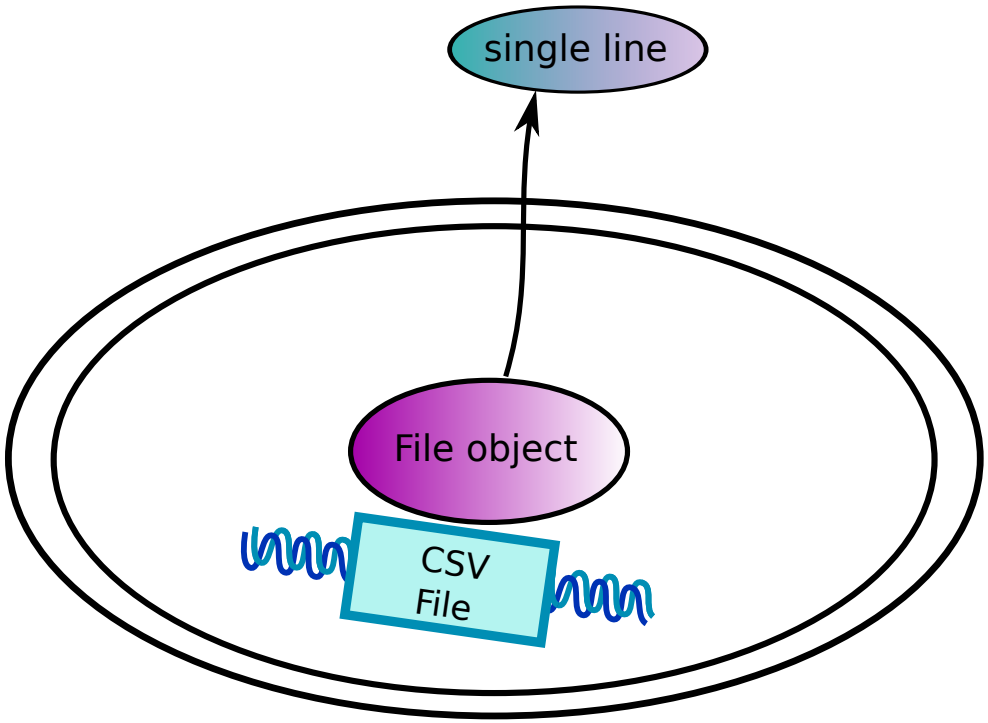
Object oriented programming



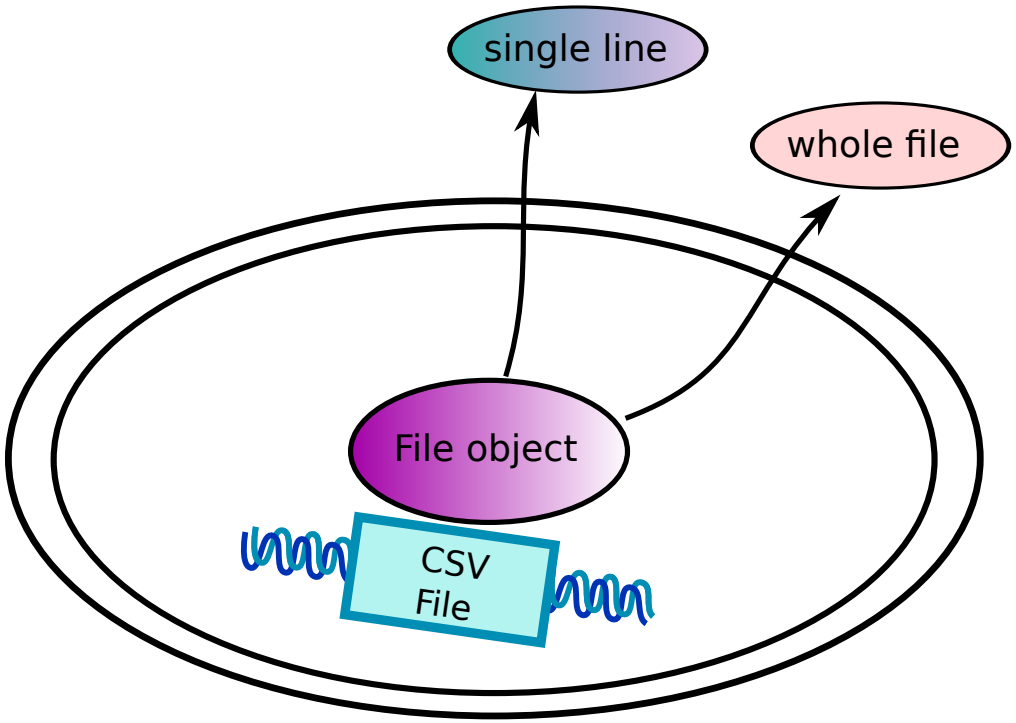
open("supp2data.csv")



open("supp2data.csv").next()



open("supp2data.csv").read()



Coercing data

Converting between data types:

```
number = float("3.5")
text = str(3.5)
```

Coercing data

Converting between data types:

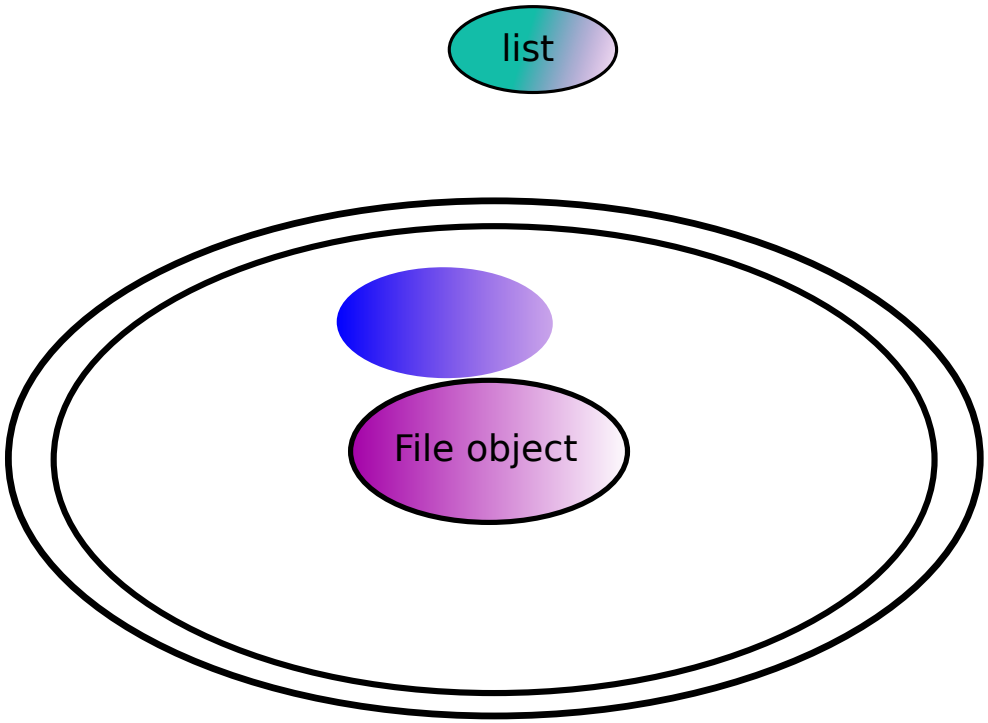
```
number = float("3.5")
text = str(3.5)
```

Catching exceptions:

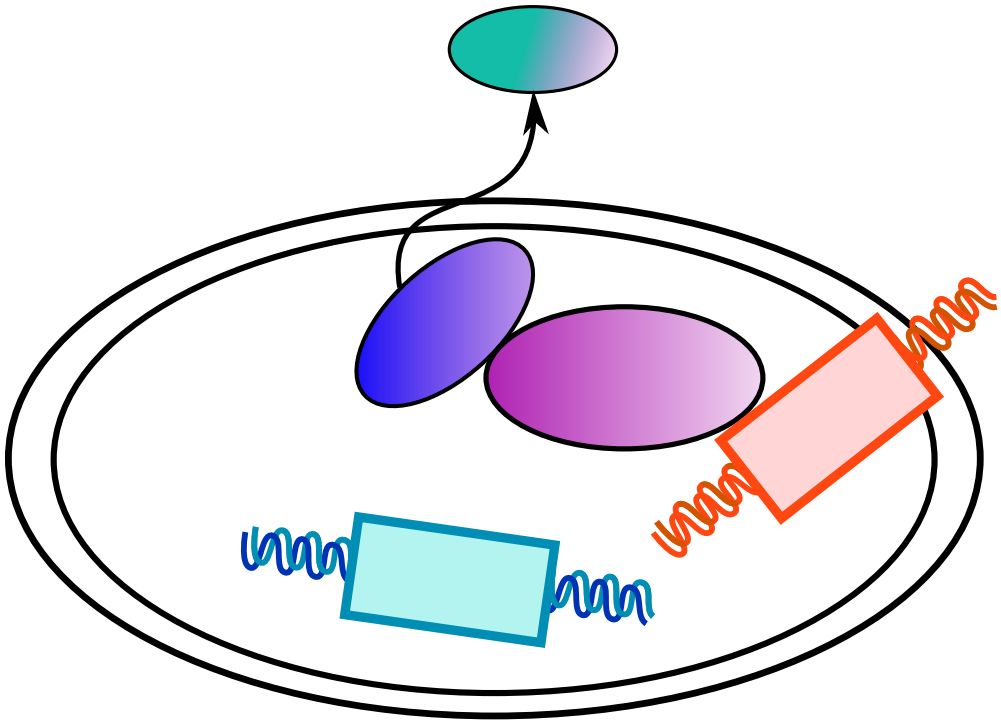
```
try:
    a = float("twenty")
except:
    a = None

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

csv.reader(open("supp2data.csv")).next()

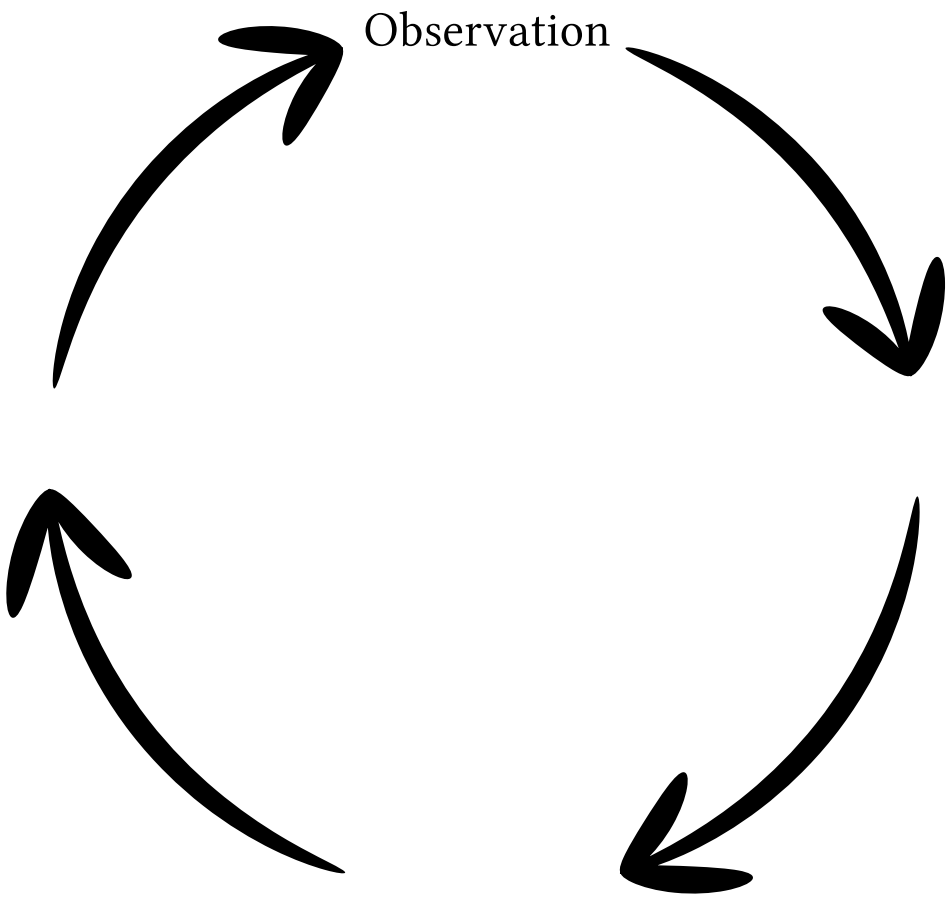


csv.reader(urlopen("http://example.com/csv")).next()



Expression Profiling Workflow

Expression Profiling Analysis



Pearson distances

Pearson similarity

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

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

Pearson distances

Pearson similarity

$$s(x,y) = \sum_{i}^{N} \left(\frac{x_i - x_{offset}}{\phi_x} \right) \left(\frac{y_i - y_{offset}}{\phi_y} \right) \tag{3}$$

$$\phi_G = \sqrt{\sum_{i}^{N} (G_i - G_{offset})^2} \tag{4}$$

Pearson distances

Pearson similarity

$$s(x,y) = \sum_{i}^{N} \left(\frac{x_i - x_{offset}}{\sqrt{\sum_i^N (x_i - x_{offset})^2}} \right) \left(\frac{y_i - y_{offset}}{\sqrt{\sum_i^N (y_i - y_{offset})^2}} \right) \quad (5)$$

Pearson distances

Pearson similarity

$$s(x,y) = \frac{\sum_i^N (x_i - x_{offset})(y_i - y_{offset})}{\sqrt{\sum_i^N (x_i - x_{offset})^2}\sqrt{\sum_i^N (y_i - y_{offset})^2}} \quad (6)$$

Pearson distances

Pearson similarity

$$s(x,y) = \frac{\sum_i^N (x_i - x_{offset})(y_i - y_{offset})}{\sqrt{\sum_i^N (x_i - x_{offset})^2}\sqrt{\sum_i^N (y_i - y_{offset})^2}} \tag{6}$$

Pearson distance

$$d_{uncentered}(x,y) = 1 - s(x,y) \tag{7}$$

Pearson distances

Pearson similarity

$$s(x,y) = \frac{\sum_i^N (x_i - x_{offset})(y_i - y_{offset})}{\sqrt{\sum_i^N (x_i - x_{offset})^2}\sqrt{\sum_i^N (y_i - y_{offset})^2}} \tag{6}$$

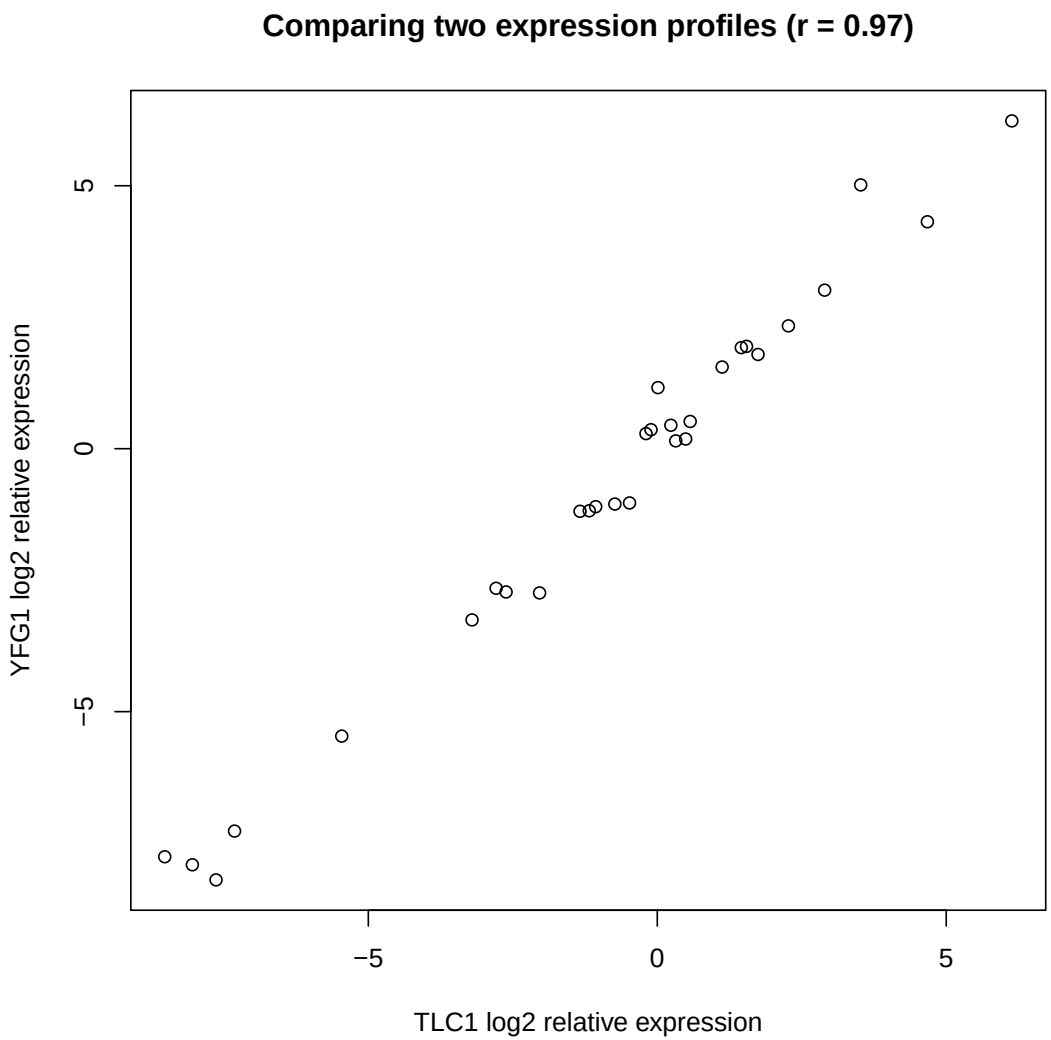
Pearson distance

$$d_{uncentered}(x,y) = 1 - s(x,y) \tag{7}$$

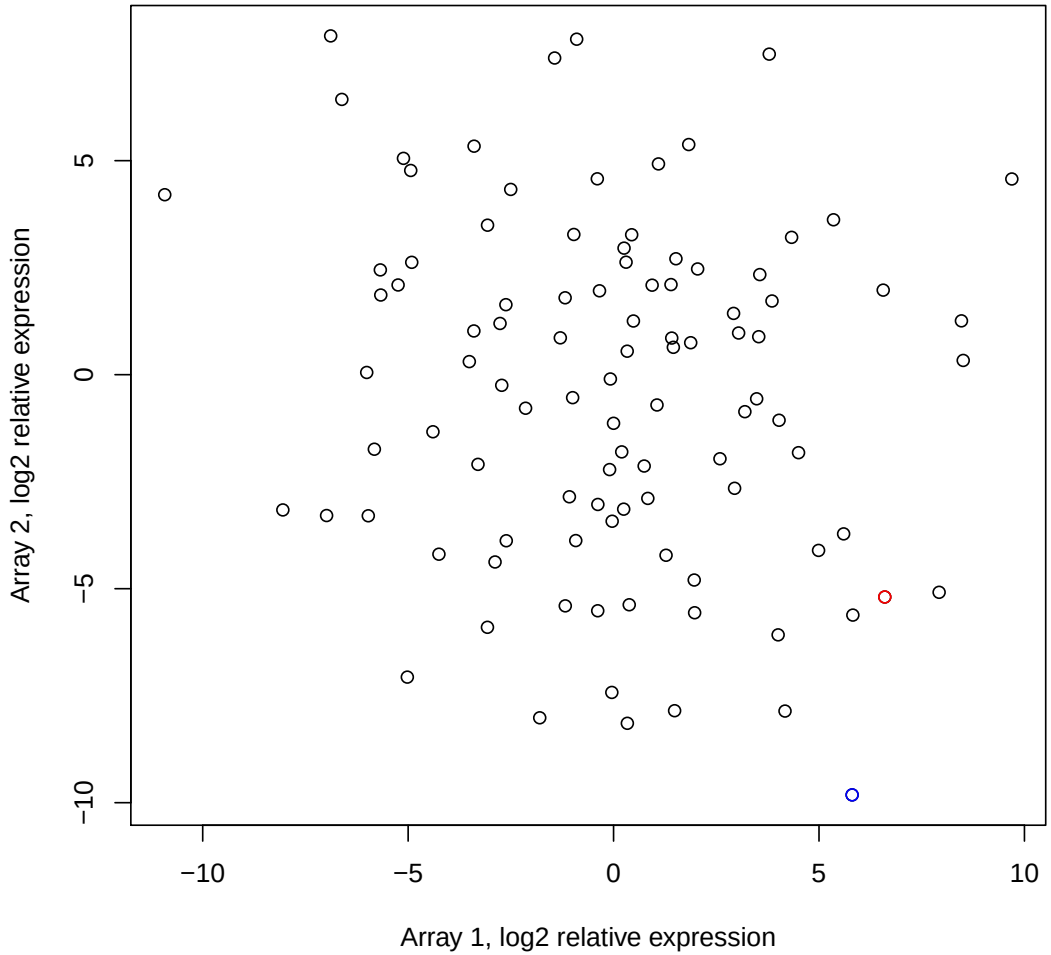
Euclidean distance

$$\frac{\sum_i^N (x_i - y_i)^2}{N} \tag{8}$$

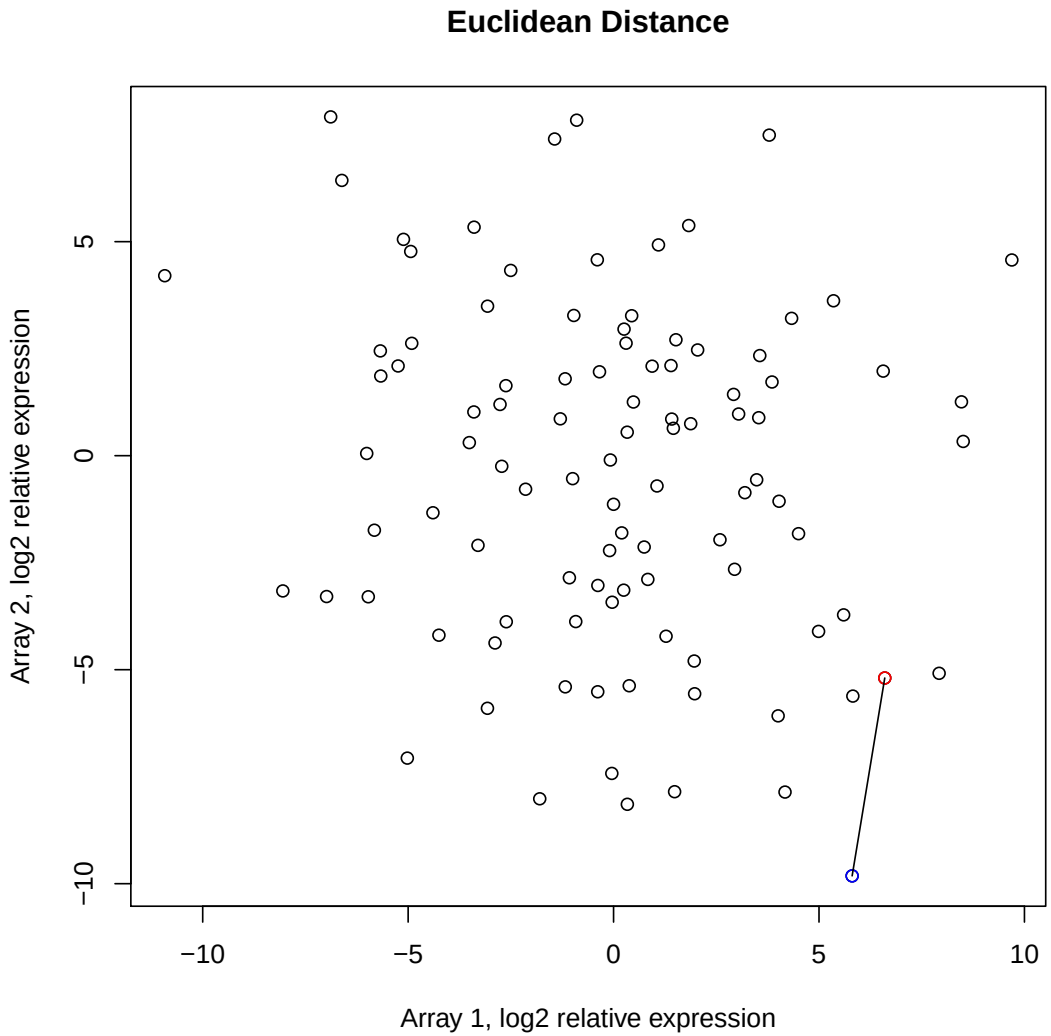
Comparing all measurements for two genes



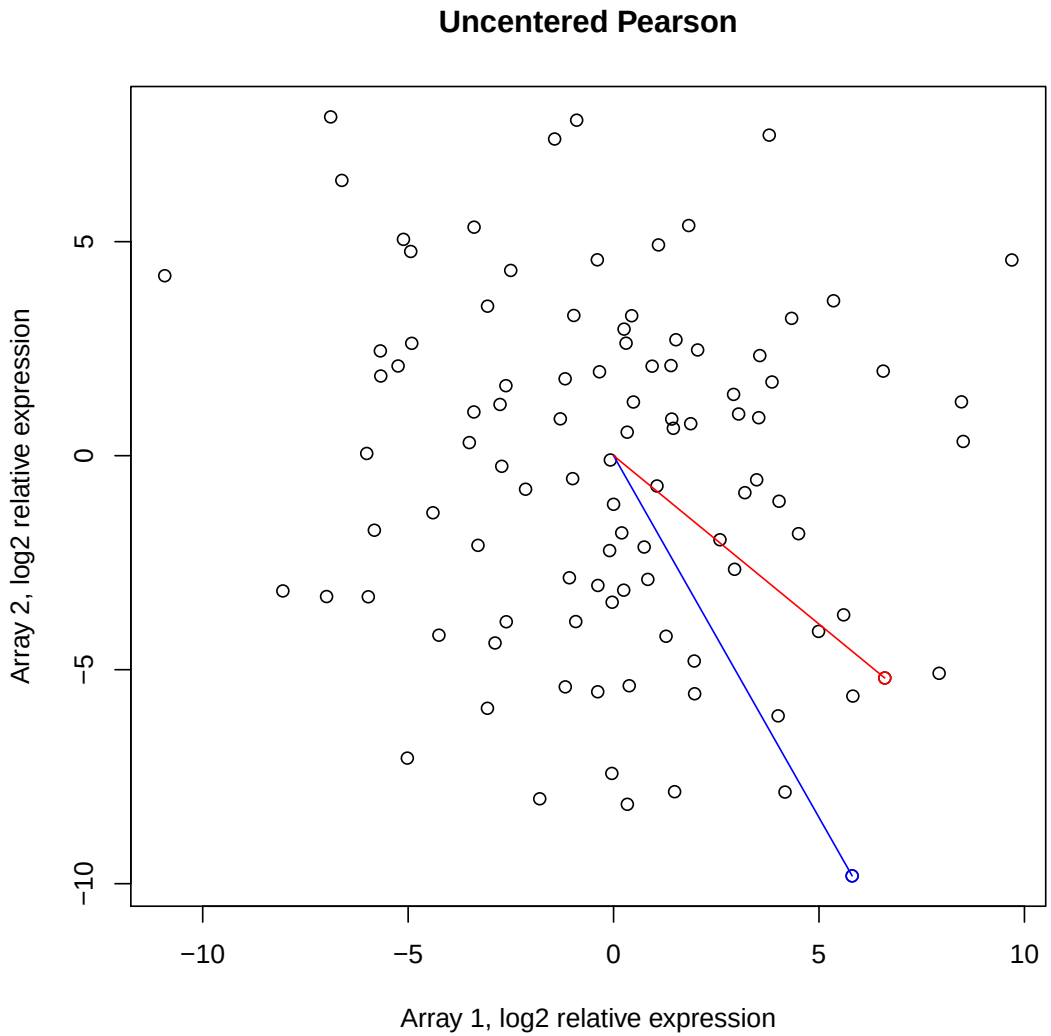
Comparing all genes for two measurements



Comparing all genes for two measurements



Comparing all genes for two measurements



List tricks

Adding data to a list:

```
mylist = []
mylist.append(3)
mylist += [4,5,6]
```

List tricks

Adding data to a list:

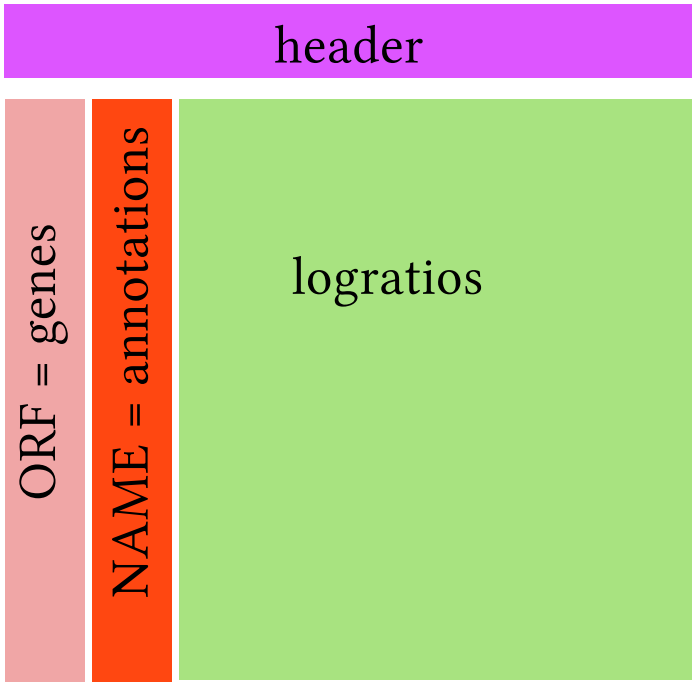
```
mylist = []
mylist.append(3)
mylist += [4,5,6]
```

Lists of lists:

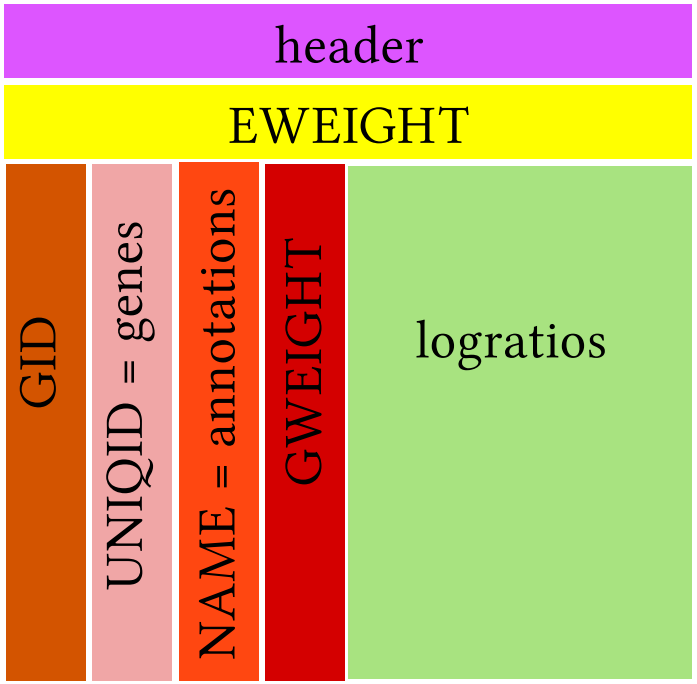
```
matrix = [[ 1, 2, 3, 4],
          [ 5, 6, 7, 8],
          [ 9,10,11,12]]
```

The CDT file format

Minimal CLUSTER input



Cluster3 CDT output



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

Distance functions

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

$$d(x,y) = 1 - \frac{\sum_i^N (x_i - x_{offset})(y_i - y_{offset})}{\sqrt{\sum_i^N (x_i - x_{offset})^2}\sqrt{\sum_i^N (y_i - y_{offset})^2}} \quad (9)$$

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.