Pointillist Tutorial and Users' Guide

November 20, 2005

Daehee Hwang, Ph.D.

Bolouri and Hood labs
Institute for Systems Biology

Table of Contents

Introduction

This document is available at http://labs.systemsbiology.net/bolouri/software/Pointillist as a Tutorial and Users' Guide for the integration methods, named Pointillist, which are introduced in:

- Hwang D. et al. (2005) "A data integration methodology for systems biology." *PNAS* **102** (48), 17296-17301.
- Hwang D. et al. (2005) "A data integration methodology for systems biology: Experimental verification." *PNAS* **102** (48), 17302-17307.

Pointillist is an open source software package that consists of the MATLAB codes (implementing all the integration methods described in the above papers) and a Java code (implementing the non-parametric integration method using Fisher's method). The integration methods in Pointillist are general-purpose and may be applied to integrate data from any existing and future technologies without requiring training datasets. In this introductory Tutorial and Users' Guide, you will see in detail how to perform your own data integration using the MATLAB codes and several practical issues that the papers describe only briefly. The following paragraphs provide the background of the data integration in systems biology as described in the papers.

Systems biology aims to describe cellular events related to a particular phenotype of interest (e.g., a diseased condition) in the form of a biological network by analyzing global datasets from various high-throughput technologies: 1) genomic (microarray, Massive Parallel Signature Sequencing, Serial Analysis of Gene Expression) and 2) proteomic (protein arrays, mass spectrometry) technologies, and 3) global assays probing protein-DNA (chromatin immunoprecipitation on chip) and protein-protein interactions (yeast two hybrids and Tandem Affinity Purification-tag).

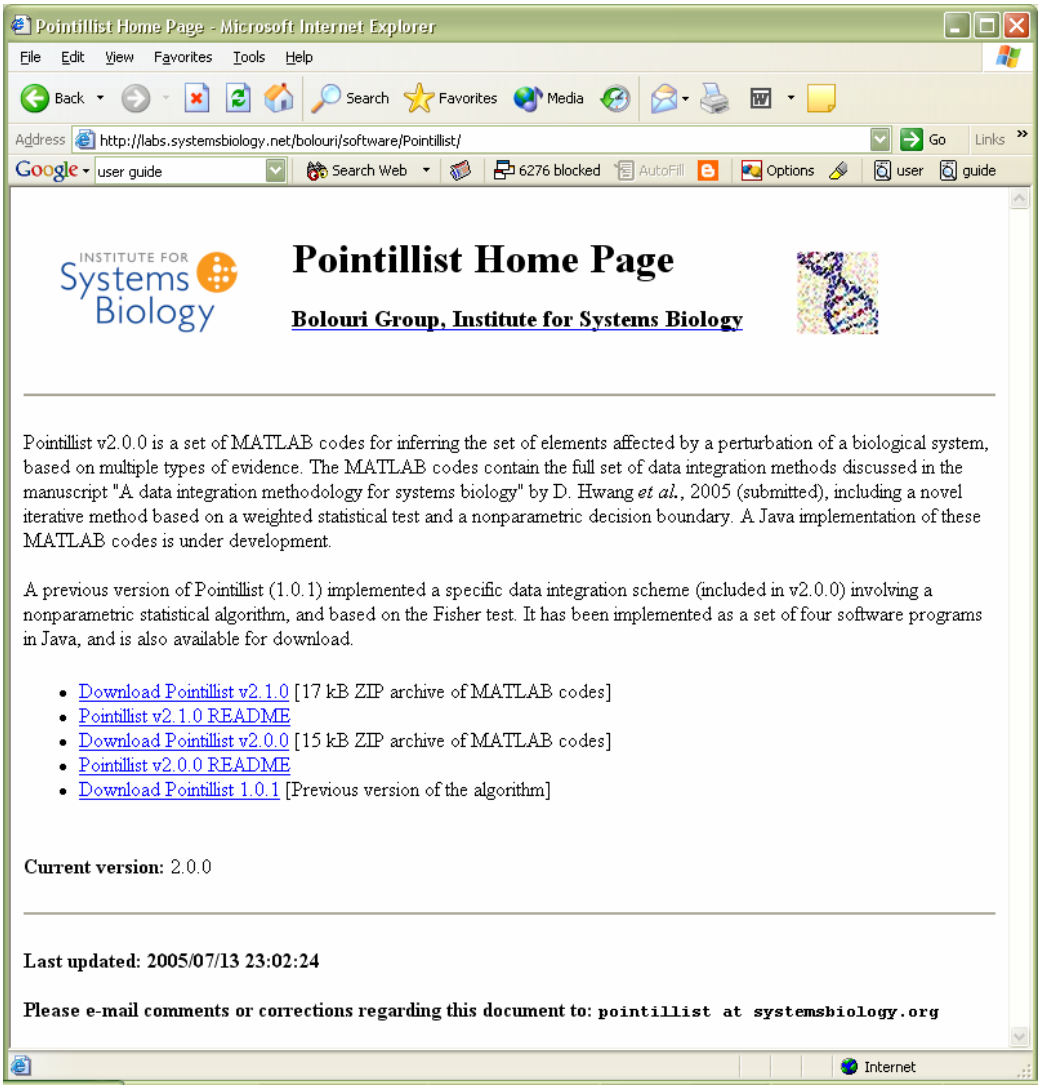
However, each of these technologies has different levels of accuracy and system coverage (statistical power), resulting in high false positive and false negative rates when each technology alone is used. These error rates can be reduced by integrating multiple datasets from different high throughput technologies by reinforcing bona fide observations and also capturing complementary information. Additionally, because of the rapid rate of progress in biotechnology, there is usually no curated exemplar (or training) dataset from which one might estimate the parameters of the integration methods requiring such training datasets.

Therefore, the effective data integration requiring no training datasets is needed to deal effectively with the difference in statistical power, thus reducing the error rates. To address these concerns, we have developed data integration methods that can handle multiple datasets differing in statistical power, type, size, and network coverage without requiring a curated training dataset. We outlined our methods and then demonstrated their performance by applying them to simulated and real datasets in the two papers mentioned above.

Getting Started

Installation and System requirements

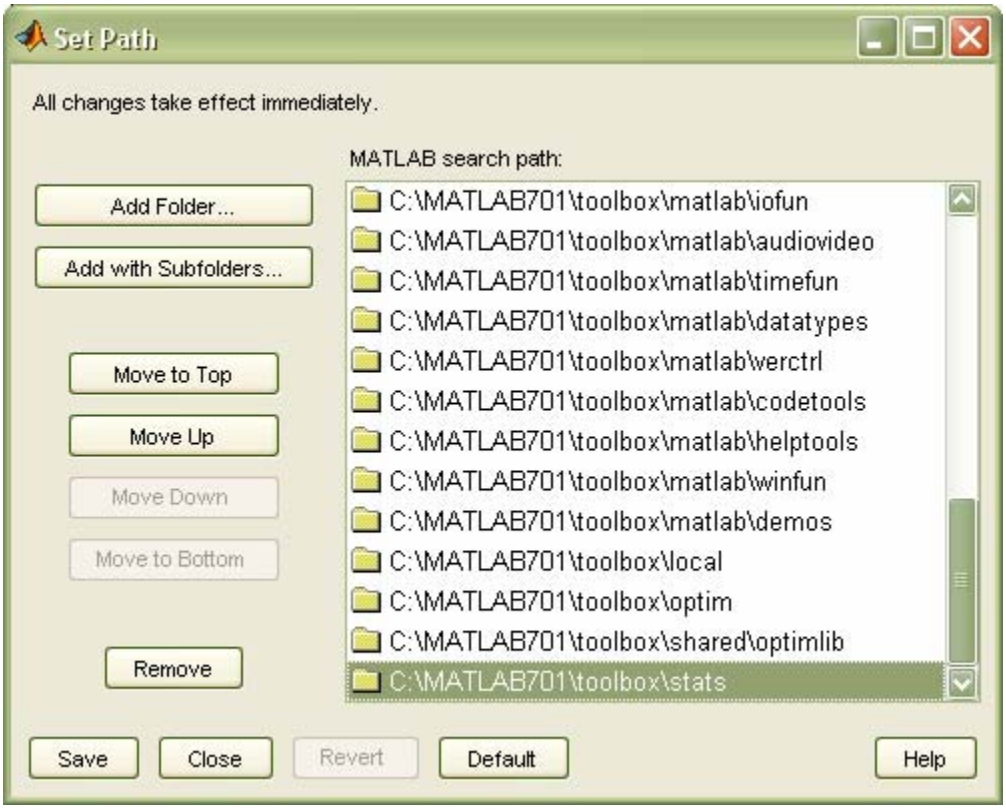
Pointillist is a software package consisting of a set of MATLAB codes available at http://labs.systemsbiology.net/bolouri/software/Pointillist. Pointillist, requires MATLAB R14 and the statistical toolbox Version 5 above.



Installation of the software can be done as follows:

1. Download the latest version of Pointillist (v2.1.1.zip) from the above website.
2. Unzip and copy the MATLAB codes into the directory of interest.
3. Within the MATLAB desktop, add this directory to the path by clicking “File”, “Set Path”, and then “Add Folder” in the Set Path window (see the figure below).

The software requires the MATLAB Statatistics toolbox. You can check if the toolbox is added to the path by clicking "File" and then "Set Path." You should see an entry ending in "toolbox\stats."



Note that all the codes were developed in Windows. Check if the MATLAB version is higher than R14 (all the codes were developed in R14. Thus, R13 below may produce errors). Please report any inquiries and errors at pointillist@systemsbiology.org.

Function Descriptions

Two kinds of MATLAB functions are included in Pointillist:

1. **Data Normalization**- the codes for normalizing the p-values (input data for the data integration programs below)

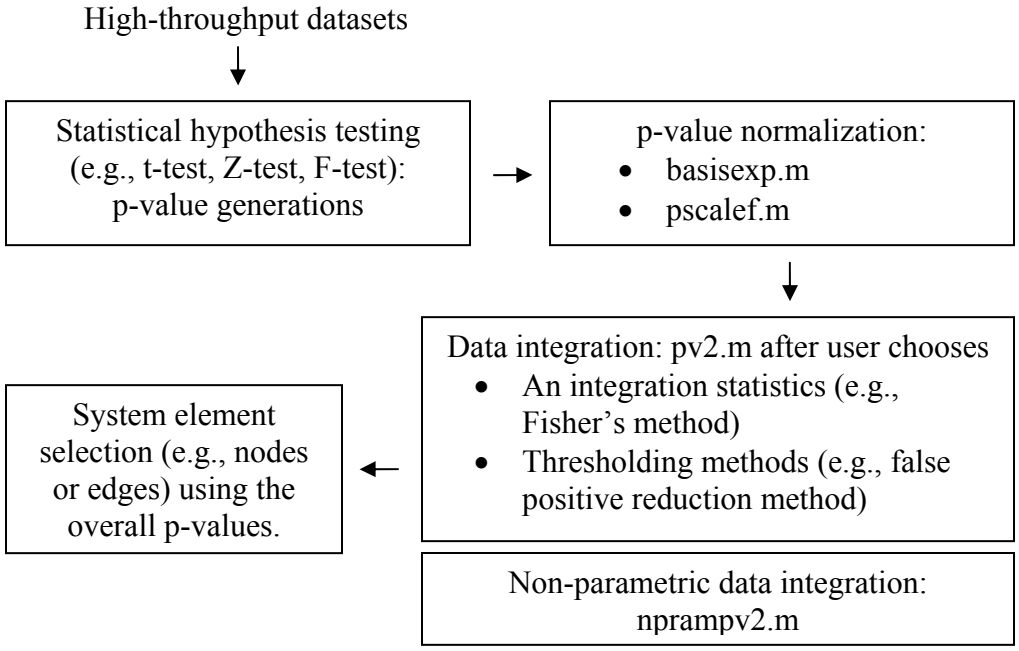
   - basisexp.m: select a dataset to be used for p-value normalization.
   - pscalef.m: quantile based normalization of p-values.

   These two functions should correct non-ideal distributions of p-values (see "Data Characteristics for further details).

2. **Data Integration**- the codes for various data integration methods mentioned in the papers.

- pv2.m: a wrapper to call esa.m and genwfunf.m depending upon the methods mentioned in the main text of the paper.
- esa.m: enhanced simulated annealing that searches for the optimal weights and parameters.
- genwfunf.m: select elements given a weight vector and alpha
- mcmc.m: Monte Carlo simulation to generate random numbers for each integration method (for example, Fisher's method).
- nprampv2.m: non-parametric method for integrating datasets.
- supsmu.m: non-parametric smoothing algorithm.

The following figure summarizes the procedure for the data integration using the above MATLAB codes.



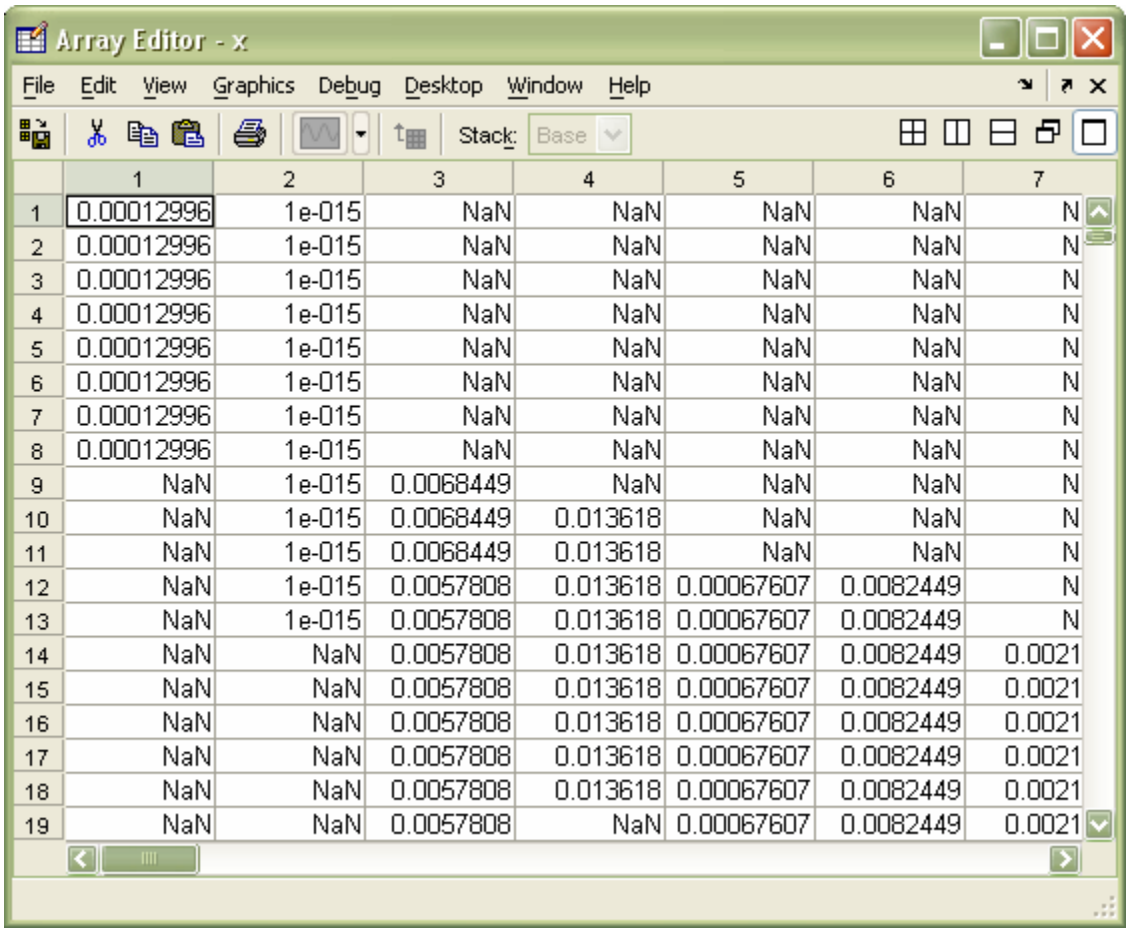
Data Characteristics

Input Data

The input data for pv2.m are p-values from hypothesis tests of each dataset. The input data should be saved to a file that can be loaded into the MATLAB workspace (for example, tab-separated file). The following figure shows the input data loaded from "input_pvalue.txt" into the MATLAB workspace by typing

```
>> x = load('input_pvalue.txt');% x would be an nxk p-value matrix,
where n and k are the numbers of the elements and the datasets,
respectively.
```

in the MATLAB command window.

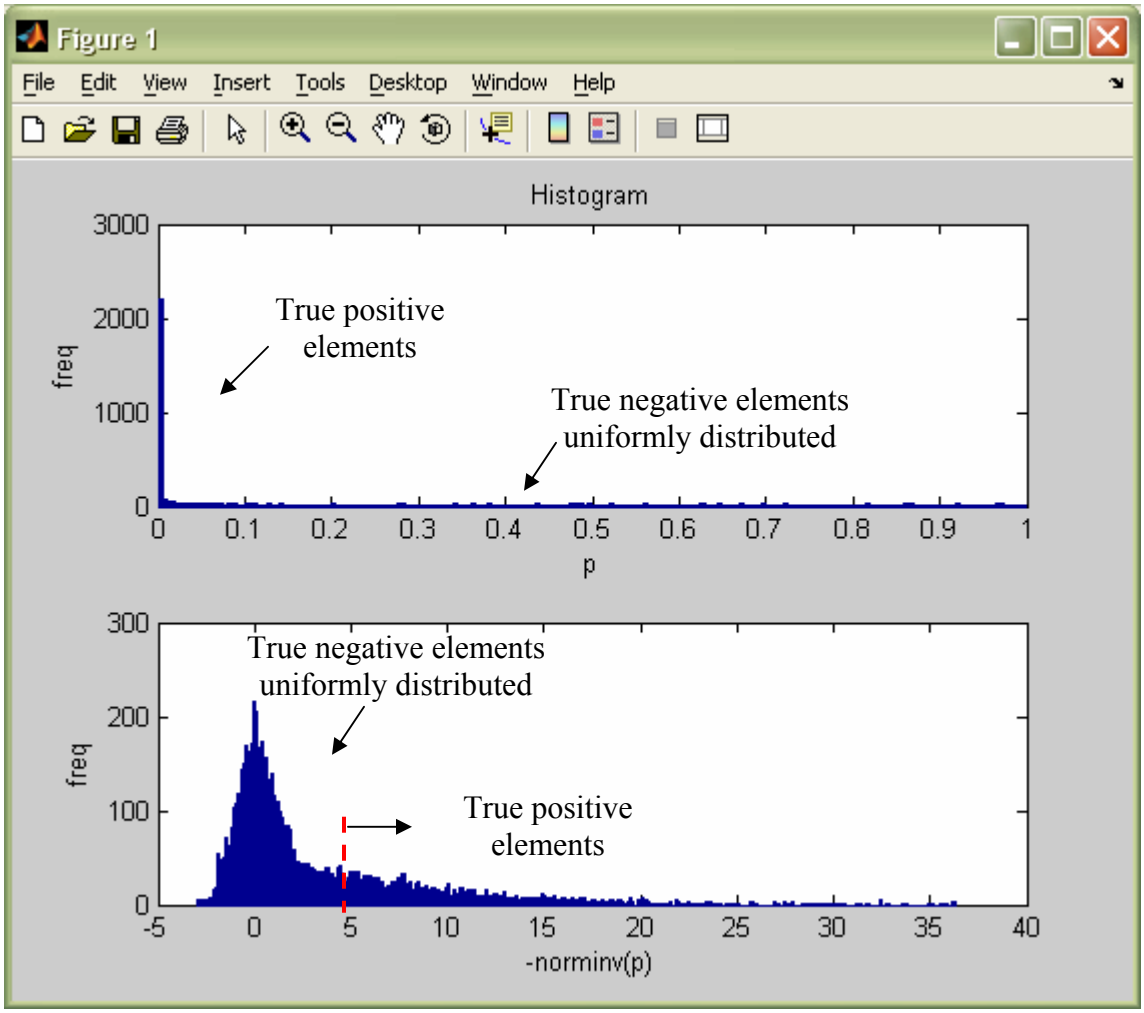


	1	2	3	4	5	6	7
1	0.00012996	1e-015	NaN	NaN	NaN	NaN	N
2	0.00012996	1e-015	NaN	NaN	NaN	NaN	N
3	0.00012996	1e-015	NaN	NaN	NaN	NaN	N
4	0.00012996	1e-015	NaN	NaN	NaN	NaN	N
5	0.00012996	1e-015	NaN	NaN	NaN	NaN	N
6	0.00012996	1e-015	NaN	NaN	NaN	NaN	N
7	0.00012996	1e-015	NaN	NaN	NaN	NaN	N
8	0.00012996	1e-015	NaN	NaN	NaN	NaN	N
9	NaN	1e-015	0.0068449	NaN	NaN	NaN	N
10	NaN	1e-015	0.0068449	0.013618	NaN	NaN	N
11	NaN	1e-015	0.0068449	0.013618	NaN	NaN	N
12	NaN	1e-015	0.0057808	0.013618	0.00067607	0.0082449	N
13	NaN	1e-015	0.0057808	0.013618	0.00067607	0.0082449	N
14	NaN	NaN	0.0057808	0.013618	0.00067607	0.0082449	0.0021
15	NaN	NaN	0.0057808	0.013618	0.00067607	0.0082449	0.0021
16	NaN	NaN	0.0057808	0.013618	0.00067607	0.0082449	0.0021
17	NaN	NaN	0.0057808	0.013618	0.00067607	0.0082449	0.0021
18	NaN	NaN	0.0057808	0.013618	0.00067607	0.0082449	0.0021
19	NaN	NaN	0.0057808	NaN	0.00067607	0.0082449	0.0021

Each row in the table represents a system element (e.g., genes, proteins, etc.) while each column is a dataset. In this p-value table, the value 0.00012996 at (4,1) is the p-value for the 4th system element (i.e., Prnp gene) and the 1st dataset (i.e., MotifLocator Transcription Factor Binding Site prediction). *NaNs* represent missing values in the corresponding dataset and system elements.

Data Structure

The data integration methods in Pointillist assume that p-values from each dataset are true: that is, the p-values for true negative data elements are uniformly distributed and uncorrelated across multiple datasets, while the p-values are correlated and are smaller than those for true negative elements. If the appropriate statistical hypothesis tests reflecting the data structures in individual datasets are performed, the conditions mentioned above should be met. The p-values for an example dataset are shown in the figure below (original p-value in the upper panel and Z score for the p-values (-norminv(p)) in the bottom).



However, the data structures of the datasets become complicated (or non-ideal) in practice due to many sources of experimental and technical variability. Thus, it is sometimes hard to find an appropriate hypothesis test providing the p-values meeting the above conditions. Therefore, the distributions of p-values should be studied first to determine if the distributions meet the conditions. Datasets that do not meet these

conditions can be normalized using the p-value normalization (see next section) to improve data integration accuracy.

P-value Normalization

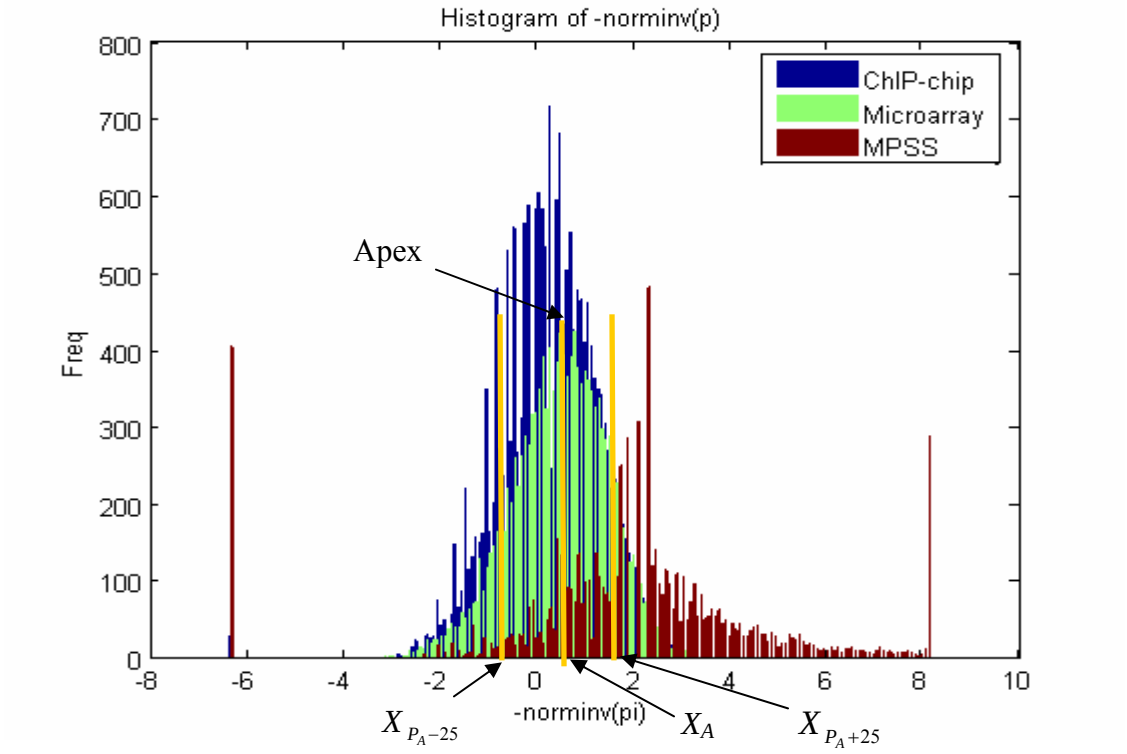
This normalization first finds the dataset with the best distributed statistic values (e.g., t values in t-test) or p-values. It can be done by typing

```
>> ind=basisexp(x,'T') %ind is the column number for the identified
dataset with the best distributed statistic values.
```

where x is the statistic value table or the p-value table as described above (see *Input Data*). The second input argument indicates that the table x is either statistic value ('T') or p-value ('P'). The criterion for selecting the best distributed dataset is "a skewness measure" defined as follows:

For the example input data (p-values; `basisexp(x,'P')`),

1. Find the apex in the histogram of each data and compute the percentile for the apex (P_A).
2. Calculate the x-values (X_A; -norminv(p)) for two percentiles P_A-25 and P_A+25. If $P_A < 25$, P_A /2 and P_A *3/2 are used instead. Also, if $P_A > 75$, $P_A - (100\text{-}P_A)/2$ and $P_A + (100\text{-}P_A)/2$ are used.
3. Compute the following ratio as the skewness measure: $S_{cri} = \left\| \frac{X_A - X_{P_A-25}}{X_A - X_{P_A+25}} \right\|$



For these example datasets, the measure for each dataset was computed as 0.9572, 1.3501, and 1.1430. Thus, the first dataset (ChIP-chip) whose measure is closest to 1 was chosen as the best distributed dataset (in this case, the output from `basisexp(x,'P')` "ind" = 3). Note that this selection method assumes that the majority of data elements (say more than 50%) are true negatives. Currently, when statistic values are the input data, the skewness measure can be only applied for the cases the statistic values follow normal-like distributions (e.g., lognormal or t-distribution).

Next, all the p-values are normalized against the p-values of the chosen dataset by typing

```
>>[y,z]=pscalef(x,ind,'P',tail)%y is the normalized p-values and z is -
norminv(y).
```

where tail = 1, -1, and 2 represent right sided and left sided one-tailed test, and two-tailed test, respectively. "ind" is the output from `basisexp(x,'P').` This normalization uses modified quantile normalization as follows (for the details of quantile normalization, see Yang et al. (2002) *Nucleic Acids Res* **30,** e15):

1. If the input data are p-values, convert them into Z scores using –norminv(p).
2. Sort the p-values of the chosen dataset.
3. Quantile normalize the p-values from all the other datasets using the sorted p-values as the mean p-values (see Yang et al.). When the data size is different (i.e., a different number of missing values), find the p-values with the nearest quantile values from the sorted p-values. Note that only the non-missing values are being normalized.
4. Scale the normalized p-values (***x***) using the given tail information using the mean (*m*) of the non-missing values and the mean (*s*) of X_{75}-X_{50} and X_{50}-X_{25}:

$$y = \frac{(x - m)}{s}$$

Data Integration

Overview

A major challenge in systems biology is that technologies which globally interrogate biological systems have inherently high false positive and false negative rates, thus each data type alone has a limited utility. The integration of data from different sources provides an effective means to deal with this issue, by reinforcing bona fide observations and reducing false negatives. Moreover, because different experimental technologies provide different insights into a system, the integration of multiple data types offers the greatest information about a particular cellular process. For example, gene perturbation experiments (e.g. knock outs, or RNAi) reveal relationships between genes that may imply direct physical interactions or indirect logical interactions. On the other hand, ChIP-chip data can reveal direct protein-DNA interactions, or co-factor associations with bound transcription factors. Combined together, these technologies can provide a much more detailed view of a transcriptional regulatory network than either alone. As a result, systems biology is predicated on the integration of experimental data from an ever increasing number of technologies, such as gene expression arrays, proteomics, and ChIP-chip assays. Integration achieves one of the most important imperatives of systems biology, namely it reduces the dimensionality of global data to deliver useful information about the system of interest.

Pointillist provides non-weighted and weighted data integration methods using the following statistics that have been used to combine multiple pieces of evidence (i.e., multiple sets of p-values) in meta-analysis. In the weighted integration methods, the weights are determined in such a way to maximize the overall statistical power of the weighted sum of non-linearly transformed p-values (see the statistics below):

- Fisher's weighted F: $F_w = -2\sum_{i=1}^{k} w_i \ln(p_i)$.
- Mudholkar-George's (MG) weighted T: $T_w = -\sqrt{\frac{15k+12}{(5k+2)k\pi^2}}\sum_{i=1}^{k} w_i \ln\left(\frac{p_i}{1-p_i}\right)$.
- Liptak-Stouffer's (LS) weighted Z: $Z_w = \frac{1}{\sqrt{\sum w_i^2}}\sum_{i=1}^{k} w_i N^{-1}(1-p_i)$.

In addition to these parametric data integrations where the decision boundaries are set by the functional structures of the integration statistics used, Pointillist also provides a non-parametric version of data integration (for further details, see Hwang D. et al. (2005) "A data integration methodology for systems biology." *PNAS* **102** (48), 17296-17301).

Pointillist also provides several ways to find an optimal threshold (alpha) to select a set of true positive elements: 1) thresholded method; 2) mixture modeling; and 3) non-parametric method. The choice of these methods and the theoretical conditions that each

method performs best will be discussed in the following sections and the "Practical considerations" section.

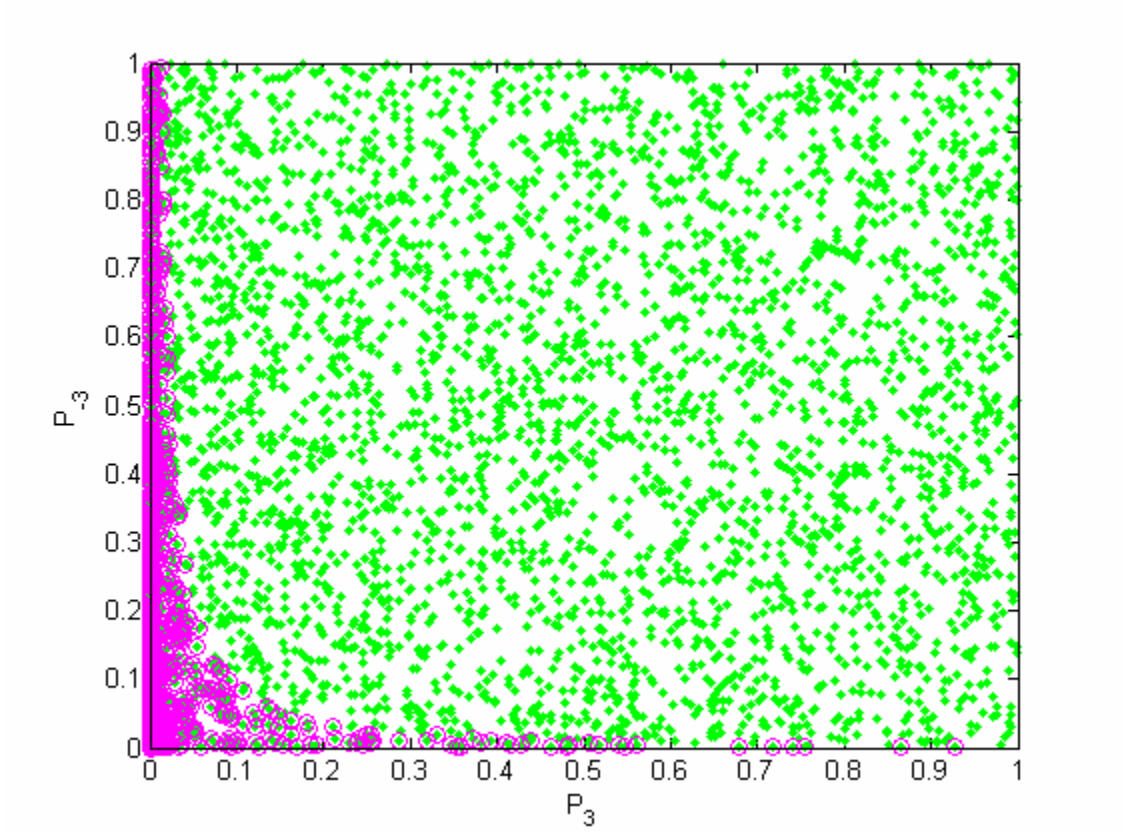
The data integration methods based on meta-analysis approaches (i.e., combining p-values) in Pointillist are different from Bayesian approaches where the probabilities (or densities, not p-values) are being combined according to Bayesian rule. In many cases of Bayesian methods, they require a good quality of training datasets to estimate the model parameters. If the training sets are too sparse due to the lack of the sampling of true positive elements, the probability estimated using the training datasets will be inaccurate. In contrast, the accuracy of p-values from hypothesis tests largely depends on the size of the sampled true negative elements and their measurement accuracy. In typical high-throughput data, the size of true negative elements is large compared to the portion of true positive elements and also their p-values normally distinguish them from true positive elements if the technologies are appropriate. Also, p-values (the cumulative density) are in general more robust measures than the probabilities (the probability density) being used in Bayesian methods.

Non-weighted Methods

Although Pointillist was developed for weighted data integration, it also includes non-weighted integration methods. They can be done by typing

```
>> opt = 1; %when Fisher's method is used
>> [y,ind]=nwpv2(x,opt); %where x is the p-value matrix (nxk for n
system elements and k datasets), y is the overall p-value (nx1), and
ind is the set of selected data elements using p=0.05.
```

Then, the following figure will appear where P_3 represents p-values for the dataset 3 and P_{-3} represents the collective p-values for the other datasets. The magenta circled data elements are selected using the p=0.05 cutoff.



Weighted Methods with p = 0.05 cutoff

This weighted integration method selects the data elements using the typical threshold value p =0.05, not using an optimal threshold value found from the methods described in the following sections. This method performs best when true negative elements are distributed uniformly while true positive elements are distributed non-centrally with a reasonable effect size (i.e., mainly distributed along the axes and around the origin in the above figure; see the paper for further details). However, when the true positive elements are distributed distantly from the origin due to various reasons (e.g., low statistical power in the measuring technology, such as low abundant transcripts that microarray cannot accurately measure the abundance differences or the transcripts containing palindromic sequences which MPSS is likely to sequence incorrectly), it increases the false positive error rate. Also, when the true negative elements are uniformly distributed, but densely distributed around the origin or along the axes, this will increase the false positive error rate.

For a set of the example datasets generated as described in Hwang D. et al. (2005) "A data integration methodology for systems biology." *PNAS* **102** (48), 17296-17301, the weighted method can be carried out by typing

```
>> opt = 1; %when Fisher's method is used
>> meth = 1; %weighted method with p=0.05;
```

```
>> y=pv2(x,meth,opt); %where x is the p-value matrix, y is a MATLAB
structure variable storing all the outputs.
```

First, the parameters stored in the "param" MATLAB structure variable will be displayed on the console.

```
param =

      H0y: [19997x2x5 double]
     meth: 1
    alpha: 0.05
      mvp: []
```

where H0y is the Monte Carlo simulated p-values sampled for the background element distribution (the H0 hypothesis distribution) and mvp is the p-value for the missing values (normally NaN if there are NaNs in the datasets; see *Input Data* above).

Then, the Enhanced Simulated Annealing (ESA) begins as follows:

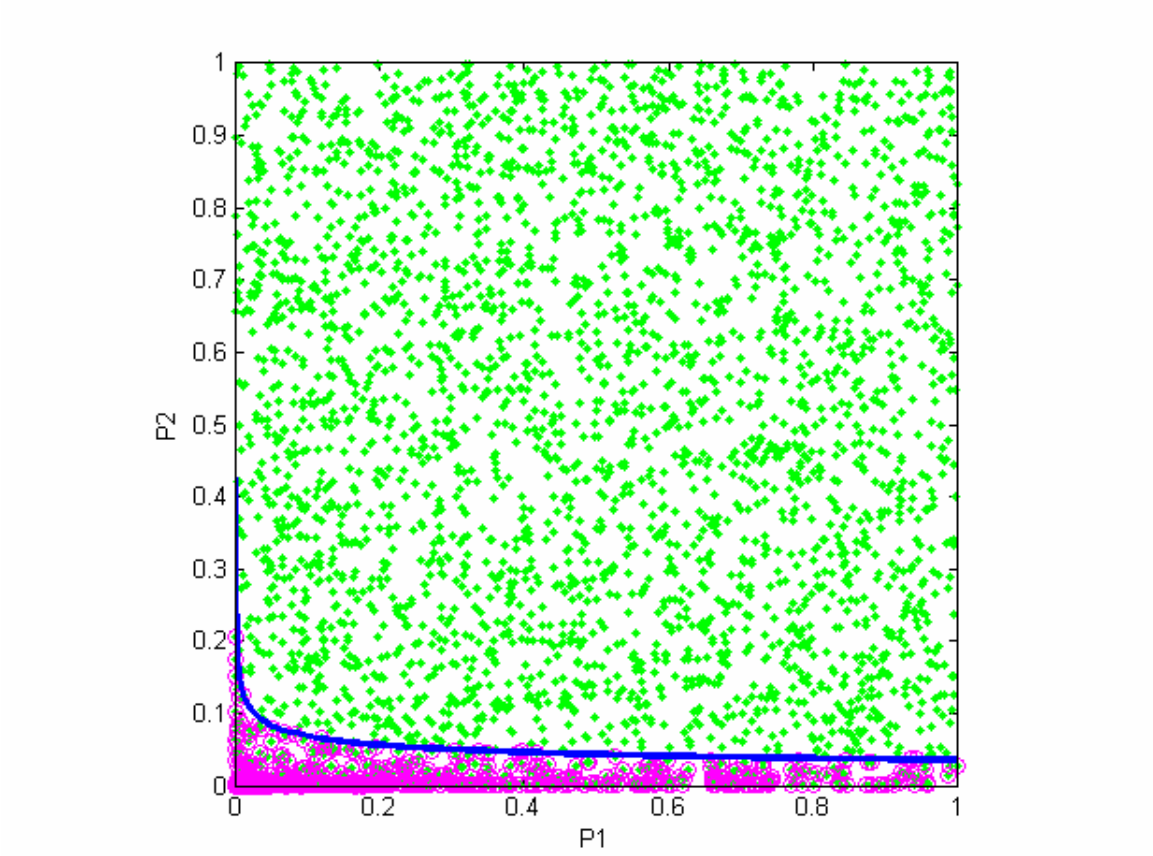
```
E_old = %Initial energy (objective function) for the initial decision
variables.

       -2981

iter = %iteration number during ESA procedure.

     1

Elowst = % Energy (objective function) being minimized during ESA.

       -3007

thetaopt = % Decision variables which decreased the energy above from
E_old to Elowst.

      0.26247
```

…

Finally, the selected elements (marked by magenta circles) with the weights and alpha = 0.05 are displayed as shown in the figure below. The output MATLAB structure variable y includes 1) the weights (y.wopt1), 2) the number of selected elements (y.Nsel1), and 3) another MATLAB structure variable (y.sel1) including the overall p-values (y.sel1.p) and the index of the selected elements (y.sel1.ts):

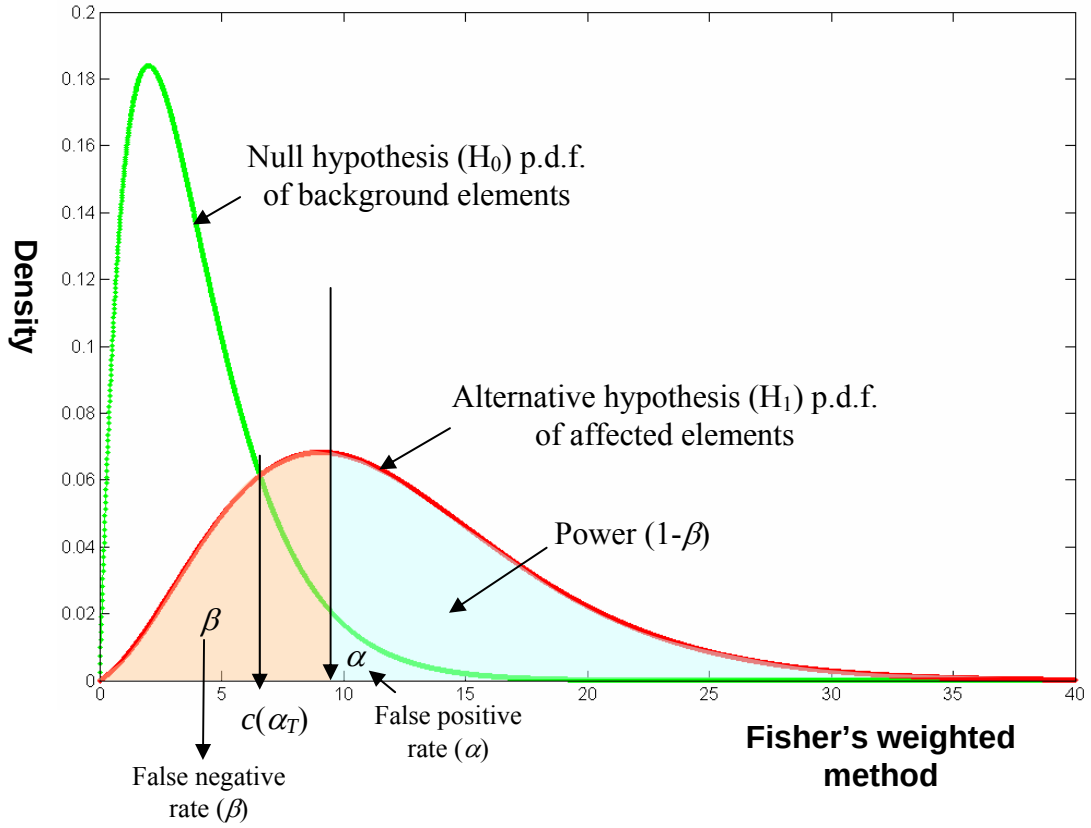
```
y =

    wopt1: [0.21741 0.78259]
    Nsel1: -3008
     sel1: [1x1 struct]

y.sel1 =
```

```
ts: [3008x1 double]
 p: [6000x1 double]
```



False Positive Reduction Methods

This integration method, also referred to as *Thresholded Method* throughout this Tutorial and Users' Guide, optimizes both the weights and the threshold value in an iterative manner as described in the paper. The choice of the α-threshold is constrained by the opposing needs to minimize both false positives (1- confidence) and false negatives (1- statistical power). That is, in the figure below, it can be seen that decreasing the false positive error rate (α) results in increasing the false negative error rate (β): the opposite case is also true. To effectively find the optimal trade-off between the two opposing error rates, it is necessary to know the two distributions for the true positive (H_1) and negative elements (H_0) because the optimal threshold (α) would be the right-hand tail of H_0 measured from the point where H_1 and H_0 PDFs meet (marked as $c(\alpha_T)$ in the figure below) See further details in the paper. In this and the following sections, we provide several ways where we indirectly or directly estimate these two distributions.



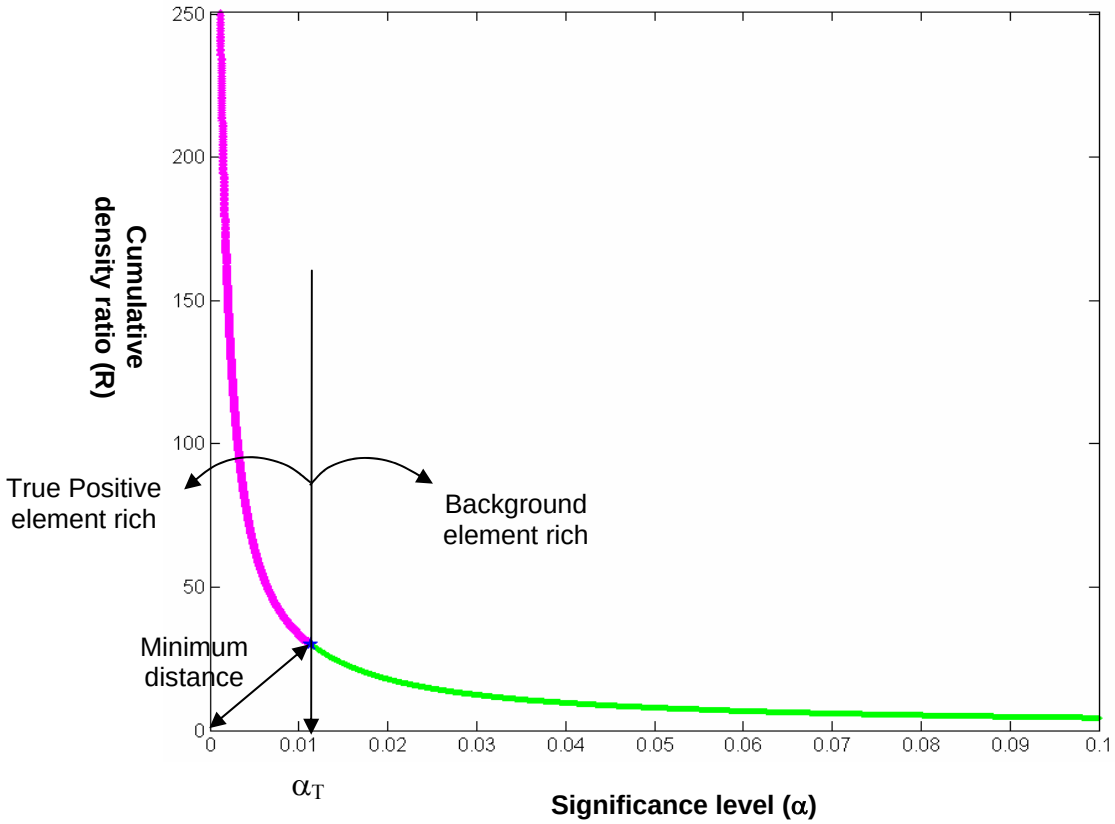
In this section, we do not intend to estimate the H_1 distribution (see "*Mixture Modeling*" that involves determining the H_1 distribution), but use the overall distribution of the observed data. This is based on the observation that that true positive data elements cluster near the axes while the remaining data elements tend to be uniformly distributed. Therefore, one way to select α_T would be to find the value of α for which the resulting decision boundary best separates uniformly distributed data from the non-uniform distribution of data elements near the axes. For a given α, the cumulative density of the data elements measures the fraction of data elements selected. The cumulative *expected* density of the background (uniformly distributed) data elements is α. Thus to compute α_T, we calculate the ratios of cumulative observed densities to cumulative expected densities as α is increased from zero. For a given weight vector, the ratios of cumulative observed densities to cumulative expected densities using the real data distributions is computed in the range of α between 0 and a:

$$R(\alpha = a) = \frac{1 - D^{cdf}\left(S_w = c(a)\right)}{1 - H_0^{cdf}\left(S_w = c(a)\right)} = \frac{1 - D^{cdf}\left(S_w = c(a)\right)}{a}$$

where $D^{cdf}(\cdot)$ is the c.d.f. of the observed data distribution, and $c(a)$ corresponds to the integration statistic S_w (e.g. F_w) value when $\alpha = a$. Note $c(\alpha = 0)$ is infinite. As a decreases, the decision boundary moves towards the origin in p-value space. When there are true positive elements, this results in a large cumulative density ratio because the

expected density is small but the observed density is large. As *a* increases, however, the ratio decreases because the observed density slowly increases relative to the expected density. The plot of the ratio versus *a* shows that the curve can be split into two regions, one rich in true positives and the other in background data elements (see the figure below). α_T is the x-axis value of the point nearest to the origin as shown below. Finally, since the data integration weights are calculated for a given α, it is necessary to re-calculate the weights for α_T. For these recalculated weights, another α_T is determined. This procedure is repeated until α_T converges (see the paper for further details).

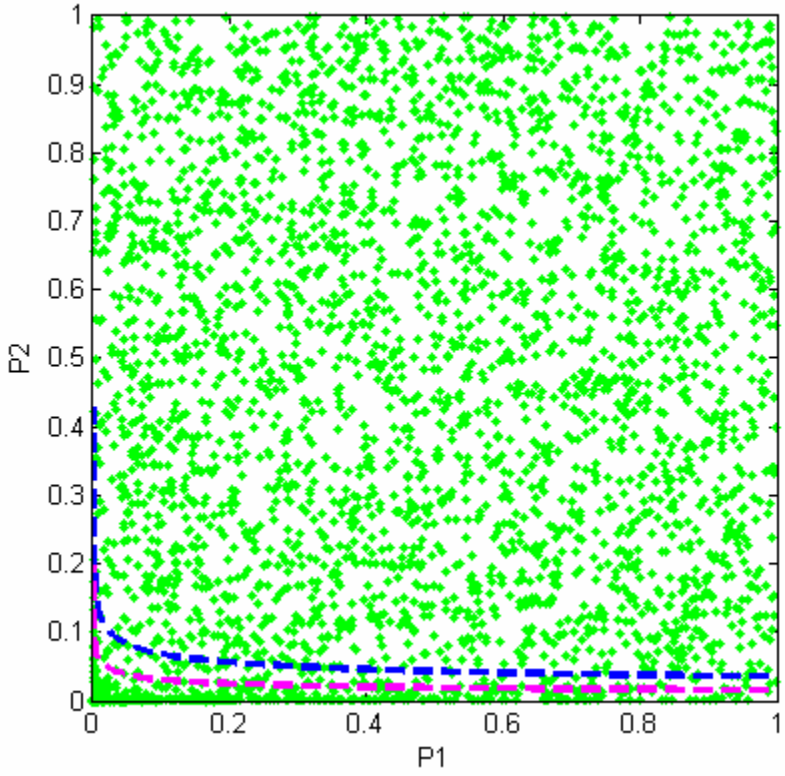
In many cases, this method for most of high-throughput datasets usually determines an α as the area smaller than that from the apex of the H_1 distribution (see the previous figure) to positive infinite. It therefore focuses on reducing false positives rather than false negatives. In most of data integration problems in systems biology, our main interest could be just the reduction of false positives because of the small fraction of true positive elements in the data and also several practical reasons. For example, with a large set of genes/proteins, it is difficult to explore the network space defined by those genes. Also, it would be good for proposing validation experiments for the network predictions. This method performs best when the portion of the H_1 distribution from the apex to positive infinite (see the figure above) is sufficiently defined (i.e., a sufficient number of the true positive elements are sampled and detected by measuring technologies). This situation will clearly define the magenta line in the true positive element rich region in the figure below. See the undersampling issues for this method in the "Practical Issues" section.



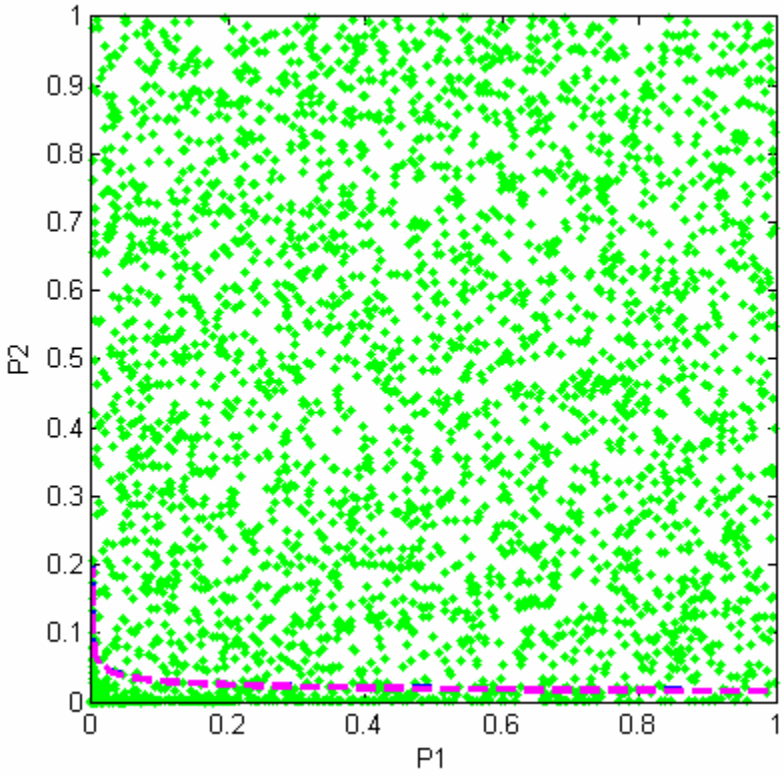
In Pointillist, this method can be executed by typing

```
>> opt = 1; %when Fisher's method is used
>> meth = 2; %thresholded weighted method;
>> y=pv2(x,meth,opt); %where x is the p-value matrix, y is a MATLAB
structure variable storing all the outputs.
```

This performs the weighted method (meth =1) first (by default) as described in the previous section and then the thresholded weighted method to compare two methods together. In the figure below, the magenta line shows the decision boundary for the α_T=0.013 obtained from this method while the blue line shows the decision boundary from the weighted Fisher's method with $\alpha = 0.05$.



In the second iteration, the α_T value did not change significantly compared to the first iteration as shown in the figure below (see the magenta and blue lines corresponding to the new and old α_T values in two consecutive iterations). Thus, the iteration is terminated. See also the undersampling issues for this method in the "Practical Issues" section, in which cases the iteration does not converge for datasets with improper data structures for this method.




```
>> y

y =

    wopt2: [0.2172 0.7828]
    Nsel1: -3008
     sel2: [1x1 struct]

>> y.sel2

ans =

        ts: [3008x1 double]
         p: [6000x1 double]
       t2s: [2905x1 double]
    alpha2: 0.0230
        p2: [6000x1 double]
```

The output MATLAB structure variable (y) includes now the overall p-values and a new alpha in y.sel2.

Mixture Distribution Models

The method above (False Positive Reduction Method) does not directly estimate the H_1 distribution when determining the α_T, resulting in focusing on reducing mainly false

positives. When the true positive elements are sufficiently samples to define the clear H_1 distribution, however, estimating the H_1 distribution from the data based on the following procedure might provide a more accurate α_T to minimize both false positive and negative error rates:

1) For an initial value of α (e.g. 0.05) determine the weight vector as described in "Weighed integration of p-values".
2) Assume that the full set of observed data elements divides into two sets H_1 and H_0.
3) The Probability Density Function (PDF) of observed data elements can be approximated by a "mixture model" M: $PDF(M) = (1-\theta)PDF(H_0) + \theta PDF(H_1)$, where $0 \leq \theta \leq 1$ is the proportion of H_1 in M.
4) For a given set of weights, the PDF of H_0 can be evaluated numerically by Monte Carlo sampling of a uniform distribution (or the corresponding χ^2, T, or Z distributions).
5) Guess an initial value for θ.
6) Estimate the PDF of H_1 as a normal (or gamma) distribution that best satisfies the relationship in (2) above using Enhanced Simulated Annealing.
7) Determine α_T as the right-hand tail of H_0 measured from the point where H_1 and H_0 PDFs meet.
8) Repeat steps 1 (using $\alpha=\alpha_T$) to 7 until α_T estimates converge. Finally, the significant elements are selected using the decision boundary given by α_T.

See the paper for further details.

This method performs best when the H_1 distribution is well-defined and the true negative elements in the data are uniformly distributed (as in the theoretical H_0 distribution). The latter is true because Pointillist explicitly fits both the H_0 and H_1 distributions to the real data distribution. Note that the previous method does not fit either of the distributions, and it does not therefore depend on the H_0 distribution at all. But since it compares the theoretical H_0 distribution to the real distribution, the right tail of the H_1 distribution should be clearly defined as explained in the previous section. As a result, this method may not converge or provide mixture modeling if there is non-ideality (resulting in the H_0 distribution far from the uniform distribution, such as correlation of true negative elements across datasets) in the distribution of the true negative data elements. See the further details in non-ideal distributions of the true negative elements in the "*Practical Issues*" section. Many of such issues are fixed by the p-value normalization (see "*P-value Normalization*").

Also, note that the current program uses "normal distribution" instead of "gamma distribution" to speed up the computation. The choice between normal and gamma distribution depends on the data structure defining the H_1 distribution. To use "gamma distribution", activate Line 145 while deactivating Line 144 in genwfunf.m by deleting the comment symbol "%" in front of the line. From our experience, normal distribution speeds up the computation while not decreasing the goodness-of-fit much (it sometimes gives a higher goodness-of-fit compared to the gamma distribution fit depending on the data structure).

This method can be implemented by typing

```
>> opt = 1; %when Fisher's method is used
>> meth = 2; %thresholded method;
>> y=pv2(x,meth,opt); %where x is the p-value matrix, y is a MATLAB
structure variable storing all the outputs.
>> meth = 3; %mixture distribution model based weighted method;
>> y=pv2(x,meth,opt,y);
```

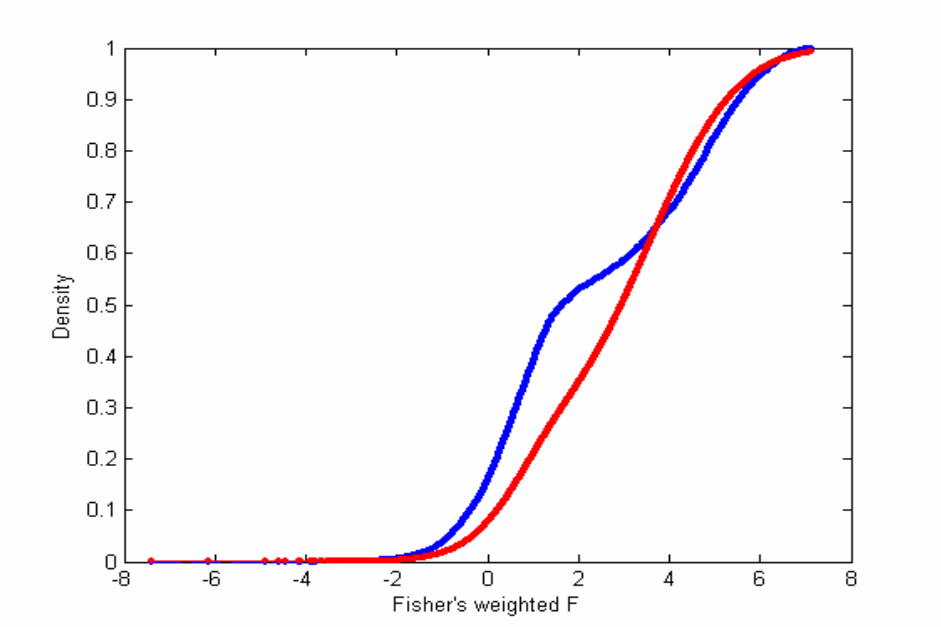
By adding the output variable (y) from the "False Positive Reduction Method", the program uses the lower and upper bounds of the decision variable: theta = [mean, standard deviation, and the H_1 fraction in the mixture model] when "normal distribution is fit (when "gamma distribution" is fit, theta = [a, b, and the H_1 fraction in the mixture model], where a and b are parameters for the gamma pdf).

The following iterations will be displayed.

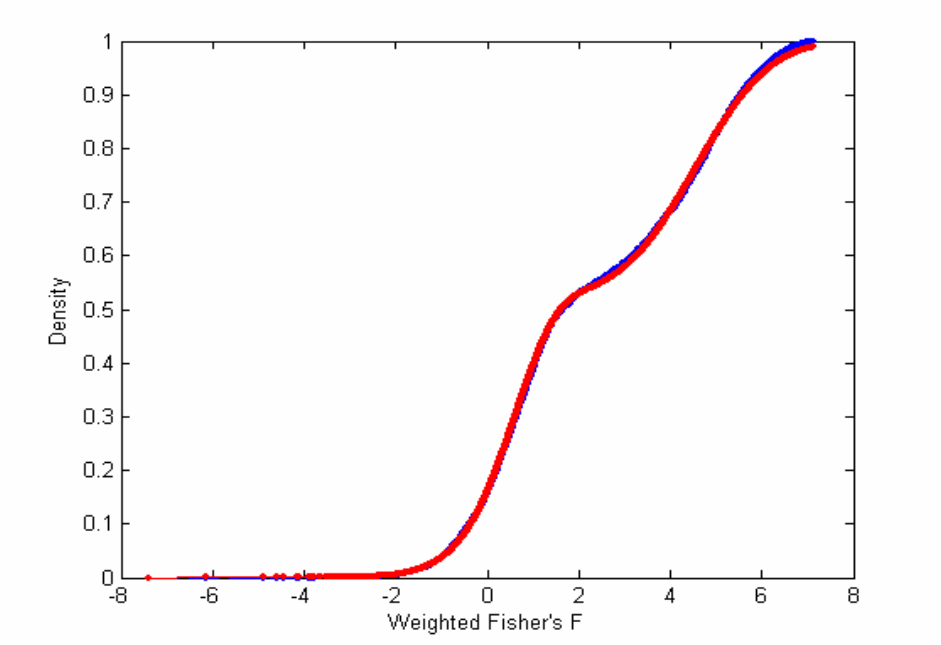
```
theta0 =

    2.5000    2.5000    0.4842

E_old =

   96.8510

iter =

     1

Elowst =

   73.8117

thetaopt =

    2.4094    1.4818    0.6894

Elowst =

   68.7801

thetaopt =

    3.5265    1.5542    0.7768

...

Elowst =

    0.1644

thetaopt =
```

```
4.5570    1.2881    0.4764
```

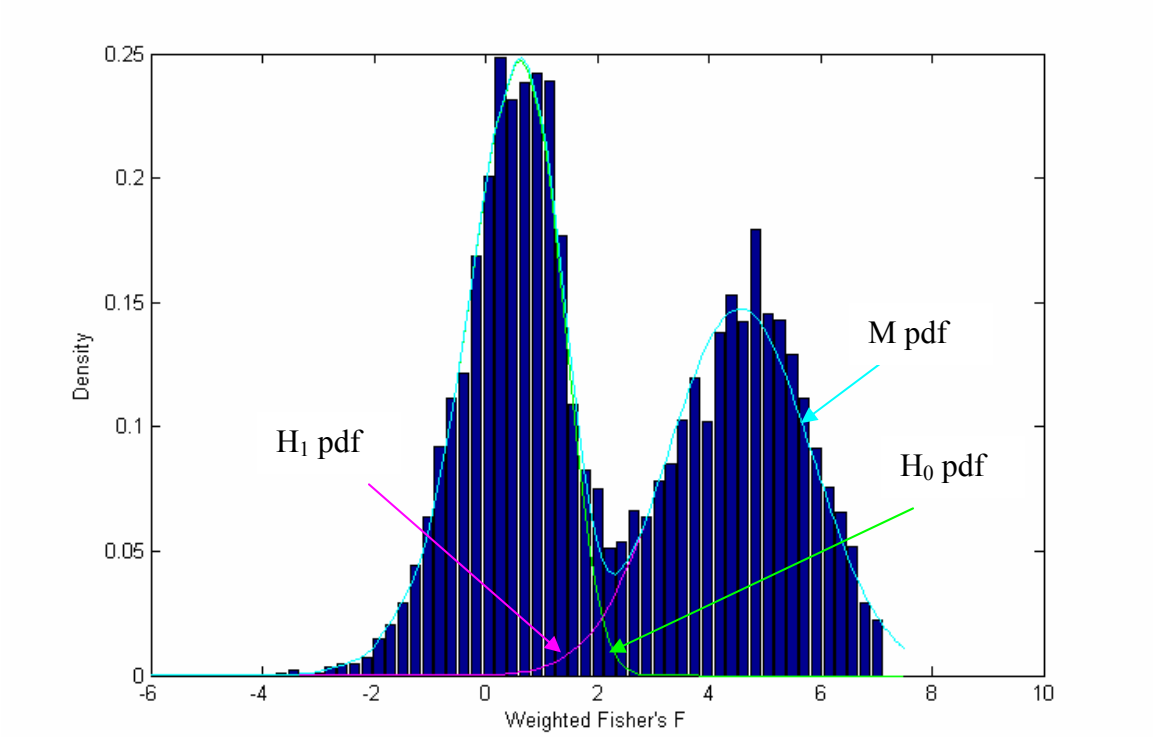
It can be seen that the parameter is being updated as the iteration goes. At the same time, how the new decision variables change the mixture distribution model fit is displayed as shown in the figure below (the blue line shows the cdf of the real data while the red shows the mixture cdf). Note that the mixture modeling is performed at the cdf, not the pdf because of the uncertainty in the choice of bin size when the pdf based mixture modeling is done).



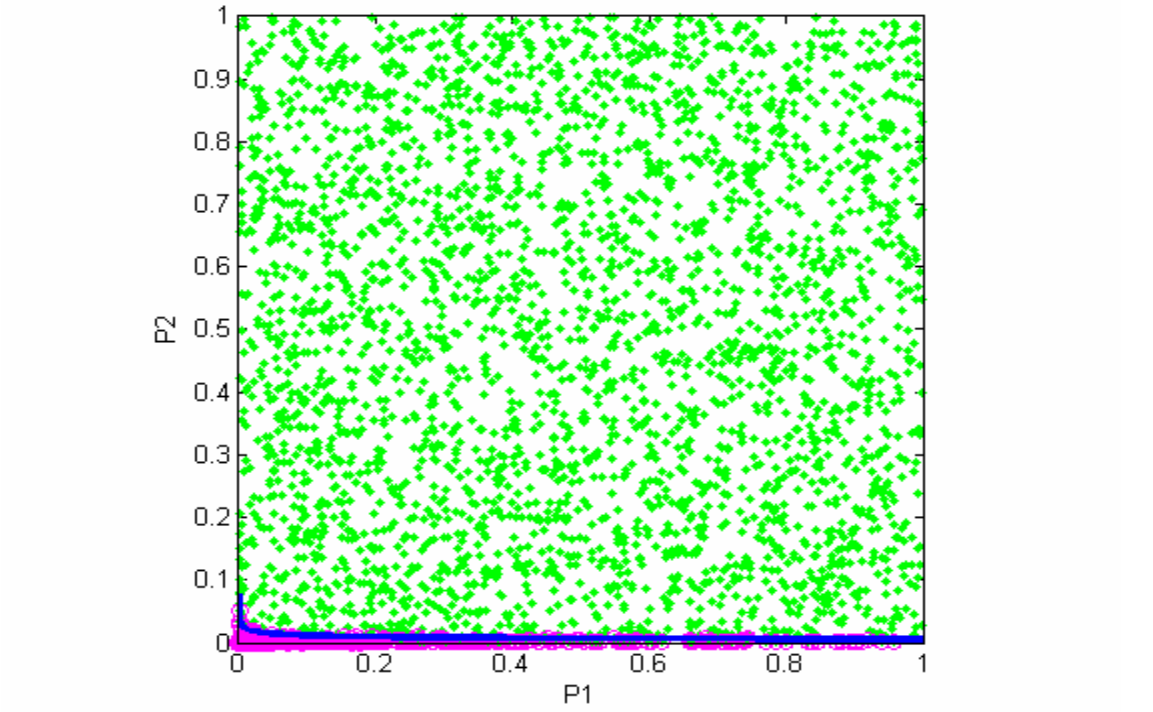
When the iteration is terminated, as shown below, the two cdfs of the real data and the mixture model should overlap with each other, showing a good fit.



Also, after the mixture modeling, how well the mixture model matches at the pdf level will be displayed using the normalized histogram of the real data and the estimated H_0 and H_1 pdfs.



The resulting decision boundary will be also displayed.



The output MATLAB structure variable includes the overall p-values and the index of the selected elements as follows:

```
y =

    wopt2: [0.2172 0.7828]
    Nsel1: -3008
     sel2: [1x1 struct]
        e: 4.5570
       df: 1.2881
    H1frc: 0.4764
    Nsel3: -2807
     sel3: [1x1 struct]
    wopt3: [0.2172 0.7828]

>> y.sel3

    ts: [2807x1 double]
     p: [6000x1 double]
```

The symbols "e" and "df" represent the mean and the standard deviation when "normal distribution" is used (a and b when "gamma distribution" is used).

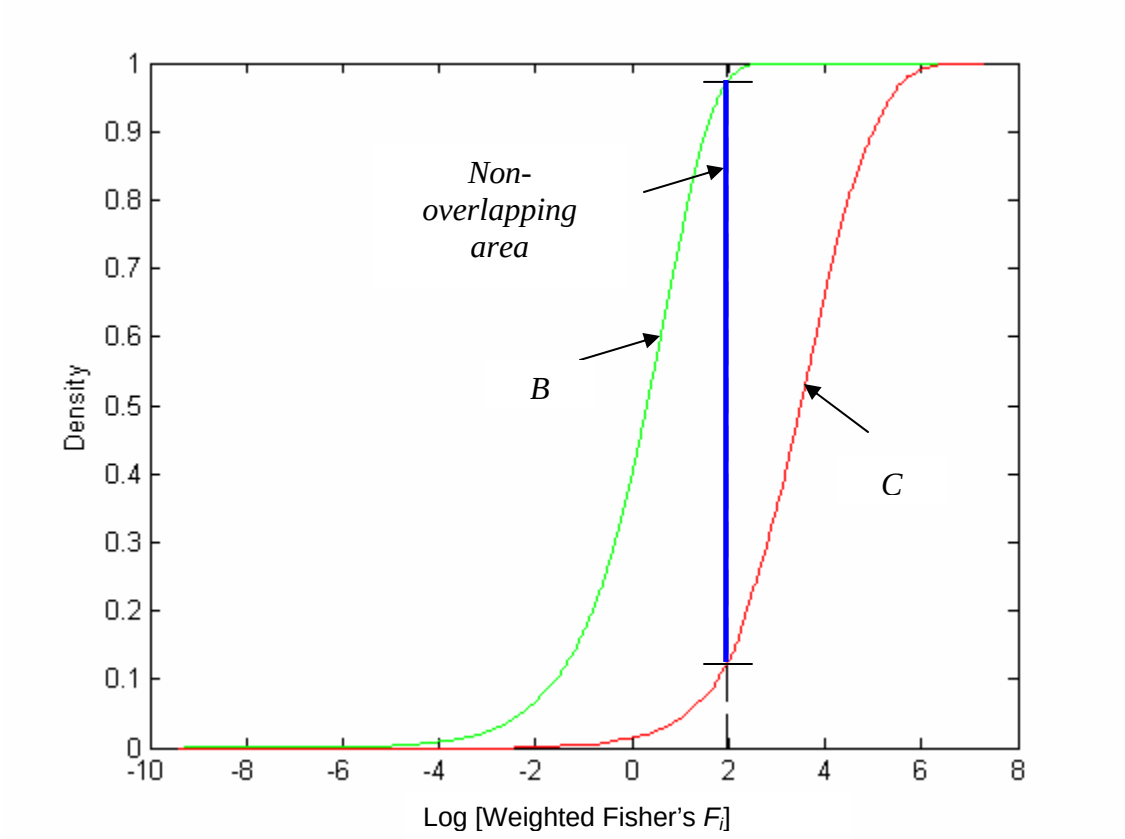
Non-parametric Methods

The above methods are all based on smooth parametric decision boundaries derived from assigning a reliability parameter (the weight) to each type of data. In practice, there are many sources of error for a given measurement error, in turn affecting the resulting p-values in an irregular manner. In such cases, smooth parametric decision boundaries may be unable to accommodate these irregularities. To allow estimation of non-smooth decision boundaries, we have developed a heuristic non-parametric method, as follows:

1) Construct a first-pass set of candidate true positive data elements C whose p-values are less than a threshold α_T in at least one dataset. For a given α_T (see below), the initial set of elements are selected using the union method:

$$C = \bigcup \{\, p_i < \alpha_T \}, i = \{1,2,\ldots,k\}.$$

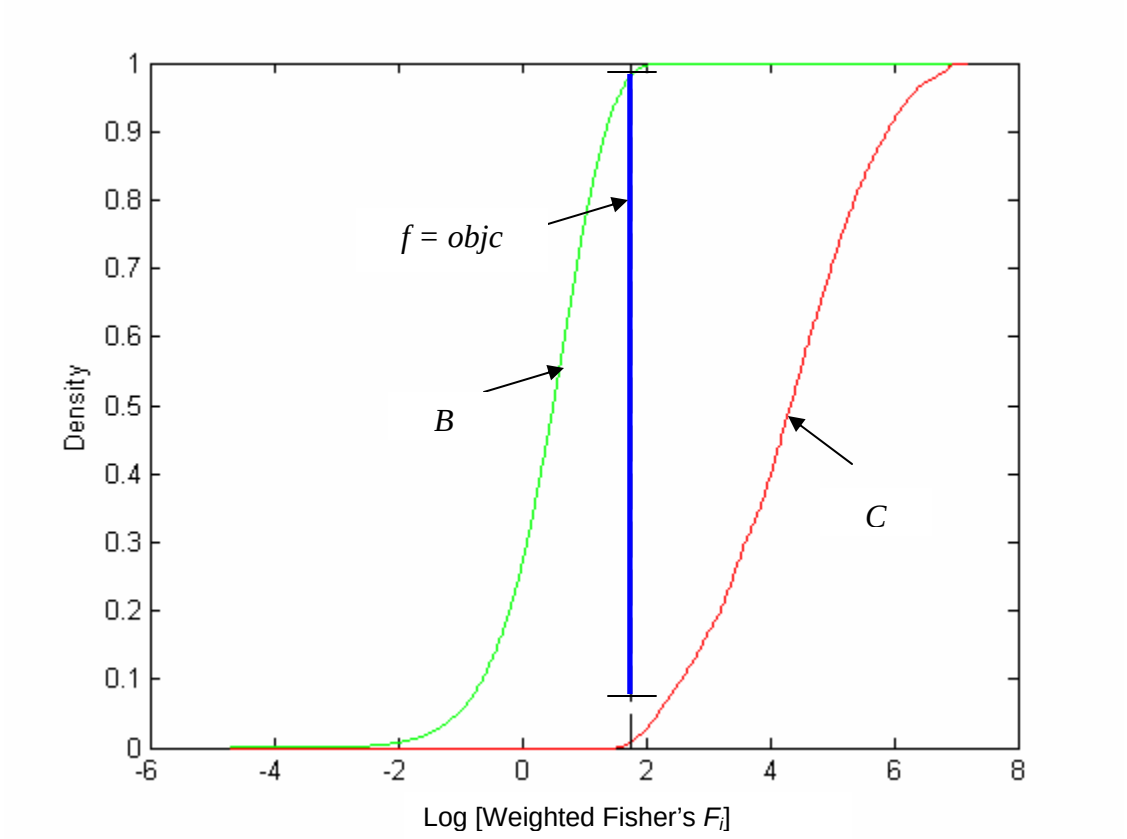
   The remaining elements are grouped into the complement dataset C^c.
2) Generate a set of random background elements (B).
3) Compute a weight for C and B proportional to the non-overlapping area between the two PDFs of transformed p-values of the elements in C and B: $w_i = \underset{S_i}{Max}\{CDF(S_i, B) - CDF(S_i, C)\}$, where S_i are the transformed p-values for dataset i (e.g. when Fisher's method is used, $S_i = -2\ln(p_i)$).



These non-overlapping areas (ar) are further normalized against the largest non-overlapping area as follows.

```
[kk,kki]=max(ar);
w=zeros(1,n);
for i=1:n
    w(i)=-max(cums(:,i)-cums(:,kki));
end
w=w+ar(kki);
w=w/sum(w);
```

4) Compute the combined p-value statistic (S_w).
5) Quantify the statistical power of the current memberships of C and B as the non-overlapping area between the corresponding S_w PDFs of C and B: $f = \underset{S_w}{Max}\{CDF(S_w, B) - CDF(S_w, C)\}$.



6) Remove putative false positive elements with high S_w values from C and add them to C^c if the non-overlapping area (f) is less than a user-specified threshold (e.g. $f_c = 0.99$).
7) Repeat steps 2 through 5 using the new C and C^c sets until f reaches f_c. Finally, once the iteration process is terminated, potential false negatives are identified as elements in C^c whose S_w values exceed that of the point where the S_w PDFs of C and C^c meet. These elements are then added to the final set of candidate true positive data elements C.

The above procedure is essentially the same as the parametric method described in the preceding section. The main difference is that, rather than moving a decision boundary (as in previous methods), here we move individual data elements between C and C^c. The final decision boundary is irregular because elements in a different part of the p-value space are removed as the weights change in each iteration. In order to speed up the optimization process and avoid potential local minima in the search process we have limited our optimization process to pruning. To be sure that the final set C includes as many of the true positive data elements as possible, we initialize C using a value for α_T that assures the inclusion of most true data elements at the expense of many false positives. These false positives are then removed iteratively. An adaptive rule is used to determine the fraction of C removed in step 5: $C_f = \min\left[\left(f_c - f\right)/f_c, 0.01\right]$. This removes 1% of C in the beginning, but decreases the proportion of elements moved as f approaches f_c.

From our experience with testing all the methods in Pointillist, this method seems to be the most robust to most of data structures in terms of convergence. That is, despite the non-ideal distributions of the true positives (H_1) and negatives (H_0), it seems to converge and produce a reasonable set of selected data elements. Therefore, we made its Java version available as an open source and free at http://labs.systemsbiology.net/bolouri/software/Pointillist. Please notify us when it does not converge at pointillist@systemsbiology.org.

This method can be implemented by typing

```
>> opt = 1; %when Fisher's method is used
>> [out,stat]= nprampv2(x,alpha,objc,opt); %where x is the p-value
matrix; alpha = 0.05 (or αT from the previous thresholded or mixture
model methods); and objc = 0.99-alpha (β=0.01) or 0.95-alpha (β=0.05);
```

The output variable stat shows how the iteration went. The columns include the following information: 1st: the iteration number; 2nd-k+1th: weights for the *k* datasets being integrated; k+2th: the objective function (*f*); and the last column: the number of selected elements.

```
stat =

1     0.37278     0.62722     0.97667     1.3144      2841  2841  3222
2     0.37278     0.62722     0.97737     1.3931      2839  2839  3139
3     0.37278     0.62722     0.97806     1.4685      2837  2837  3080
4     0.37273     0.62727     0.97876     1.4801      2835  2835  3073
5     0.37273     0.62727     0.97945     1.5347      2833  2833  3028
6     0.37267     0.62733     0.98015     1.5597      2831  2831  3015
```

The output variable out stores the final weights and the index of selected elements.

```
out =

    w: [0.37267 0.62733]
    G: [1x3015 double]
    p: [6000x1 double]
```

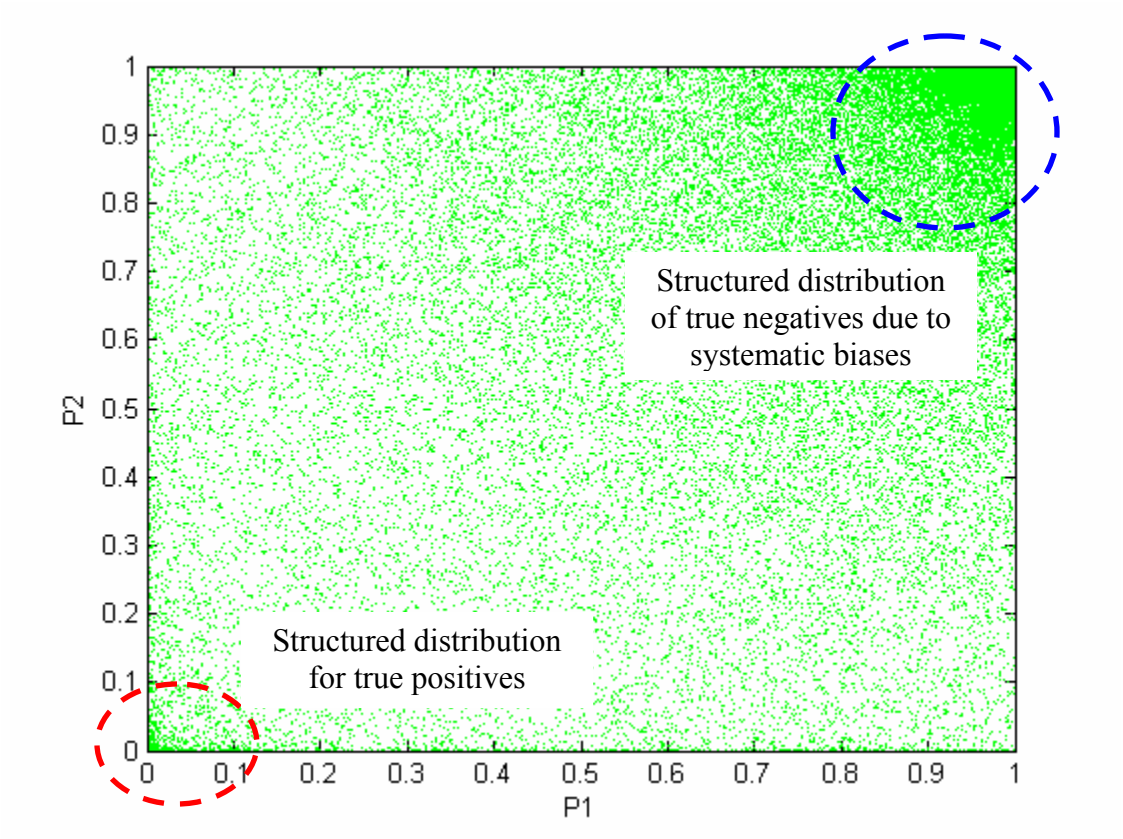
Overall p-values

The overall p-values are the recalculated p-values for the combined statistic values (see the combining equations in the beginning of this "Data Integration" section). These p-values can be subjected to the second thresholding using a user preferred α instead of α_T suggested from the thresholded and mixture distribution model because it depends on only the weights, not the α_T in all the parametric methods. Assessing the accuracy of the data integration results from Pointillist, studying the list of top system elements can be useful after sorting the elements by this overall p-value. In the current version of Pointillist, the overall p-values are being estimated by fitting the gamma distribution to weighted Fisher's statistic values and the normal distribution to MG and LS methods.

Practical Issues

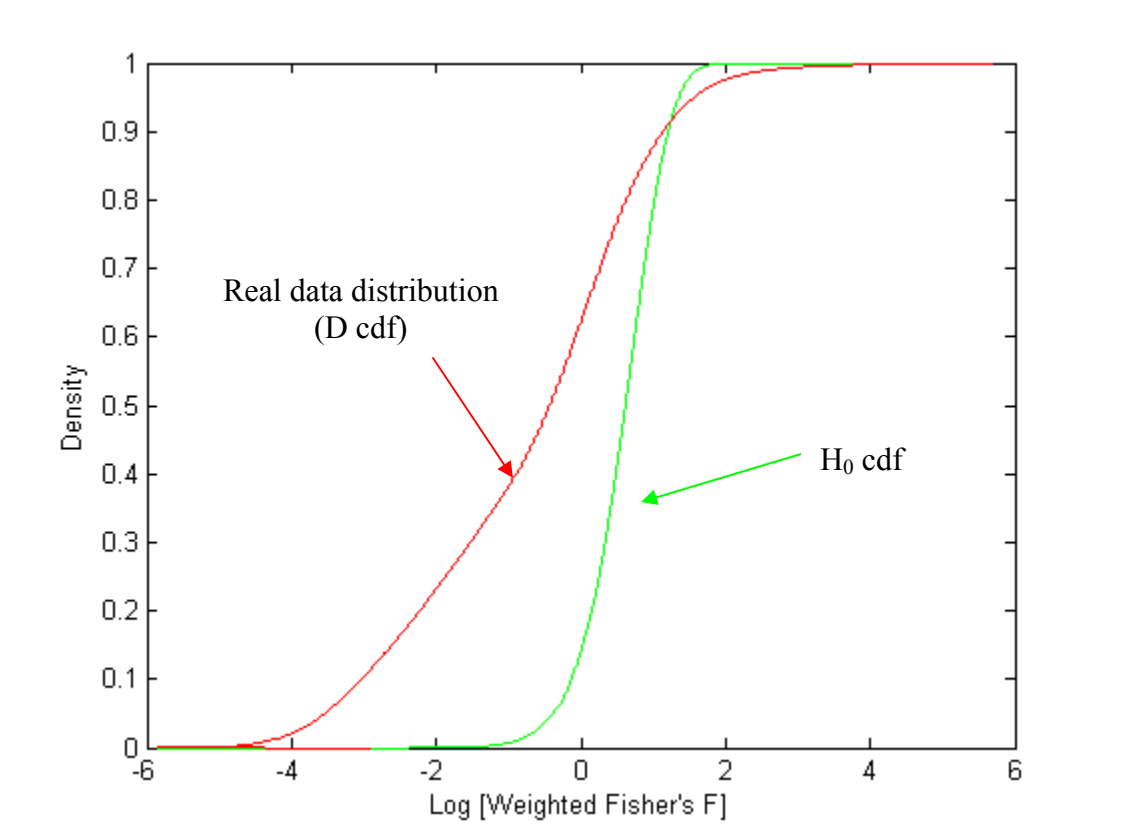
Although all the weighted integration methods work well with most of high-throughput datasets, there are in practice various situations in which these methods do not converge or fail to provide a reasonable set of selected data elements. Such situations occur mainly due to the inappropriate data structures such as:

1) The true negative data elements are distributed not uniformly, but in a structured manner. The following figure shows a structured distribution of the true negative data distribution near [1,1]. This structured distribution is due to a weak correlation among p-values of the true negative data elements between different datasets. This situation can occur when high-throughput technologies have similar systematic biases for the true negative elements so that their p-values can be weakly correlated. For example, the technologies detecting protein-protein interactions all can produce plausible p-values for sticky proteins. Also, microarrays can produce low intensities with similar patterns across experiments for low abundant transcripts (below the resolution of microarray).



These datasets produced the following non-ideal cdf (the red line), when compared to the theoretical H_0 distribution, in the sense that it is larger than the H_0 cdf in the most of range. For the ideal data, it should be lower than the H_0 cdf in the most of range

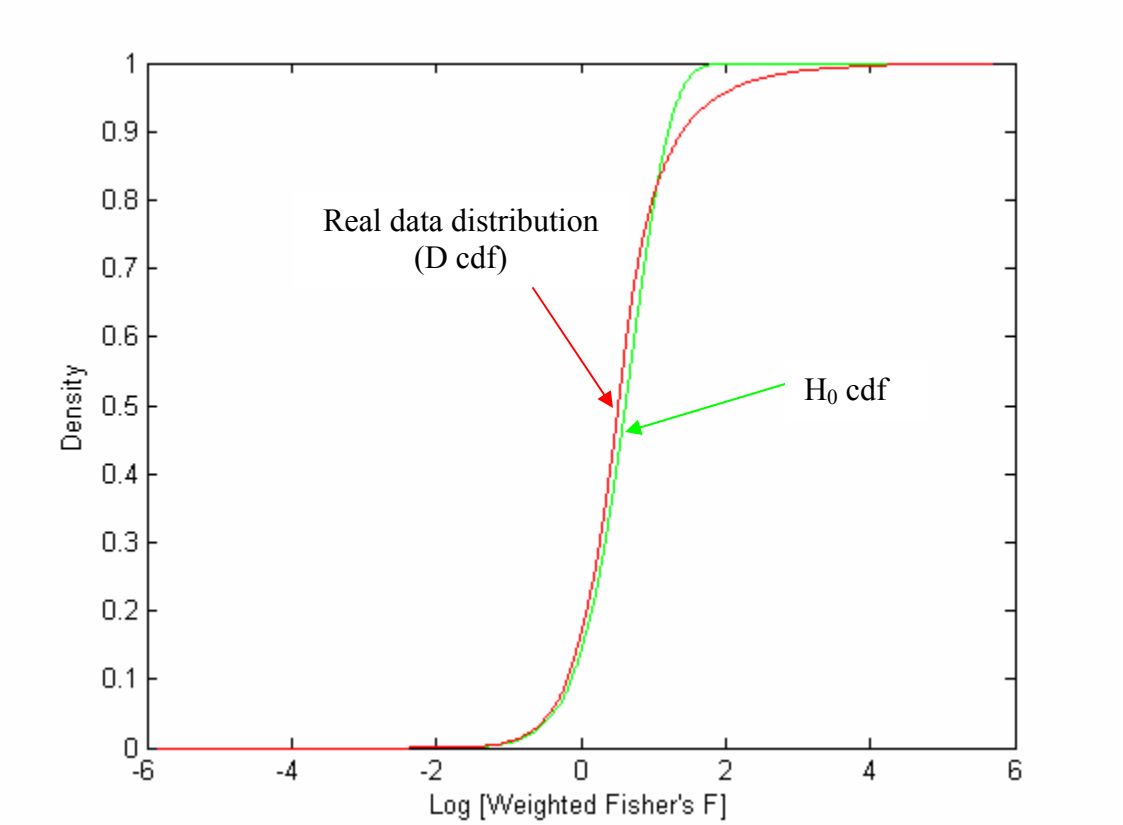
because it includes more true positive elements with the large weighted Fisher's F than the the H_0 population.



This kind of non-ideality can often be fixed by matching the apexes of the real data distribution and the H_0 pdf. This is based on the assumption that the fraction of true positive elements in the real high-throughput data is lower than that of true negative elements. In such cases, the apexes in the pdf in both the real data distribution and the H_0 pdf should be located at the same x-value (weighted Fisher's F). This idea was implemented by adding the following function that artificially matches the apexes of the H_0 and the real data distributions.

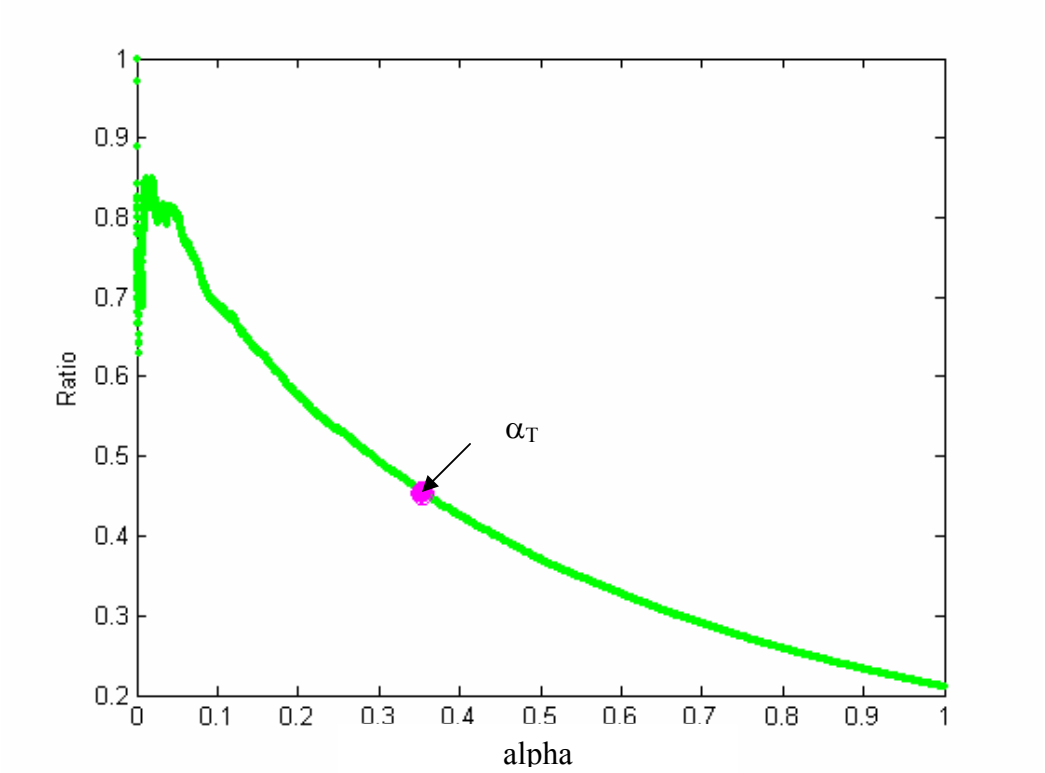
```
function y=cdfcorc(x,c,xc,cc,Tint)
[px,pxq]=qdistpeak(x);
[pc,pcq]=qdistpeak(c);
y=xc;
[xii,xjj]=min(abs(xc-pxq));
[cii,cjj]=min(abs(cc-pcq));
n=length(xc);
if pc>=px
    y(1:cjj)=cc(1:cjj)*pxq/pcq;
    njj=n-cjj;
    intjj=round(linspace(xjj+1,n,njj));
    y(cjj+1:end)=xc(intjj);
end
```

After the adjustment, the real data distribution looks as follows:

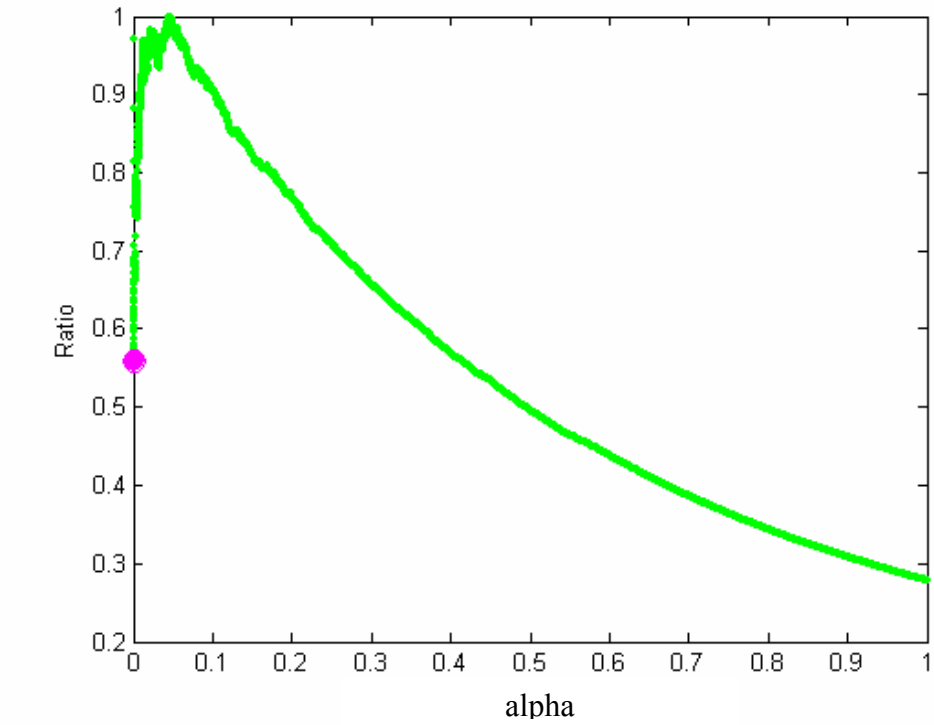


Although the real data distribution still stays slightly above the H0 cdf, it provides a reasonable set of selected genes by the "False Positive Reduction Methods." From our experience, many of these non-ideal situations from the structured distribution of true negative elements can be fixed by this additional adjustment of the real data distribution.

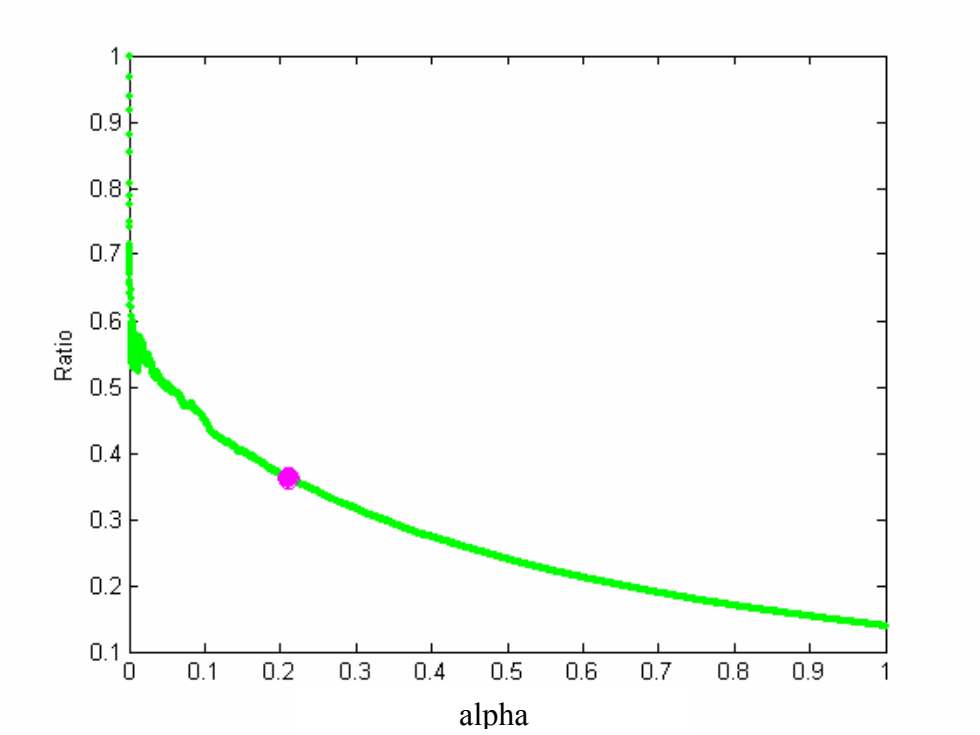
2) Undersampling of true positive elements can result in the failure of the "False Positive Reduction Methods" as well as "Mixture Distribution Model Methods." All these methods rely largely on whether the real distribution contains sufficient information on the distribution (H_1 pdf) of the true positive elements. They deal with the distribution of the true positive elements in a different manner. "False Positive Reduction Methods" requires the clear definition of the H_1 distribution from the apex to positive infinite (see the discussions in "False Positive Reduction Methods" section). On the other hand, "Mixture Distribution Model Methods" require a well-defined H_1 distribution over the entire range to ensure the accurate estimation of the H_1 distribution. Undersampling of the true positive elements decreases the fitting accuracy. The figure below shows one of cases where the true positive elements are undersampled, leading to the irregular pattern (fluctuations in the low alpha range; compare this to the figure above in the "False Positive Reduction Methods" section) in the ratio vs. alpha plot of "False Positive Reduction Methods." This irregular pattern eventually caused the method not to converge. Especially, after the second (and fourth) iteration, when the alpha value became small, it should start to converge into its optimal value. In this example, it fluctuates forever with having two attractors during the iterative optimization.



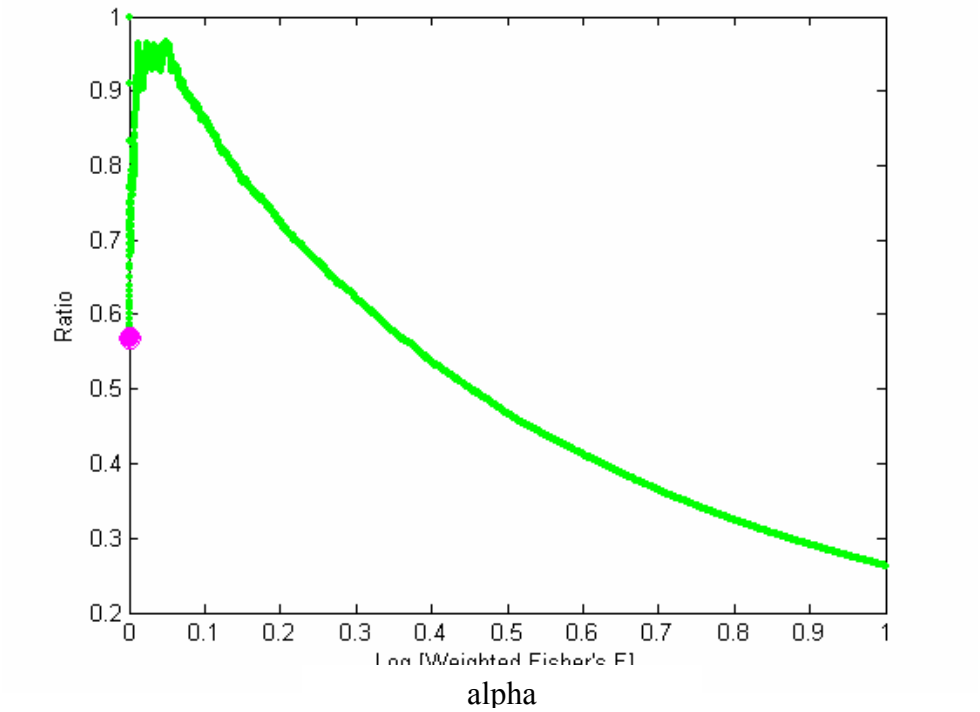
After the first iteration in "False Positive Reduction Method."



After the second iteration in "False Positive Reduction Method."



After the third iteration in “False Positive Reduction Method.”



After the fourth iteration in “False Positive Reduction Method.”

For these situations for which our integration methods do not converge, we suggest using one of the following methods instead: 1) the multi-objective optimization described in the

next section, 2) the weighted integration methods, and 3) non-parametric methods with alpha = 0.05.

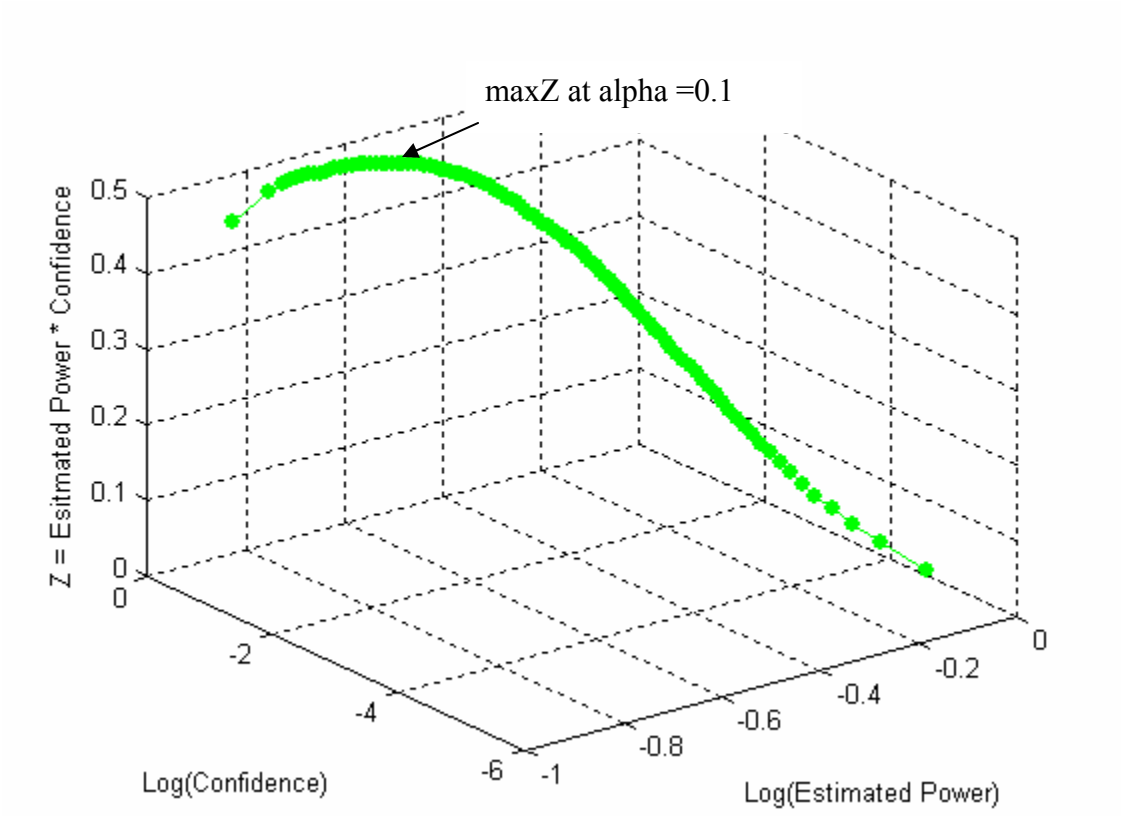
Extensions

Optimizations of multi objectives.

All the data integration methods are designed to find the parameters (α_T, and the weights) minimizing both the false positive and negative error rates at the same time (or maximizing statistical power and confidence). In all the methods, both α_T and the weights are optimized in iterative manners where several non-ideal aspects in the data structure can cause the convergence issues in certain cases. But, this problem can be also seen as a normal multi-objective optimization problem where the product of estimated power and the confidence is being maximized. A Pareto space for given datasets can be constructed by typing in the MATLAB command window:

```
>> y=paretospace(x);% %where x is the p-value matrix, y is a MATLAB
structure variable storing all the outputs.
```

Then, the objective function (Z = the product of estimated power and confidence) will be plotted as shown below:

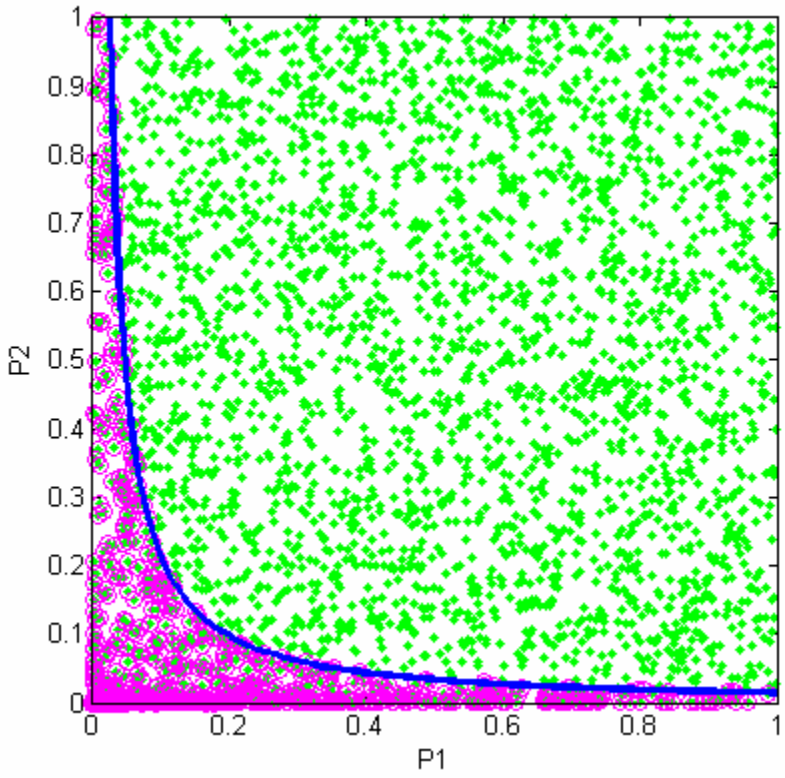


Typing y in the commend window will provide 1) Estimated power, confidence, and their product for each alpha value (mobj), 2) the optimal alpha with the maximum Z (alpha), 3) the weights for the optimal alpha (w); and 4) the index of the selected elements (ts):

```
>> y
```

```
y =

       ts: [3191x1 double]
    alpha: 0.1001
        w: [0.5313 0.4687]
     mobj: [100x3 double]
```

The resulting decision boundary would be as follows:



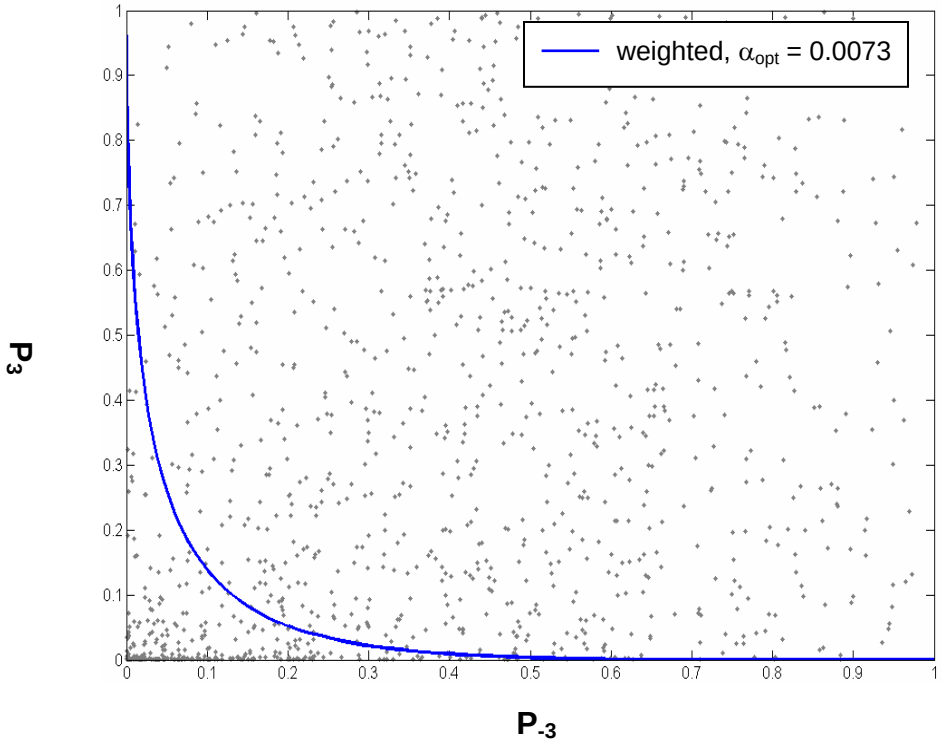
We recommend using this method in any case to see how these conflicting objectives (power and confidence) vary with the alpha values on the Pareto space. Also, this method gives normally a reasonably accurate alpha and the weights to maximize both the power and confidence at the same time.

Handling Missing Values

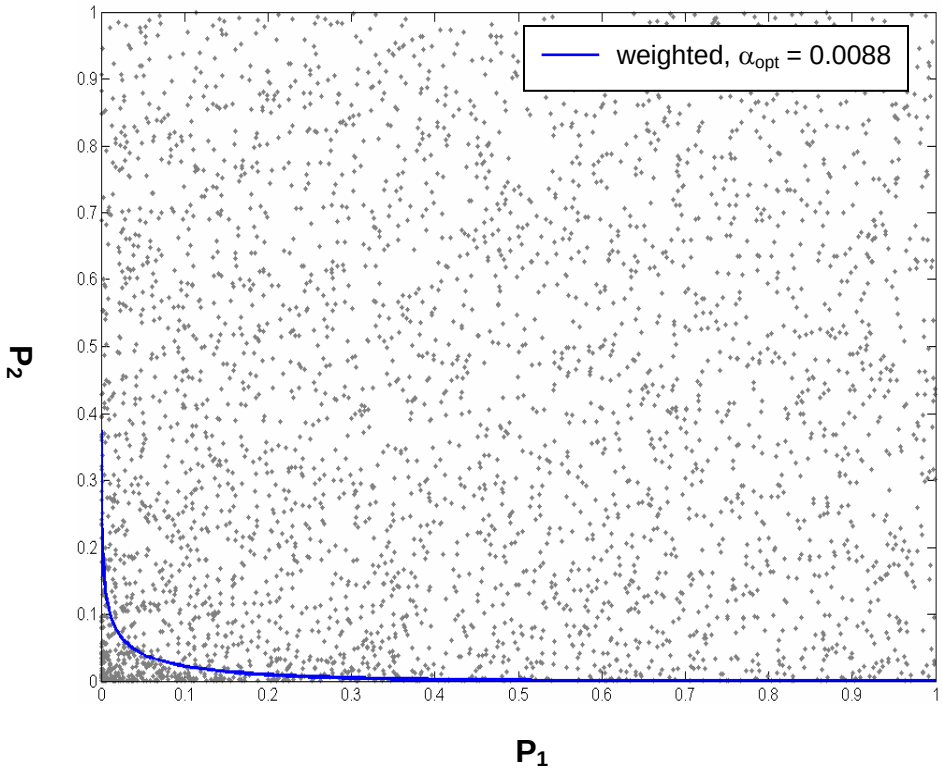
Missing data points can arise from systematic biases in high-throughput technologies, e.g. favoring high abundance species. Missing values also occur when we integrate global datasets with curated database information. This systematic bias also affects statistical power by decreasing the sample size, unlike random noise lessening statistical power by decreasing the measurement accuracy. For methods that combine independent uniformly distributed p-values into a uniformly distributed p-value, missing values may be handled by using the available p-values to compute each combined p-value. The simplest way of handling these missing values would be to assign a fixed p-value in place of the missing data (e.g. p = 0.5 to indicate an equal chance of being a true positive or a true negative, or p = 1.0 to discourage elements with missing values from being selected). If one or more datasets include many missing values, the above approach can distort the calculated weights. To avoid such a scenario, we exclude data elements with missing values from the calculation of weights, thereby avoiding the above distortion problem. However, we include them in the data selection process, so that data elements with missing values can still be selected. This approach is called substitution analysis below.

In addition to this simple approach, using Pointillist, we can also handle these missing values by grouping the elements according to the status of missing values and integrating each group separately using the weighted integration methods above (called local analysis). However, for this local analysis, it is required that each group should have a sufficient number of elements so that the correct weights can be found by maximizing statistical power.

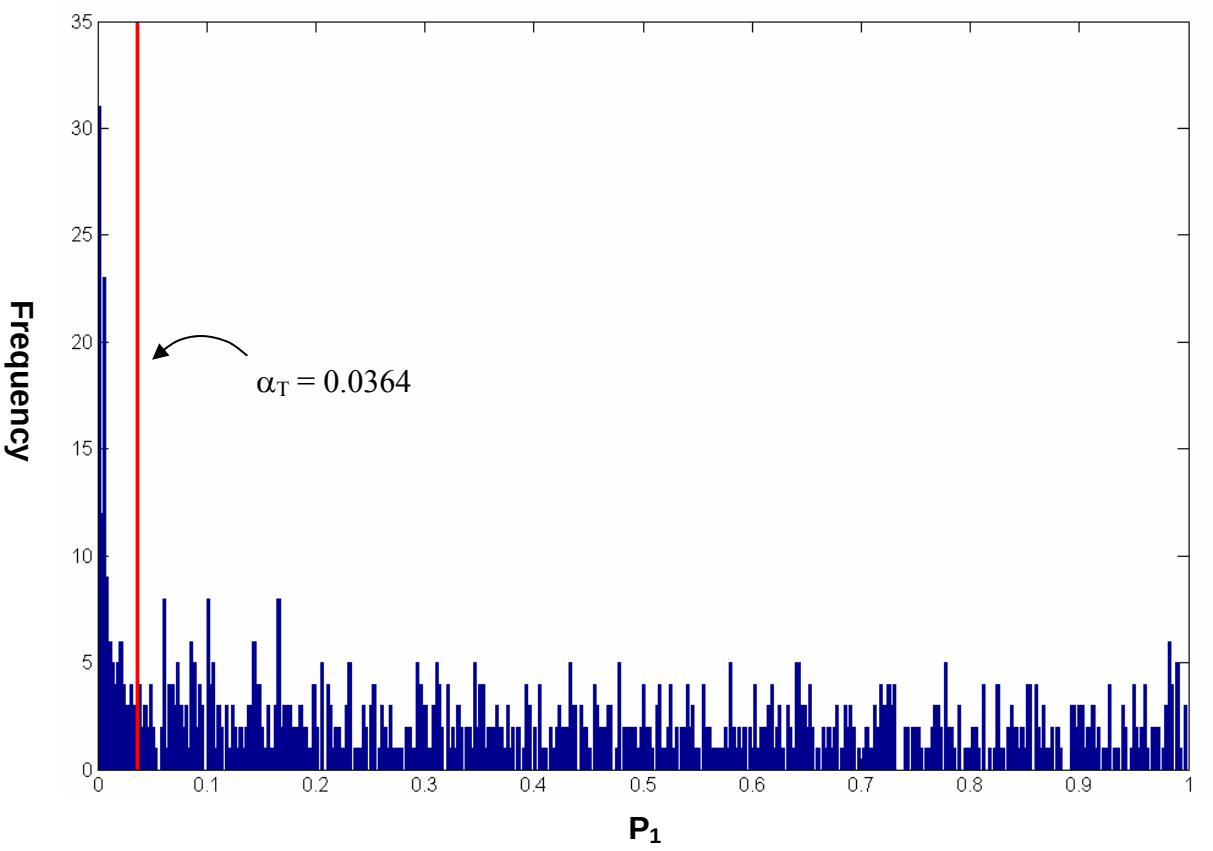
To evaluate its performance in the presence of missing values, three simulated data sets were generated as described in the paper, which mimic microarray, MPSS, and iCAT data with zero, 20%, and 75% percentages of missing data points. The noise levels reflecting the measurement accuracy were used as 0.5, 0.4, and 0.3 for the three data sets, respectively. For local analysis, we first integrated a subset of data in which each element has all the data points available in three data sets. The following figure shows the 222 elements selected by the decision boundary for the weighted Stouffer's Z_w with α_T = 0.0073 using "False Positive Reduction Method."



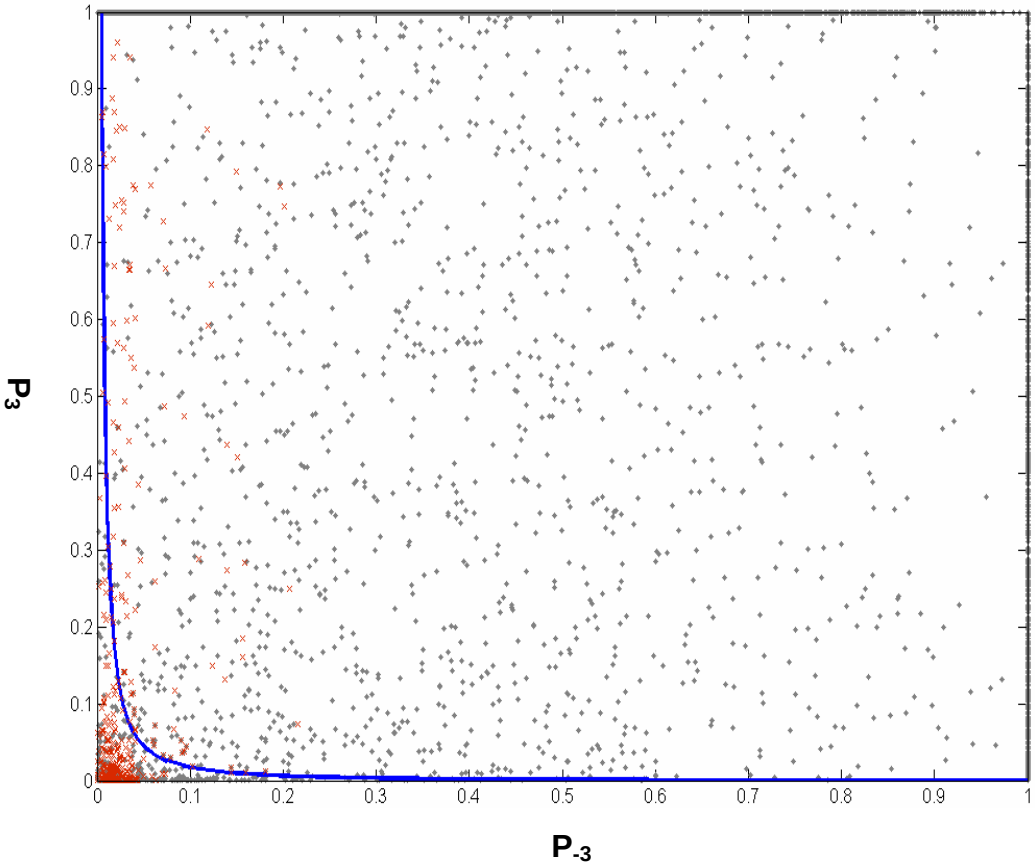
Second, we integrated another subset of data in which each element has the data points available in the first two data (microarray and MPSS data). The following figure shows the 366 significant elements selected by the decision boundary for the weighted Stouffer's Z_w with $\alpha_T = 0.0088$ using "False Positive Reduction Method."



For the remaining elements with only data points available in the first data set (microarray), the following histogram shows the 120 significant elements selected the decision boundary for the weighted Stouffer's Z_w with $\alpha_T = 0.036$ using "False Positive Reduction Method." As a result, total 708 elements were selected.



Using substitution analysis, on the other hand, we replaced the p-values of the missing values with p=1 and then integrated the three data sets. The following figure shows the decision boundary for the weighted Fisher's F_w with $\alpha_T = 0.0062$. In this case, we used Fisher's F_w to better handle the complementary nature in the datasets, given p = 1 for the missing values. The 414 significant elements were selected using this boundary. This indicates that substitution analysis seems to sacrifice the maximum statistical power that can be achieved by local analysis because we selected a less number of significant elements (708 versus 414). The numbers of overlapping elements of these 414 elements with the three sets of elements selected by local analysis were 205, 93, and 28 (total 326), respectively. The other 88 elements selected by substitution analysis stay very near the three boundaries in the above figures. The 382 elements selected by local analysis but not by substitution analysis were overlaid in the figure below (represented by the red "x"s). It turned out that among the 382 elements, the 111 selected elements staying outside the boundary are false positives. This may be considered that local analysis increases the sensitivity (or local power) but slightly decreases the robustness, compared to substitution analysis. When there are a sufficient number of data points to optimize the weights by maximizing statistical power, local analysis is recommended.


P3
P-3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1