Practical Bioinformatics

Mark Voorhies

6/25/2010

Better answers to Mike's questions

- Yes, in practice a great deal of bioinformatics programming is re-using, stitching together, and tweaking pre-existing programs. The main utility of implementing an algorithm from the ground up (as with our dynamic programming examples) is to develop your own understanding of how the algorithm works (and to confirm that a given tool works the way that you think it does).

Better answers to Mike's questions

- Yes, in practice a great deal of bioinformatics programming is re-using, stitching together, and tweaking pre-existing programs. The main utility of implementing an algorithm from the ground up (as with our dynamic programming examples) is to develop your own understanding of how the algorithm works (and to confirm that a given tool works the way that you think it does).
- We discussed that one response to finding a bug in a program is to fix it and send a patch to the original author. The other thing you should do is re-run any important calculations the were generated from the buggy code and check that your inferences are still correct. This is why it is important to keep a good record of your calculations (*e.g.*, in the form of time-stamped python scripts). A revision-control tool like git simplifies correlating buggy code versions with the calculations that they were applied to.

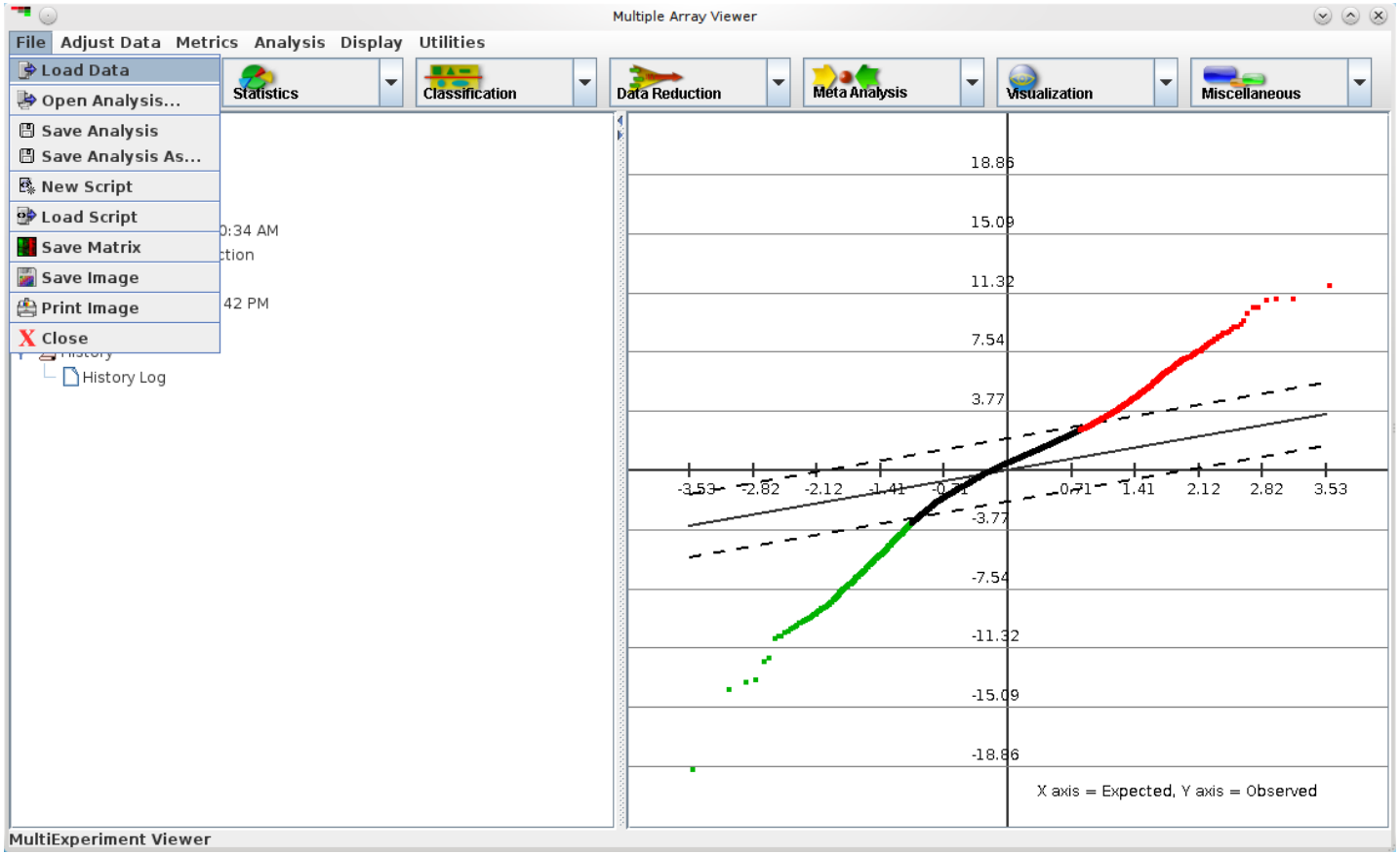
MeV SAM: Extract, normalize, and filter data

```
from ExpTransform import getExpTransform
from Database import DbLite
db = DbLite()
genome = db.GetGenomeByName("G217B")
oligos = db.Oligos()
from CdtFile import CdtFile
wgtadir = "/home/mvoorhie/papers/WGTA/output.4_1_2010/"
validCdt = CdtFile.fromCdt(open(
    wgtadir+"wgtaValid_exprValid_orthoValid_mgtaValid.cdt"))
spots = []
for i in validCdt:
    gene = genome.GetGene(i.Uniqid())
    for j in oligos.CognateSearch(gene):
        spots.append((j, gene))
def formatRatio(x):
    if(x is None):
        return ""
    else:
        assert(type(x) == float)
        return str(x)
(header, rows) = getExpTransform("Merge of Y/pool, M/pool data",
                                 [i[0] for i in spots])
out = open("valid.ymp.txt","w")
out.write("\t".join(["UID","NAME"]+list(header))+"\n")
for ((uid,gene), row) in zip(spots, rows):
    out.write("\t".join([uid.Uid(),gene.Name()]+
                        [formatRatio(i) for i in row])+"\n")
out.close()
```

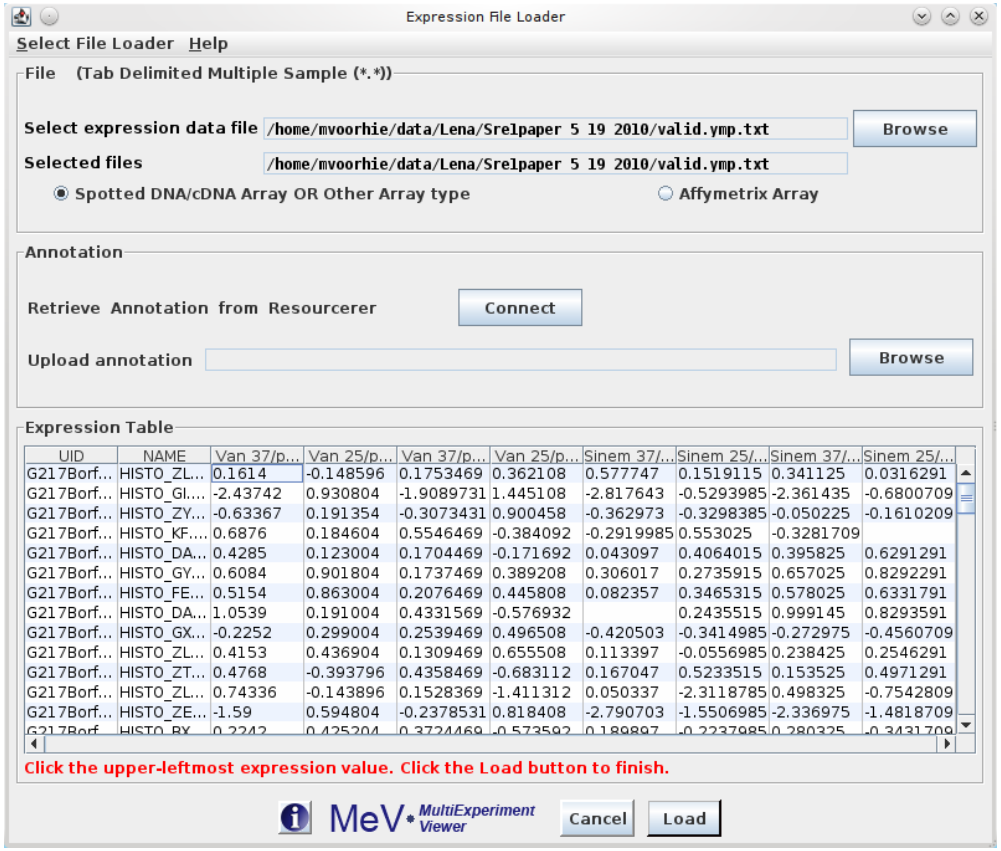
MeV SAM: Extract, normalize, and filter data

```
for (expVectorID, expVectorName) in cursor.fetchall():
    names.append(expVectorName)
    for (oligo, retval) in zip(oligos, logratios):
        if(cursor.execute("""
        SELECT weight, R, G, log2norm, flag
        FROM ExpVectorTerm, ExpSpot, ExpArray, MetaSpot
        WHERE ExpVectorID = %s
        AND ExpArray.sid = ExpVectorTerm.sid
        AND ExpArray.sid = ExpSpot.sid
        AND ExpArray.aid = MetaSpot.aid
        AND ExpSpot.mid = MetaSpot.mid
        AND MetaSpot.uid = %s
        """, (expVectorID, oligo)) > 0):
            logratio = 0.0
            for (weight, R, G, log2norm, flag) in cursor.fetchall():
                if((flag < 0) or (None in (R, G))):
                    logratio = None
                    break
                try:
                    logratio += weight*(R-G+log2norm)
                except:
                    logratio = None
                    break
                if(2**R + 2**G < threshold):
                    logratio = None
                    break
            retval.append(logratio)
```

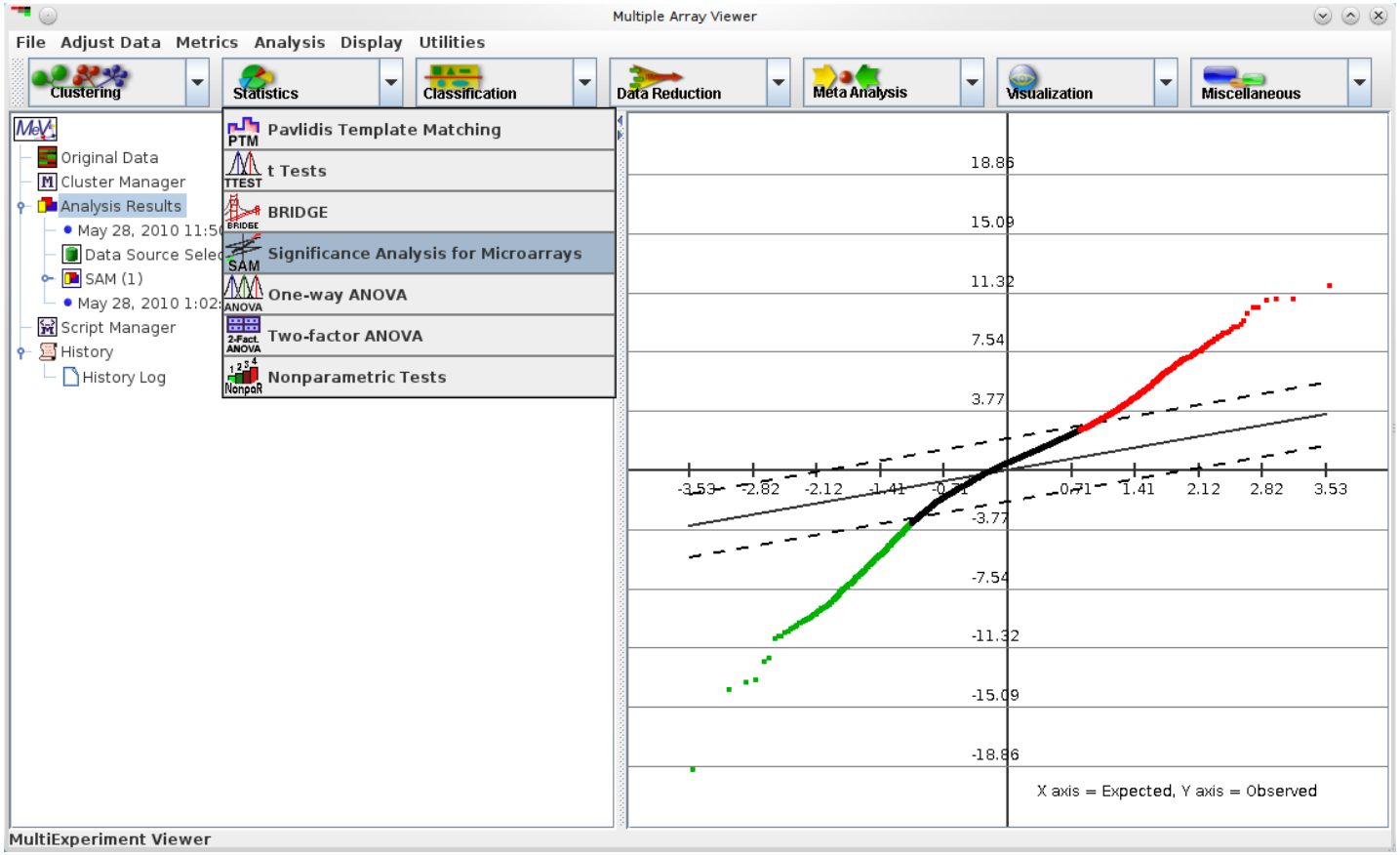
MeV SAM: Load data



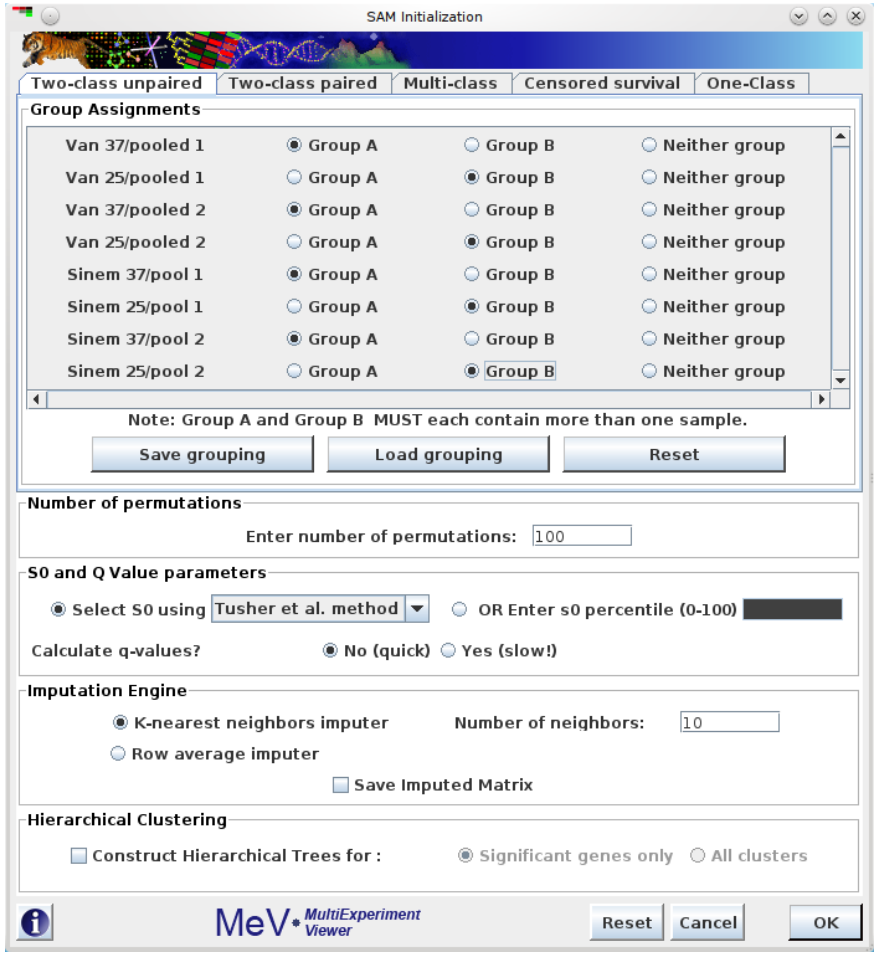
MeV SAM: normalized, log transformed, TDT data



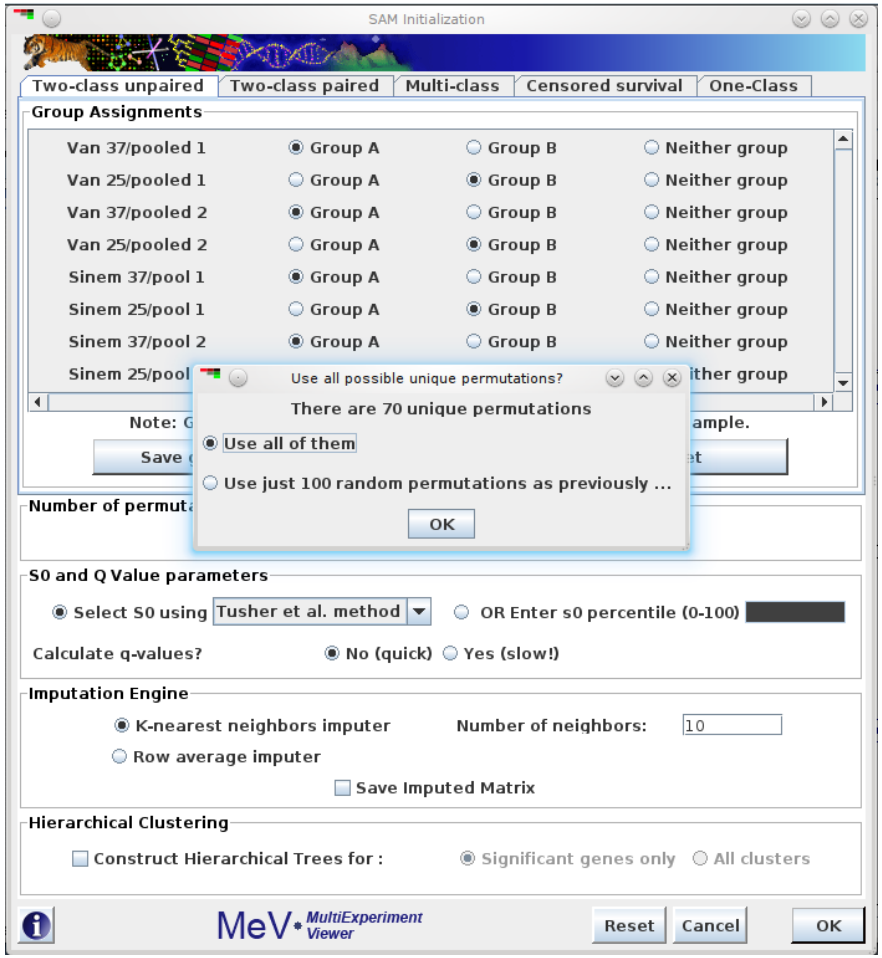
MeV SAM: Choose SAM



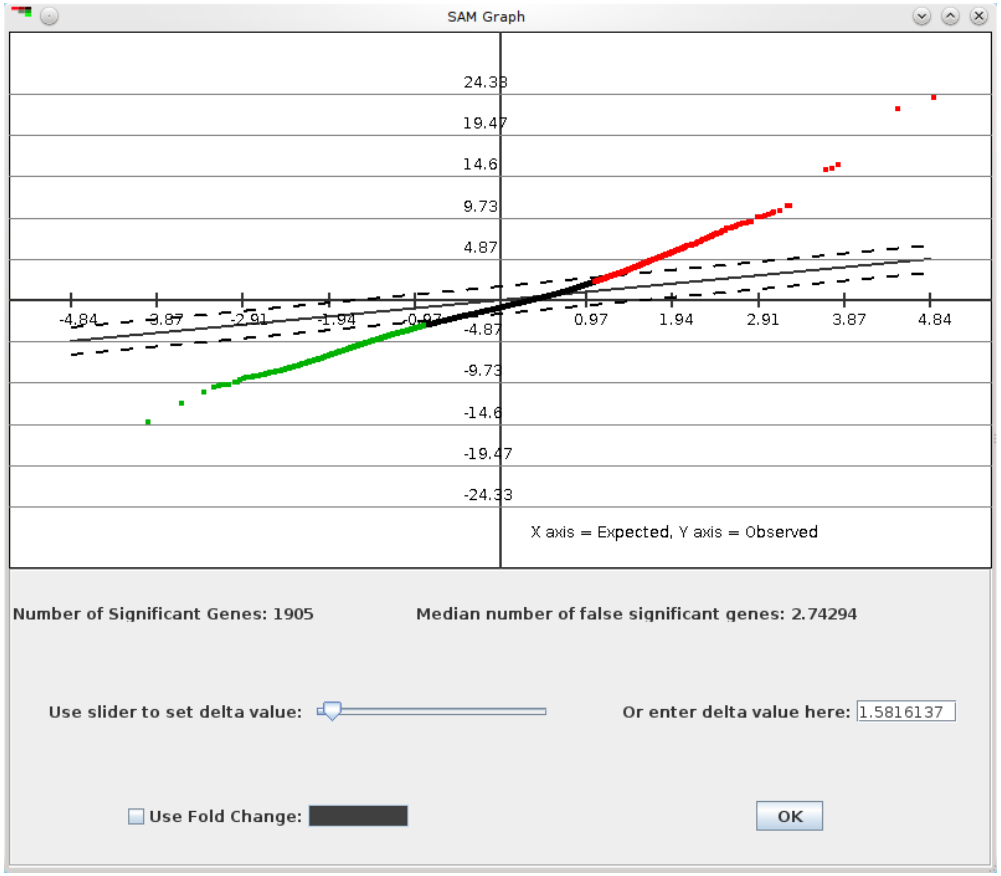
MeV SAM: Describe experiment, choose parameters



MeV SAM: Choose permutations for FDR



MeV SAM: Choose delta



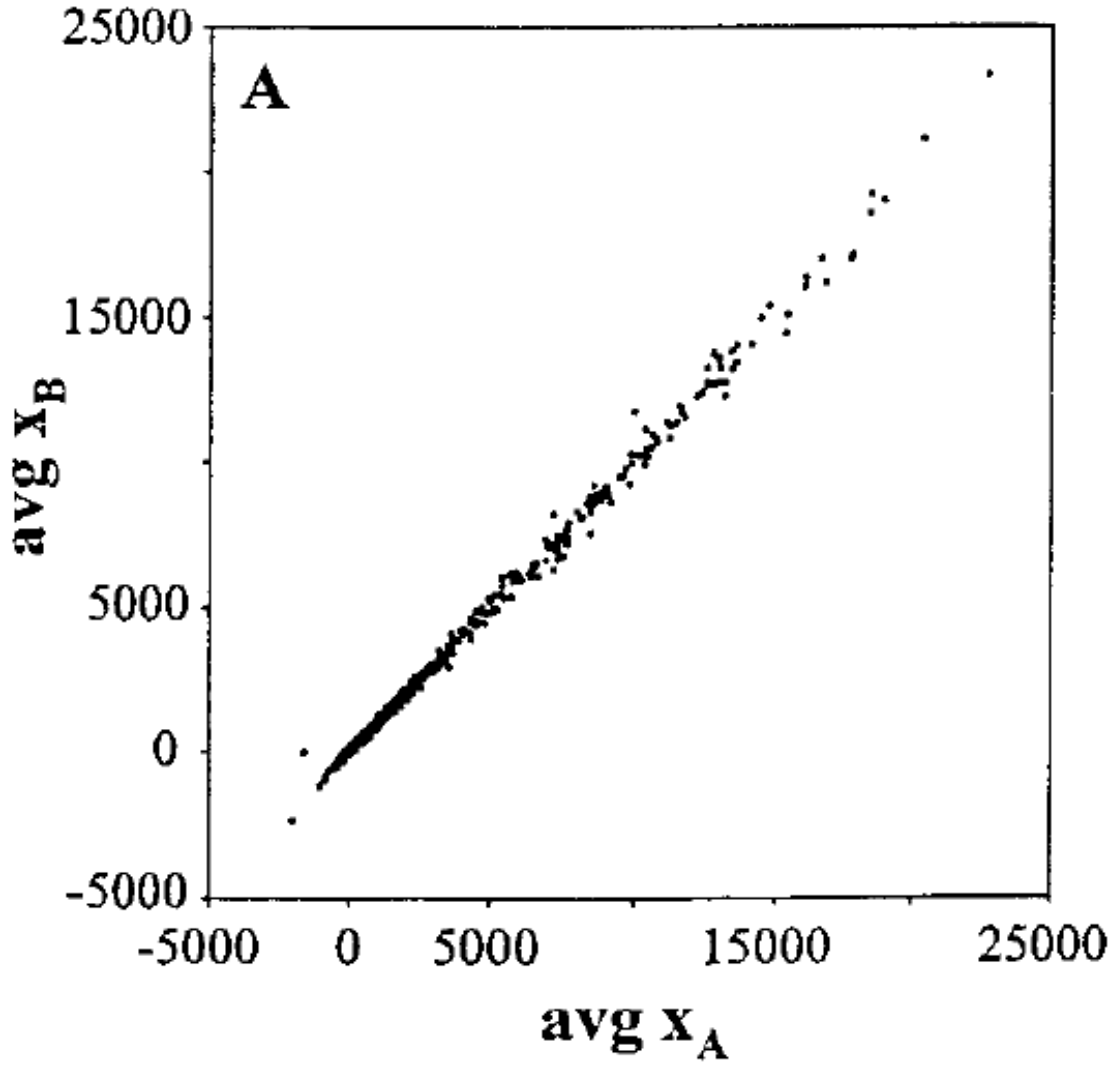
Significant analysis of microarrays applied to the ionizing radiation response

Virginia Goss Tusher, Robert Tibshirani, and Gilbert Chu

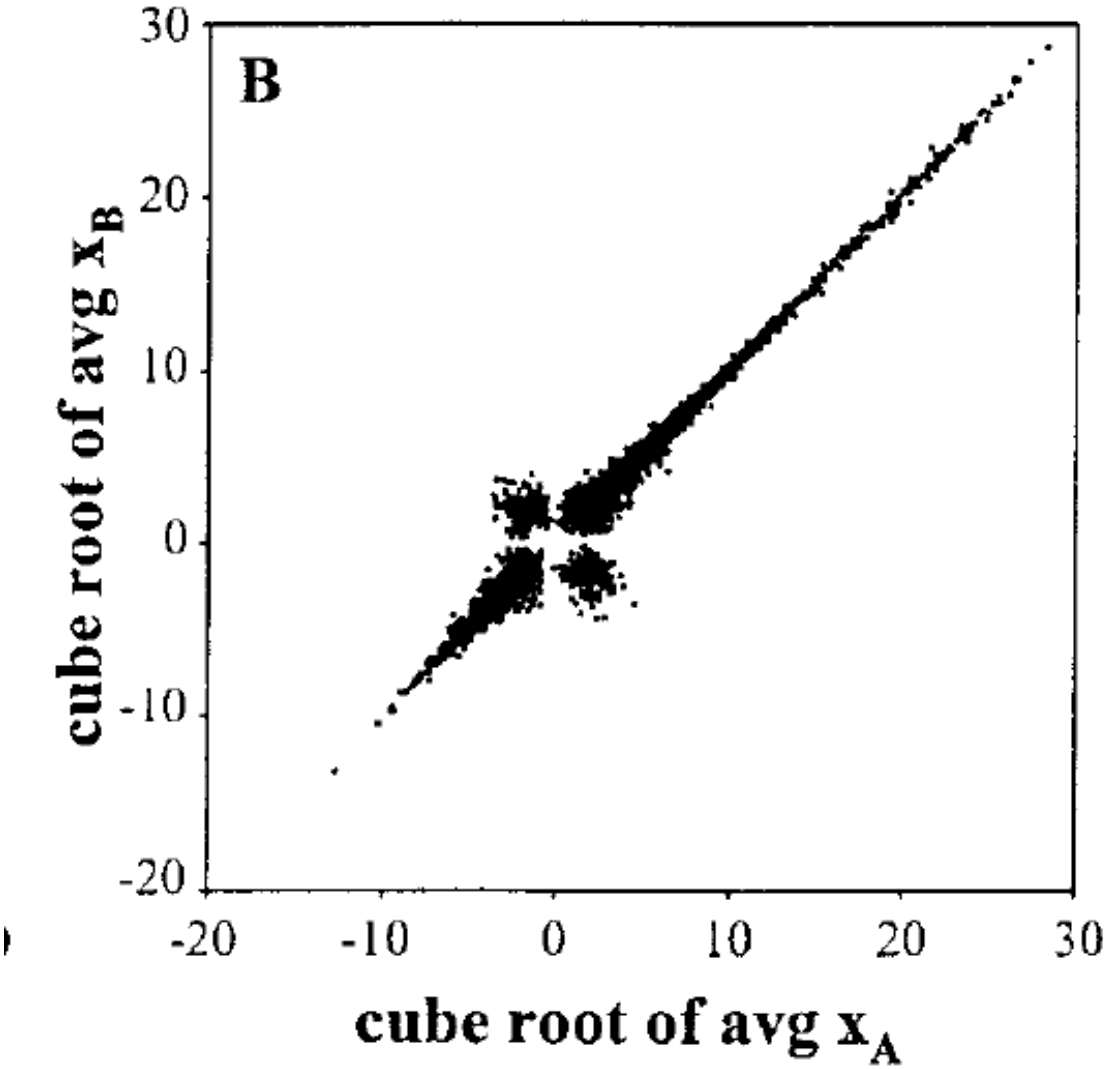
Normalizing Affy Arrays with Technical Replicates

This is very similar to mean normalization for two color arrays.

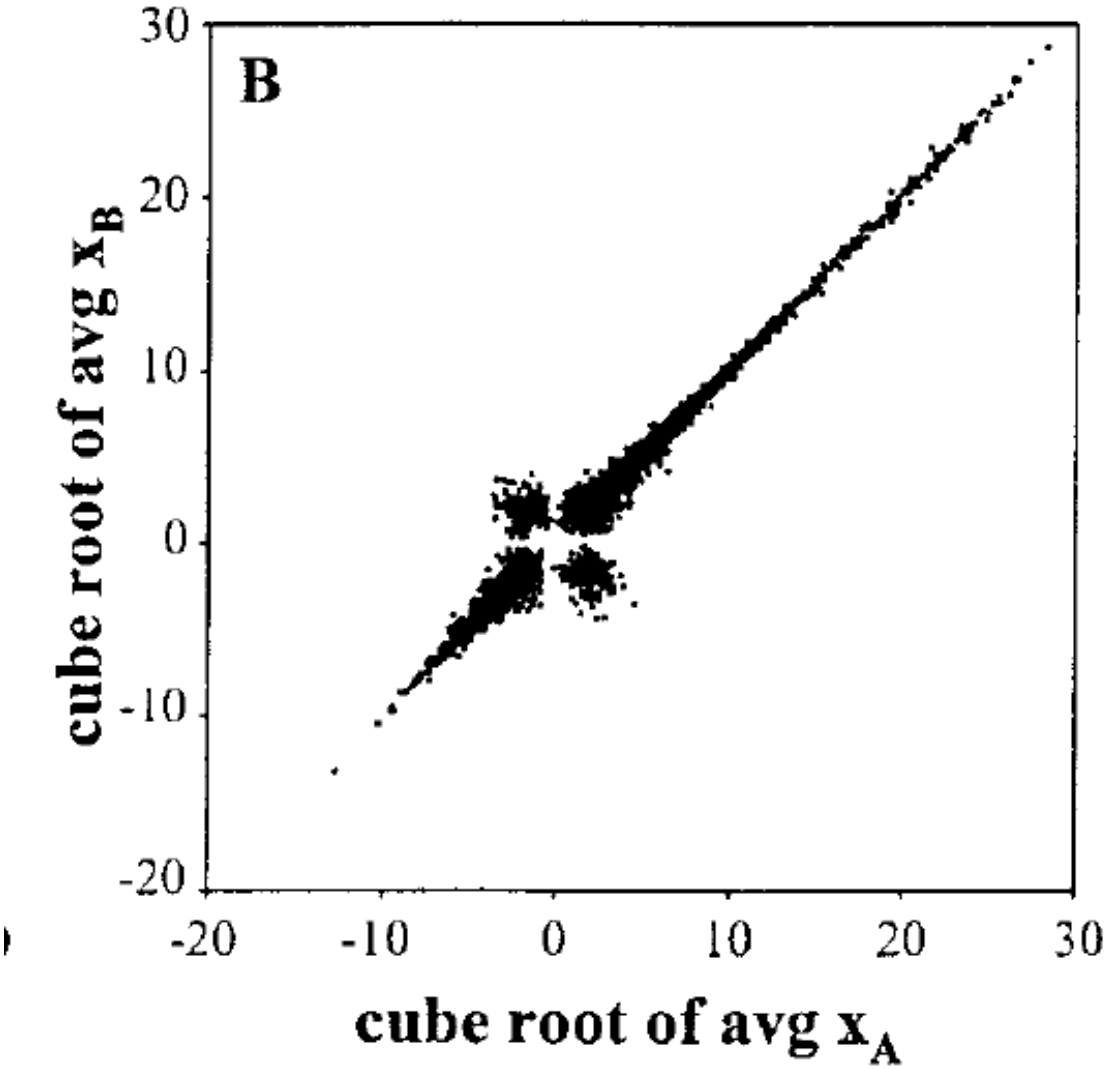
1a: Comparing normalized data



1b: Cube Root Transform

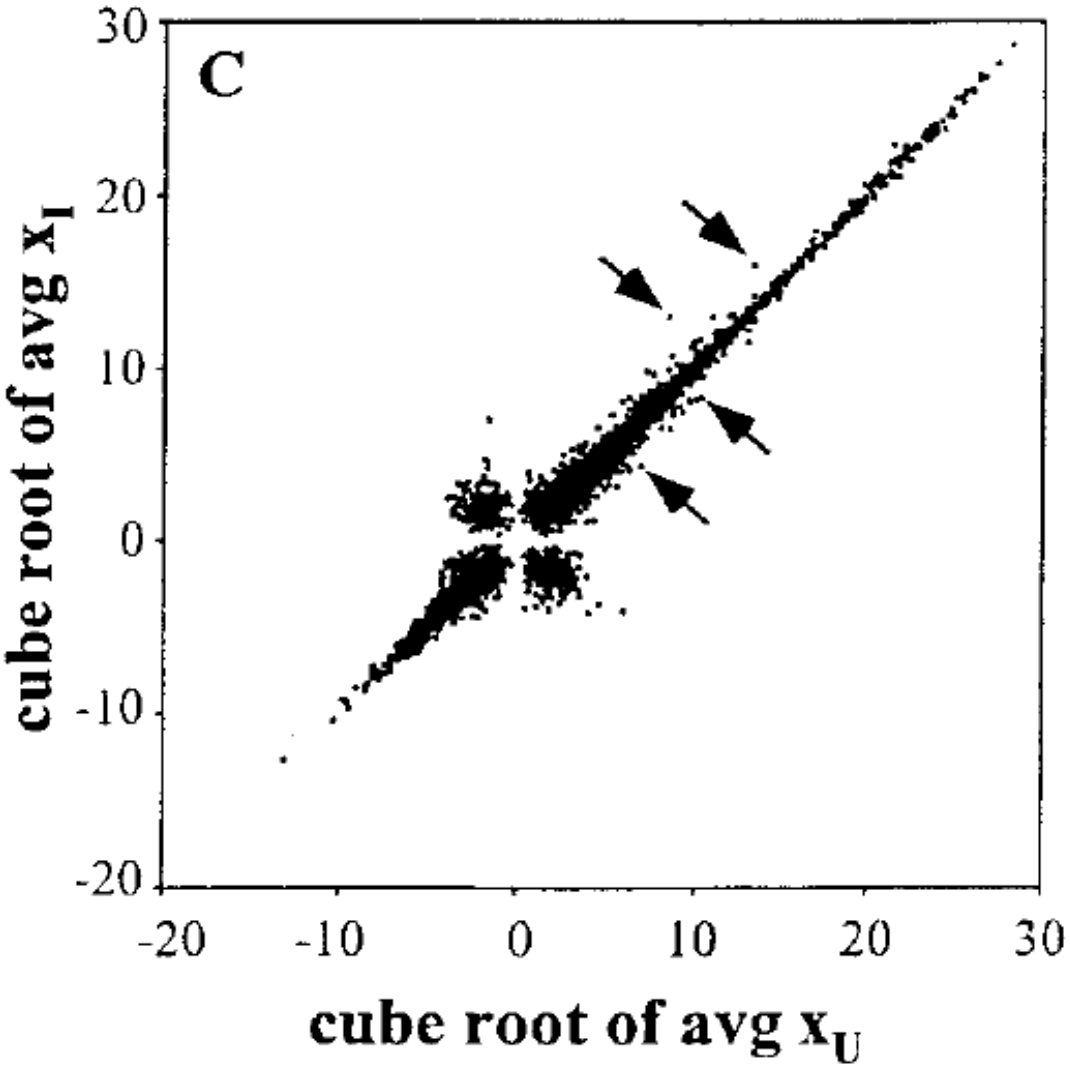


1b: Cube Root Transform

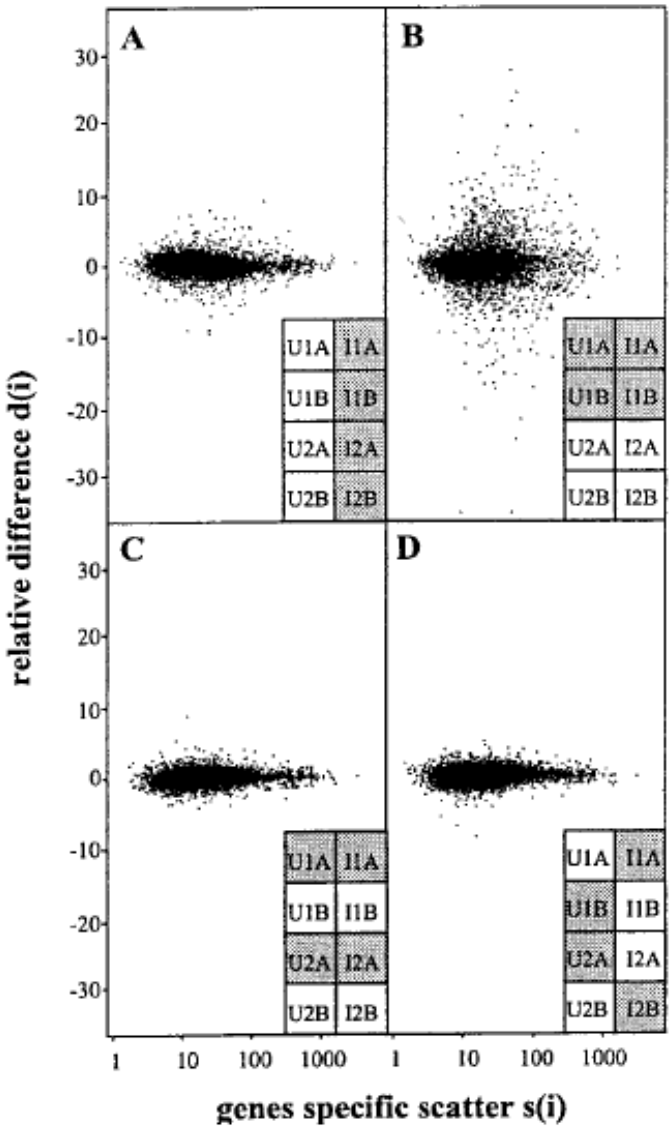


- Motivation is to get good resolution of all of the data
- Problems: weird behavior near zero, compression of error for negative values, not biologically motivated
- Better: add constant offset (or filter low intensity data) and log transform

1c: Outliers in treatment comparison

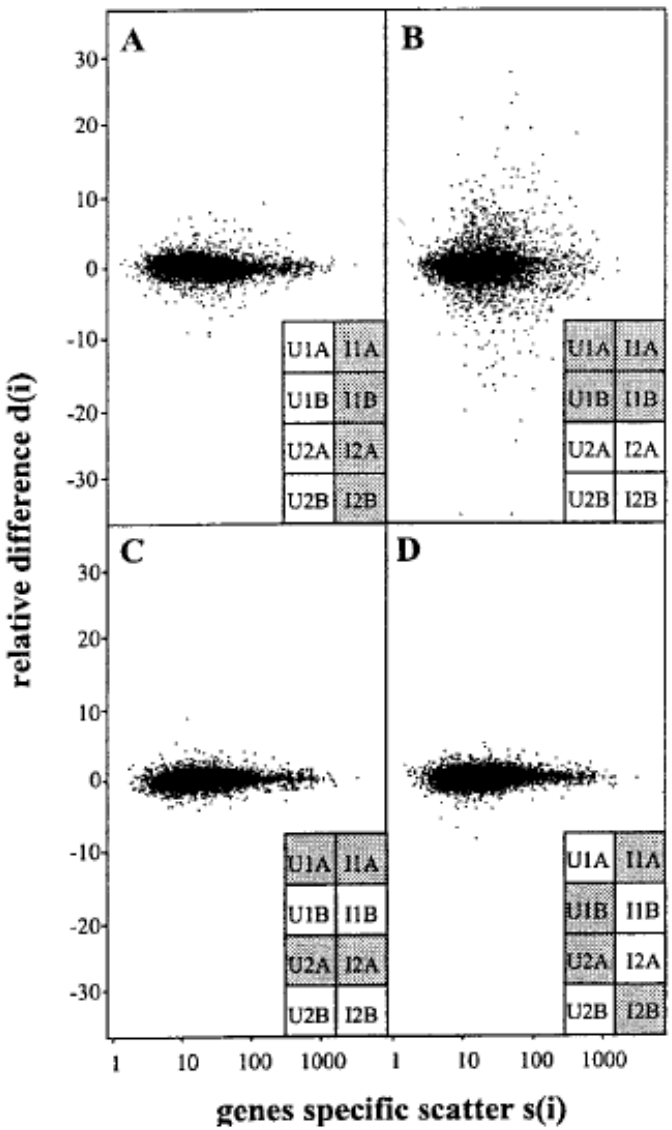


2: d(i) statistic



$$d(i) = \frac{\bar{x_I}(i) - \bar{x_U}(i)}{s(i) + s_0}$$

2: d(i) statistic

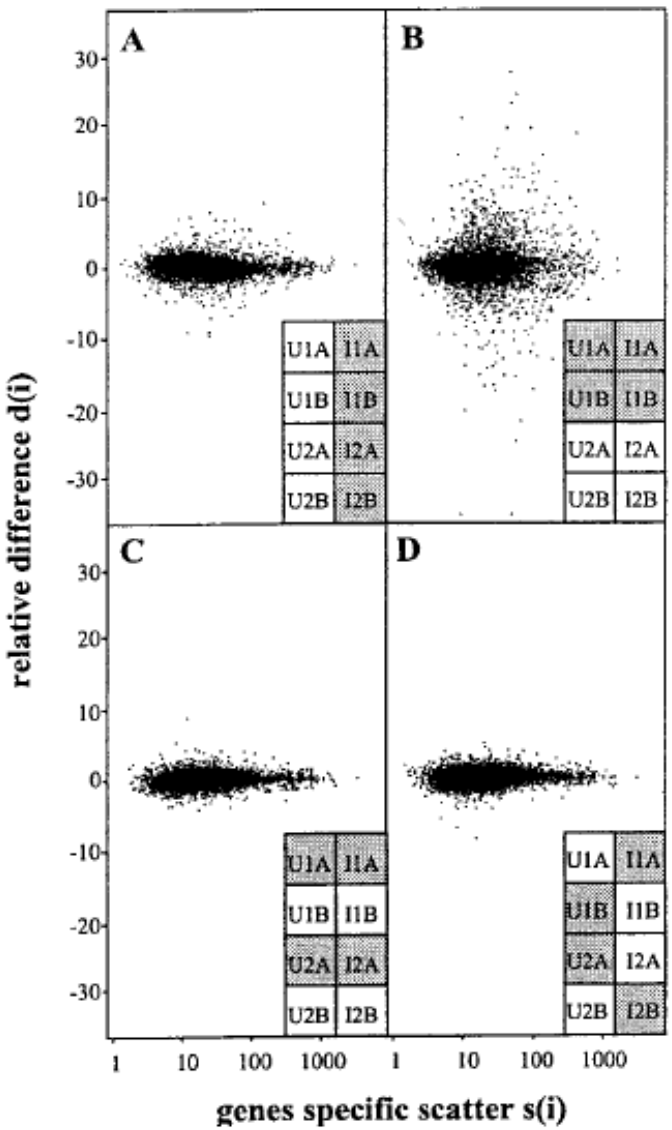


$$d(i) = \frac{\bar{x_I}(i) - \bar{x_U}(i)}{s(i) + s_0}$$

$$s(i) = \sqrt{a \sum_m [x_m(i) - \bar{x_I}(i)]^2 + \sum_n [x_n(i) - \bar{x_U}(i)]^2}$$

$$a = (1/n_I + 1/n_U)/(n_I + n_U - 2)$$

2: d(i) statistic



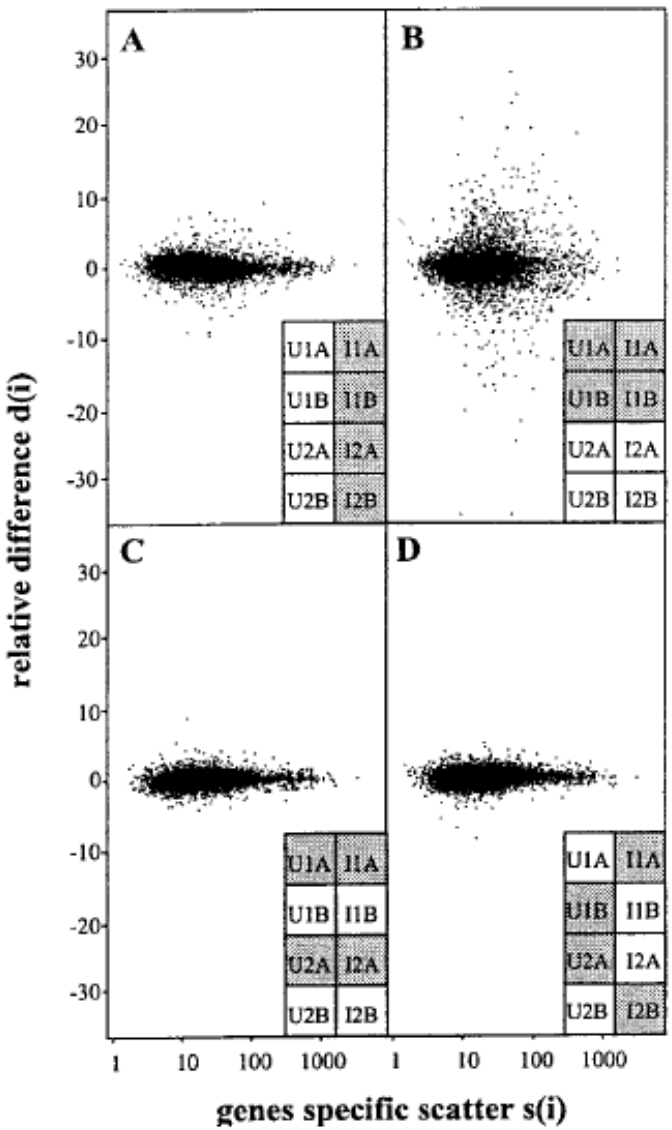
$$d(i) = \frac{\bar{x_I}(i) - \bar{x_U}(i)}{s(i) + s_0}$$

$$s(i) = \sqrt{a \sum_m [x_m(i) - \bar{x_I}(i)]^2 + \sum_n [x_n(i) - \bar{x_U}(i)]^2}$$

$$a = (1/n_I + 1/n_U)/(n_I + n_U - 2)$$

- 1 vs. 2 = biological replicate
- A vs. B = technical replicate
- I vs. U = treatment

2: d(i) statistic



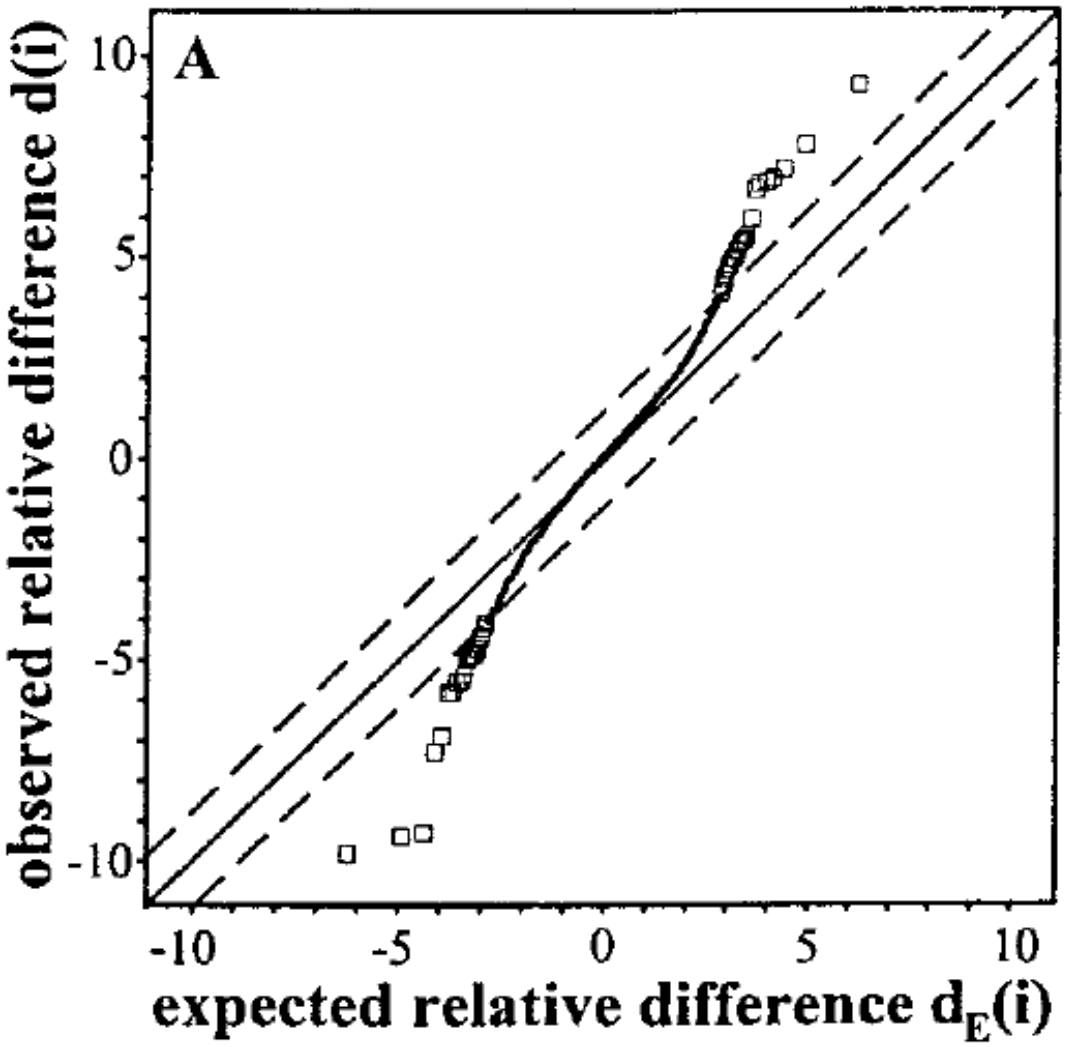
$$d(i) = \frac{\bar{x_I}(i) - \bar{x_U}(i)}{s(i) + s_0}$$

$$s(i) = \sqrt{a \sum_m [x_m(i) - \bar{x_I}(i)]^2 + \sum_n [x_n(i) - \bar{x_U}(i)]^2}$$

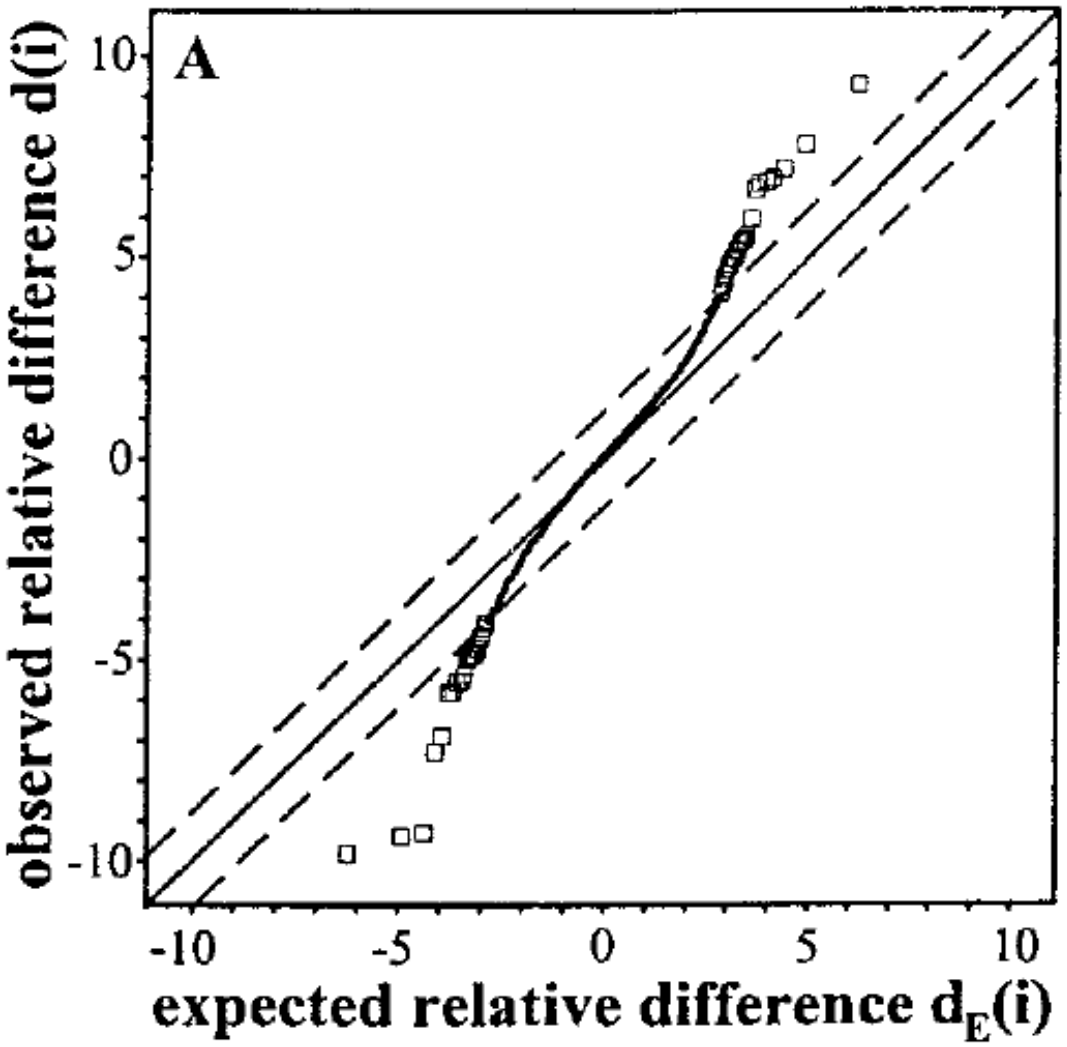
$$a = (1/n_I + 1/n_U)/(n_I + n_U - 2)$$

- 1 vs. 2 = biological replicate
- A vs. B = technical replicate
- I vs. U = treatment
- s_0 forces a minimum variance for the low intensity data

3a: The SAM plot

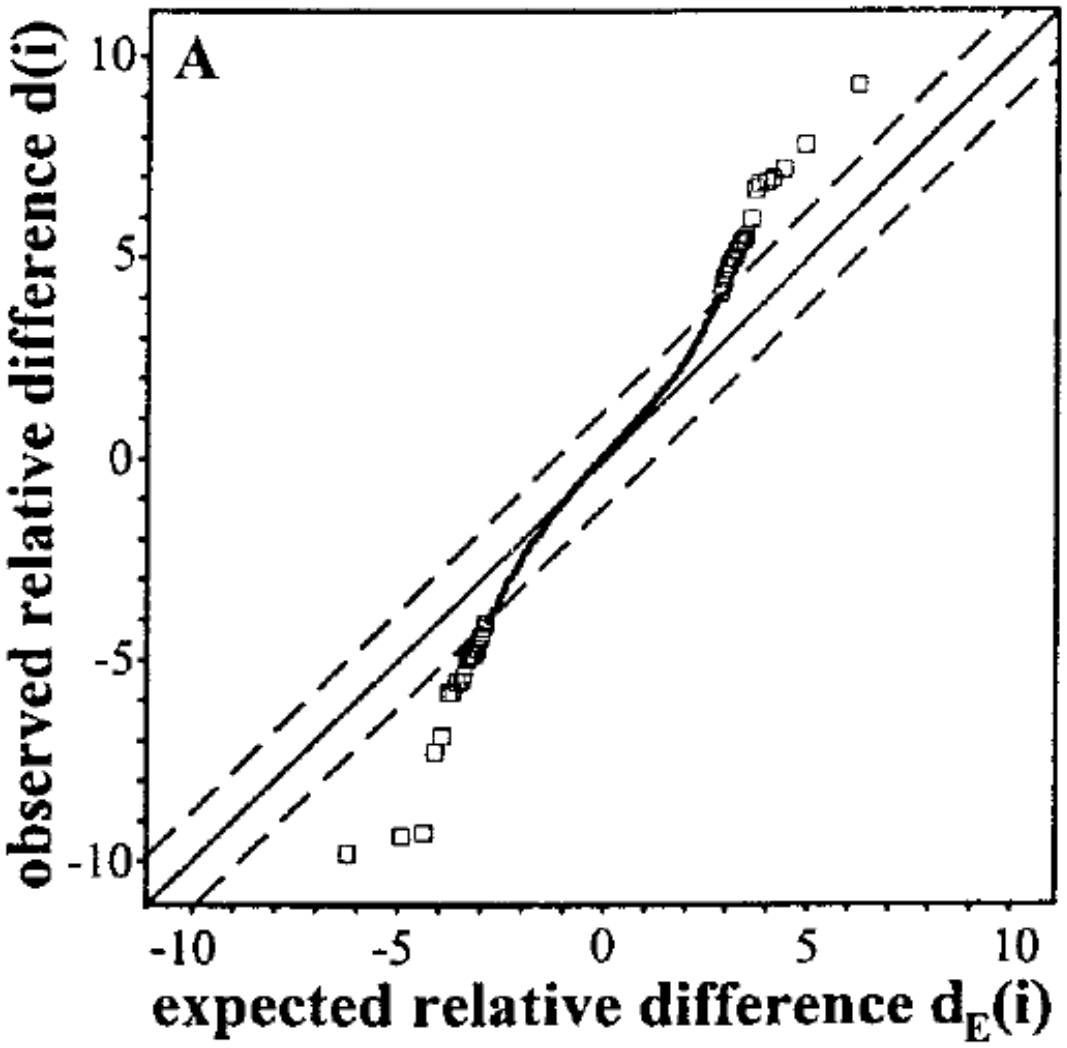


3a: The SAM plot



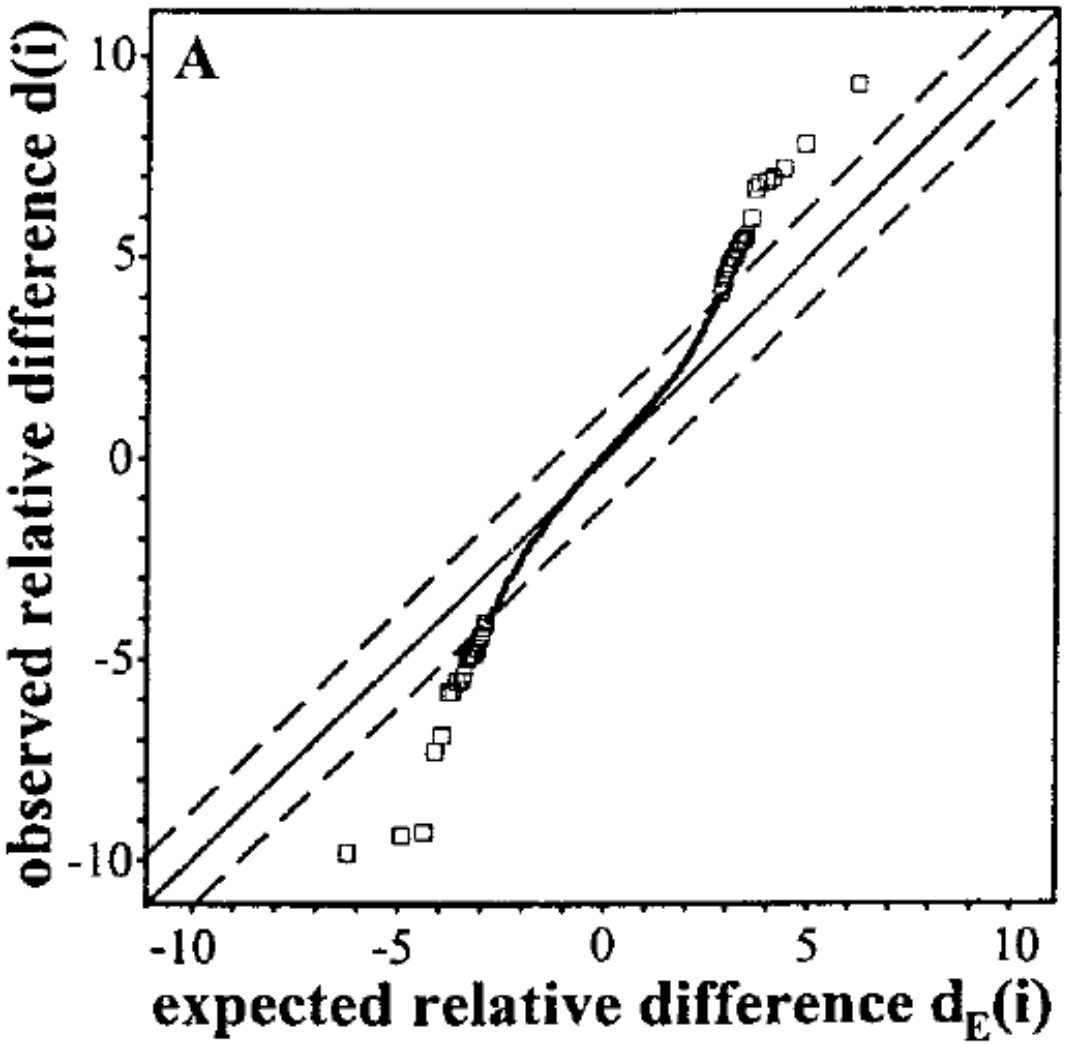
- "Expected" is the average d(i) for all "balanced" permutations of the data.

3a: The SAM plot



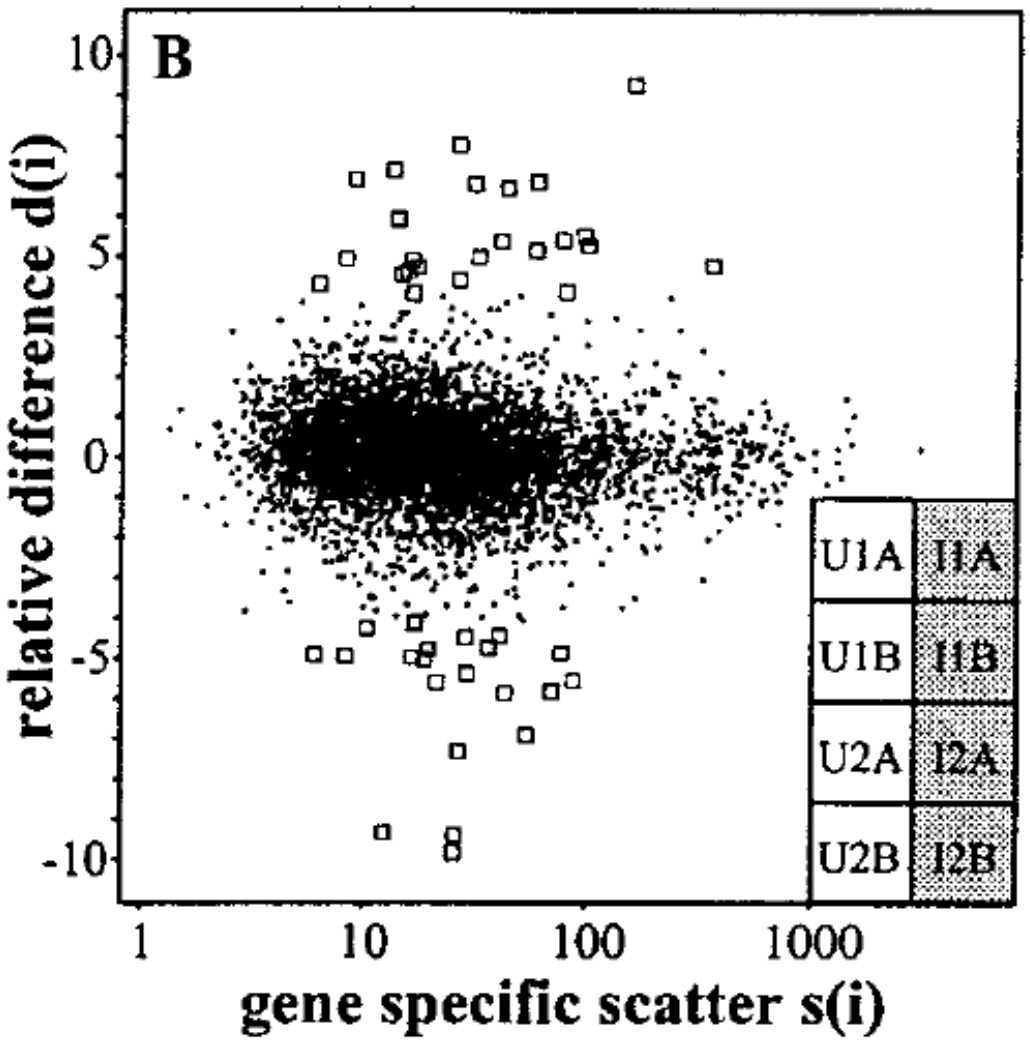
- "Expected" is the average d(i) for all "balanced" permutations of the data.
- "delta" is an offset from the line of best fit, giving two diagonal thresholds.
- The plot is monotonic, so diagonal thresholds are also horizontal and vertical thresholds.

3a: The SAM plot

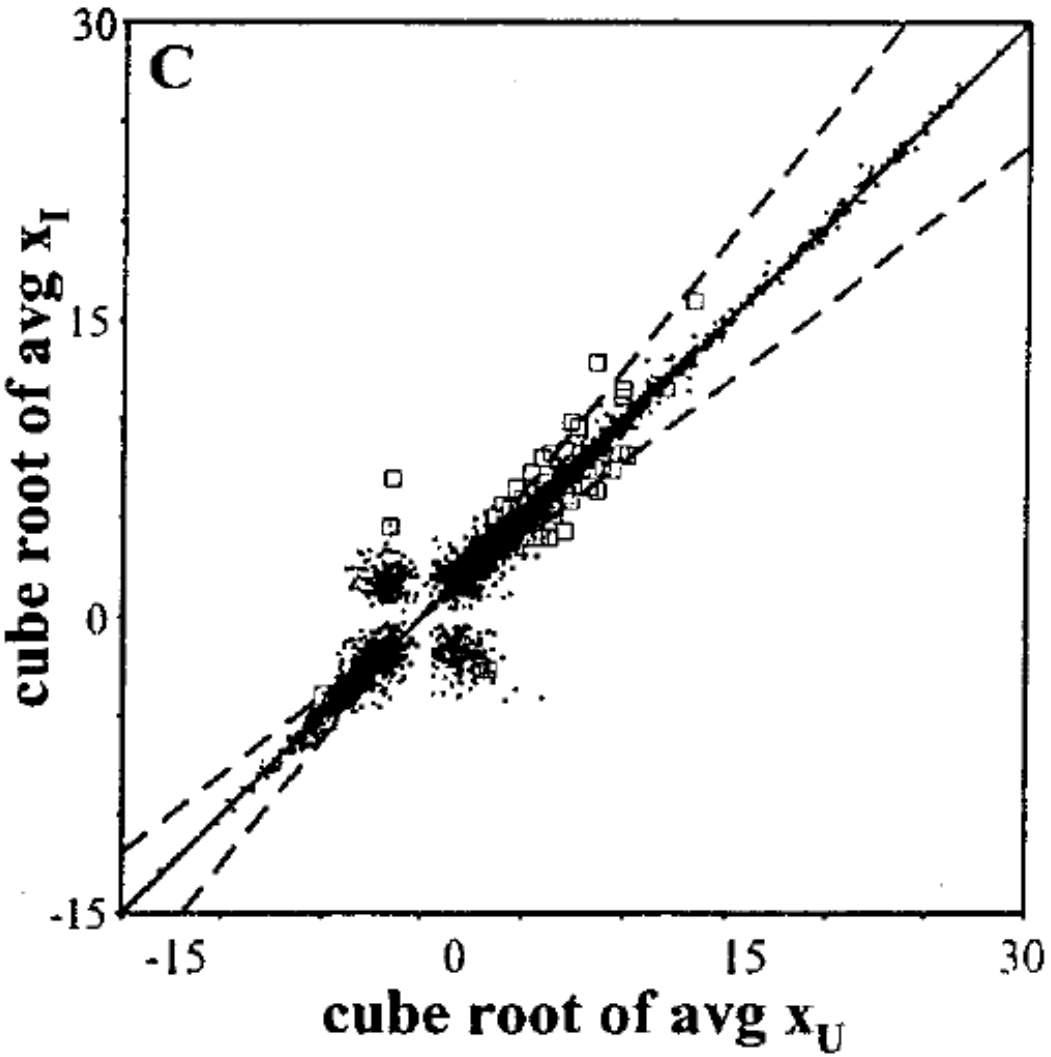


- FDR is calculated by replacing "observed" with each of the balanced permutations in turn (or a random sample for large data sets).

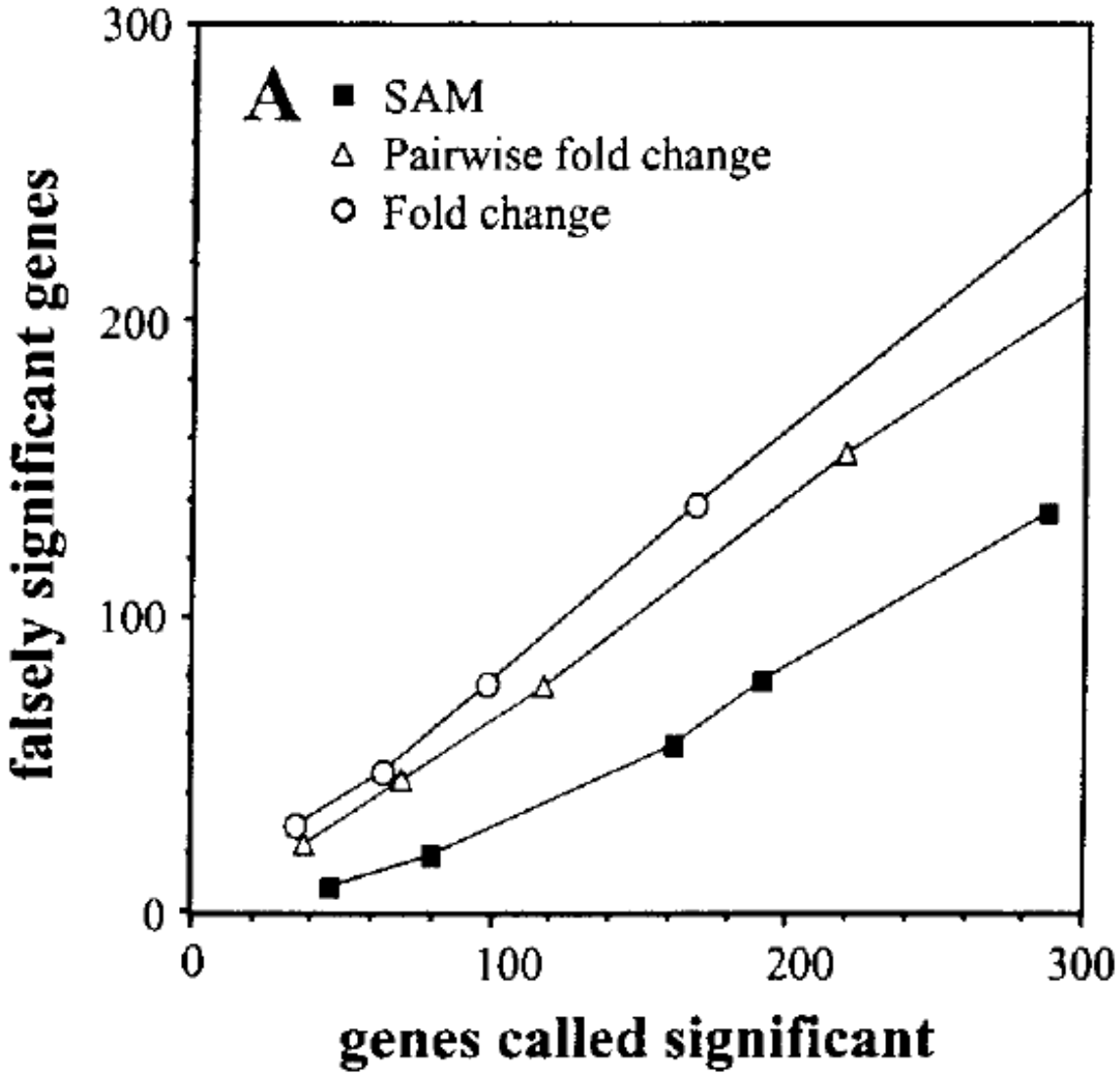
3b: Variance of significant genes



3c: Expression levels of significant genes

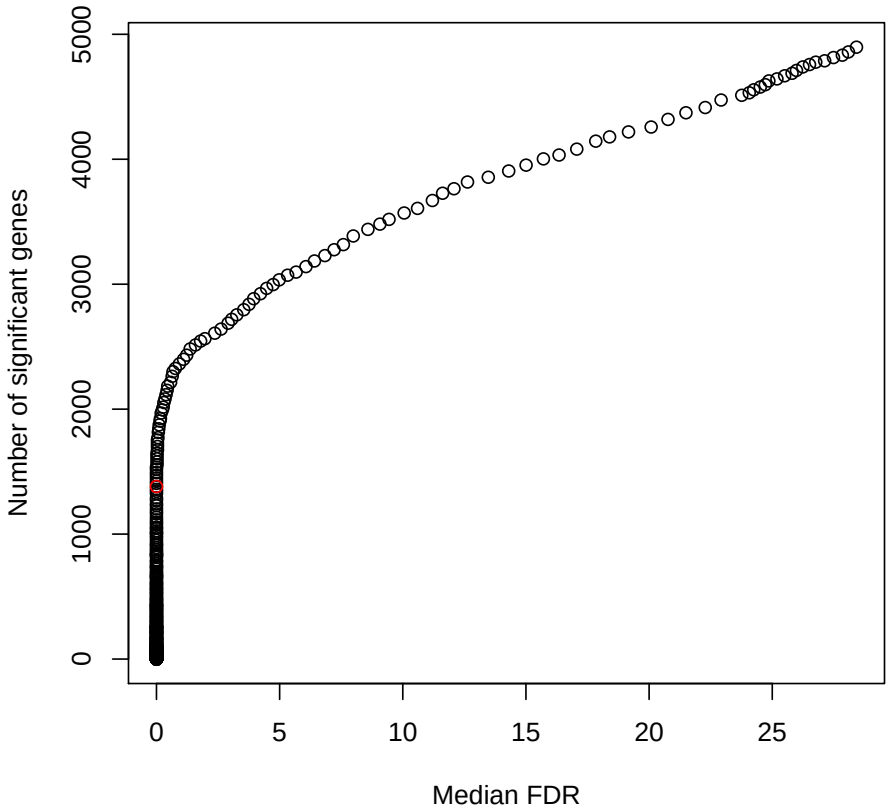


4a: Sensitivity vs. Specificity

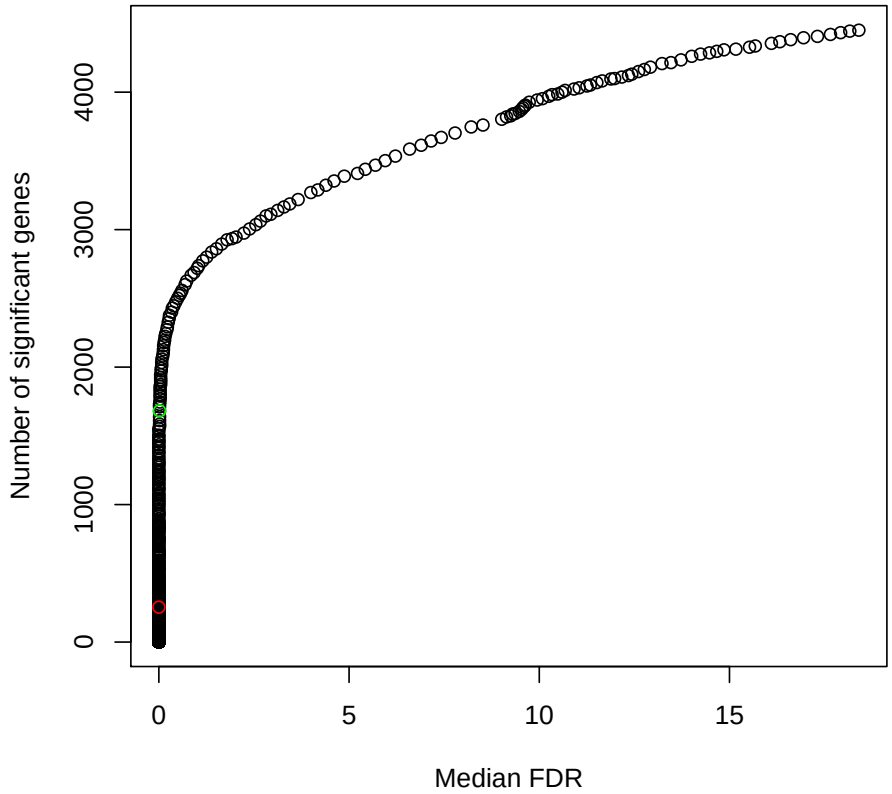


Sensitivity vs. Specificity: Pseudo-ROC plots

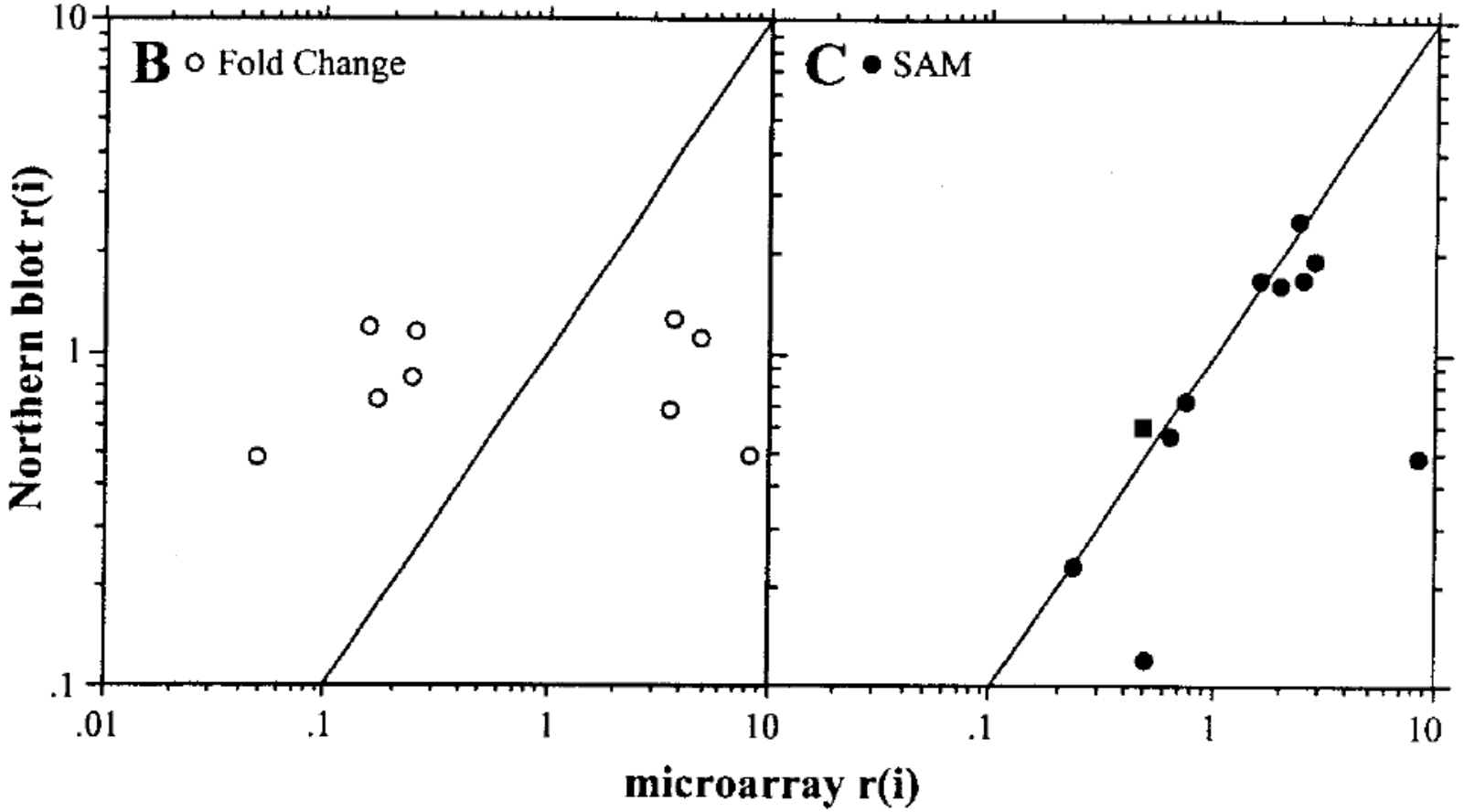
Two-class SAM



One-class SAM



4b and c: Experimental Validation



Groovy Packages

- Python modules
  - numpy (numeric)
  - matplotlib
  - scipy
  - rpy
  - Pycluster (Biopython)
  - MySQLdb
- Distributions
  - Enthought Python Distribution
  - Ubuntu Linux