The Mathematica® Journal

Detecting Differential Gene Expression Using Affymetrix Microarrays

Todd D. Allen

This article describes the development of a novel program to process Affymetrix microarray files, which are used in the biological sciences to establish differences in gene expression between two conditions (e.g., diseased tissue versus healthy tissue).

Introduction and Background

Affymetrix gene expression microarrays ("chips") are a commercial implementation of a powerful concept originally introduced to the world by Shena and colleagues in 1995 [1]. When successfully implemented, gene expression microarrays let a biologist measure the expression of thousands of genes simultaneously in a biological sample, such as heart tissue, and further, to compare that measure of expression between two biological states, such as diseased heart tissue and healthy heart tissue. In many ways, the introduction of microarray technology has created a revolution in biology, transforming the field into a "big data" science like its sister disciplines of physics and chemistry.

Gene expression microarrays (Figure 1A) are manufactured by attaching strands of deoxyribonucleic acid (DNA), corresponding to different genes of an organism, across the surface of a glass slide. The basic process of identifying genes that are expressed begins with the extraction of messenger RNA (mRNA) from a source, for example, healthy heart cells (Figure 1B). The molecule mRNA is made by cells when a gene is expressed, meaning its physical presence—assuming it can be reliably detected—is an indicator of gene expression. By fluorescently labeling the mRNA and hybridizing it to the surface of the chip, it is possible to quantify the intensity of multiple genes' expression from that biological source by using a scanner able to measure a fluorescent signal. When this process is repeated on a different biological source, such as diseased heart tissue, a separate gene expression profile is created for the diseased tissue, which can then be computationally compared to the expression profile from the healthy tissue. In this manner, genes that are more highly expressed or more highly repressed in diseased tissue can be identified by comparing their expression profile to the expression profile of the same genes in the healthy tissue. This has obvious implications for determining which

genes may be playing a role in disease development or any other biological process of interest, from cancer metastasis and drug resistance in medicine to fruit ripening and drought resistance in agriculture.

Figure 1A

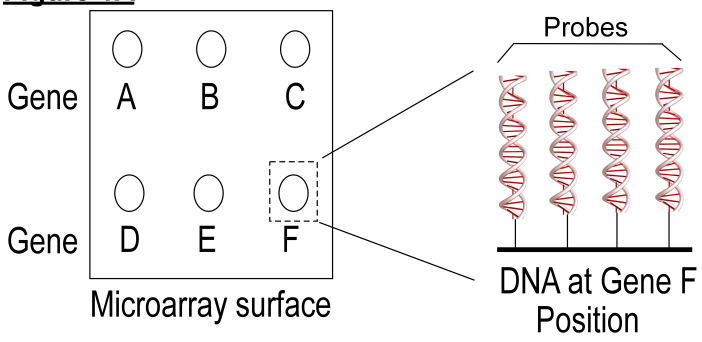


Figure 1B

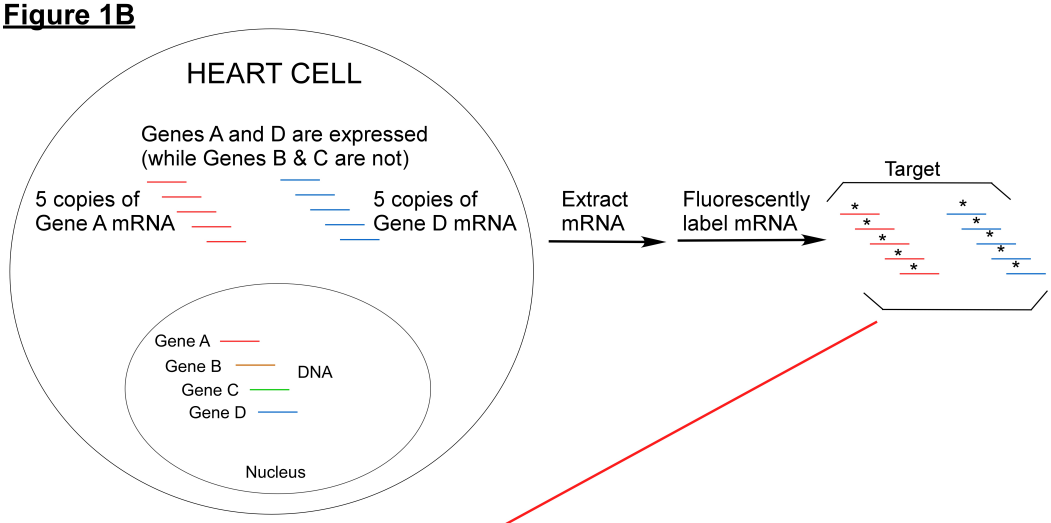
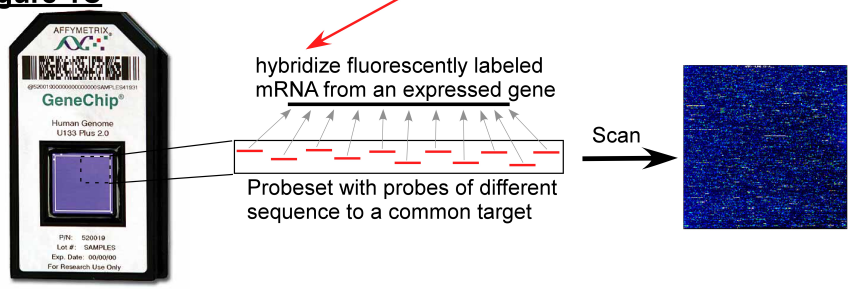


Figure 1C



▲ **Figure 1.** (A) Basic microarray design and layout. (B) Extraction of mRNA—expressed from the nucleus of a healthy heart cell—and its subsequent fluorescent labeling. (C) Hybridization of the fluorescently labeled mRNA ("target") to the surface of an Affymetrix microarray chip, and its subsequent scanning to quantify the fluorescent signal, assumed to be proportional to gene expression.

The Affymetrix implementation of gene expression microarrays utilizes probesets, synthesized in place, on the surface of the microarray chip (Figure 1C). Probesets are groups of small DNA fragments that are complementary to different regions of the same mRNA molecule made whenever a gene is expressed. By combining the fluorescent signal of the probeset group, a single measure of gene expression is arrived at computationally, which is the primary focus of the algorithm presented here. Probesets are composed of groups of perfect match ("PM") and mismatch ("MM") probes. In the context of microarray analysis, the term "probe" refers to the strands of DNA physically tethered to the microarray chip and the term "target" refers to the fluorescently labeled sample of mRNA obtained from a biological source, which will be hybridized to the probes to measure gene expression. Perfect match probes are single-strand sequences of DNA, usually 25 nucleotides in length, which have perfect complementarity to the mRNA sequence to which they are designed to hybridize. Mismatch probes are identical to PM probes, with the exception of one nucleotide in the center of the molecule (typically at position 13) that is not a proper match to the mRNA with which it is designed to hybridize. The purpose of MM probes is to measure background fluorescent signal off the surface of the chip, which is one source of technical noise.

The analysis of microarray data involves numerous steps, some of which are not universally performed, but whose combination is collectively referred to as an "analysis pathway." The steps in an analysis pathway typically involve:

1. background correction: performed to remove fluorescent signal not due to biology
2. probe normalization: used to place the individual datasets of an experiment on the same scale, so the datasets can be compared accurately
3. perfect match (PM) probe correction: used to correct biases in the PM probe signal, often due to differences in DNA sequence between the probes
4. summarization: performed to obtain a single measure of gene expression from the multiple measurements obtained by each probeset; this process often attempts to correct "probe" and "chip" technical noise
5. probeset normalization: sometimes performed to make the probesets between datasets more directly comparable
6. differentially expressed genes (DEG) test: uses a statistical test to identify "true" differentially expressed genes

Despite all of its positive aspects, microarray technology requires considerable knowledge to use effectively, as the raw signal that is generated from the technology is almost always noisy. The scientific literature is rife with algorithms designed to remove various sources of known error, and it is immediately clear that there is no "perfect" algorithm that is universally useful in all experimental situations. Even so, a seminal article by Zhu and colleagues [2] evaluated different combinations of commonly used algorithms—representing over 40,000 different analysis pathways—using a precisely controlled "spike-in" dataset, which allowed the researchers to identify the most important steps common to a good analysis pathway.

The algorithm presented here represents a merging of several of the "best step" analysis pathway practices identified in [2]. Specifically, the algorithm presented here uses the following analysis pathway:

1. background correction: none
2. probe normalization: performed using quantile normalization [3]
3. perfect match (PM) probe correction: none
4. summarization: performed using median polish [4]
5. probeset normalization: none
6. differentially expressed genes (DEG) test: while probability data is provided to aid interpretation, the identification of differentially expressed genes is not performed with statistical methods, but instead relies on graphical interpretation of the processed data

The algorithm presented here does not perform background correction, perfect match probe correction, or probeset normalization because the evidence presented in [2] suggests that these steps are at best unnecessary and sometimes even detrimental. Readers interested in a deeper discussion of microarray technology and data analysis are referred to the excellent reviews in [5, 6].

The Affymetrix Differential Gene Expression Detection (AffyDGED) Algorithm

The AffyDGED algorithm is template-driven, meaning that the algorithm expects several pieces of user-defined information to be provided in a notebook cell used as a template for entering the information.

```
cellocation =
  "C:\\Users\\Wookie\\Desktop\\Mathematica
    Projects\\Mathematica Journal Projects\\Data\\Saliva
    Biomarkers for Pancreatic Cancer Project\\Saliva
    Biomarkers For Pancreatic Cancer - raw CEL data\\";

affycdflocation =
  "C:\\Users\\Wookie\\Desktop\\Mathematica
    Projects\\Mathematica Journal
    Projects\\Data\\AffyChip Description
    Files\\HG-U133_Plus_2\\LibFiles\\";

savelocationroot = "C:\\Users\\Wookie\\Desktop\\";

qnormversion = all;

studyname = salivadataset;
```

```
experimentchips = {GSM356796, GSM356797, GSM356798,
   GSM356799, GSM356800, GSM356801, GSM356802, GSM356803,
   GSM356804, GSM356805, GSM356806, GSM356807};

controlchips = {GSM356808, GSM356809, GSM356810,
   GSM356811, GSM356812, GSM356813, GSM356814, GSM356815,
   GSM356816, GSM356817, GSM356818, GSM356819};
```

The example above contains several variables that must be completed by the user to let AffyDGED do its job properly.

The variables requiring user input are:

1. `cellocation`: This variable holds the directory location for finding the CEL data of the microarray hybridizations. CEL files contain the raw fluorescent data from a microarray experiment using Affymetrix technology.
2. `affycdflocation`: This variable holds the directory location for finding the Affymetrix CDF library file. CDF stands for "chip description file" and refers to an Affymetrix file that describes, among other things, the location of the probes on the specific type of Affymetrix chip being used.

   Caution: Users should take care not to confuse the Computable Document Format (.cdf) of Wolfram Research with the chip description files (.cdf) from Affymetrix. Affymetrix CDF files cannot be opened directly by the Wolfram *CDF Player*.
3. `savelocationroot`: This variable holds the location where the user would like the final results of the analysis to be saved.
4. `qnormversion`: This variable lets the user select between two options. The option "all" can be entered to perform quantile normalization using all the chips involved in an experiment at once, or alternatively, the option "condition" can be entered, which directs the algorithm to perform quantile normalization by condition, that is, perform quantile normalization twice, once using the data in the experimental condition (such as diseased heart tissue) and then a second time using the control condition data (such as healthy heart tissue). When in doubt, it is recommended that "all" be used.
5. `studyname`: This variable lets the user name the experiment/study being processed by the AffyDGED algorithm. The output of AffyDGED is saved using this name to the location provided in "savelocationroot" above.
6. `experimentchips`: This variable contains a list of the experimental condition datasets being studied.
7. `controlchips`: This variable contains a list of the control condition datasets being studied.

To illustrate the features of the AffyDGED algorithm, we use data from a modestly sized microarray experiment involving the detection of differentially expressed genes between the saliva of pancreatic patients and healthy individuals [7]. All microarray data used in this study and presented here is publicly available at NCBI's Gene Expression Omnibus portal (www.ncbi.nlm.nih.gov/geo), using the access number GSE14245.

The first tasks completed by AffyDGED include the loading of raw data, the determination of the physical dimensions of the chip data, and the conversion of Affymetrix probe position coordinates to equivalent *Mathematica* indices.

```
chipdimensions[chip_] := Module[{n, k},
  n = Length[chip[[All, 1]]];
  k = Length[chip[[1]]];
  {n, k}]

affyindextoMMAindices[affyindex_, chipsize_] :=
 Module[{ax, ay, mmai1, mmai2},
  ay = Floor[((affyindex - 1) / chipsize[[2]])];
  ax = (affyindex) - chipsize[[2]] * ay;
  mmai1 = (ay + 1);
  mmai2 = (ax + 1);
  {mmai1, mmai2}]


starttime = AbsoluteTime[];
SetDirectory[cellocation];
celfilenames = FileNames[];
celvarnames =
 Table[StringSplit[celfilenames[[i]], {"."}][[1]],
  {i, 1, Length[celfilenames]}];

Table[microarray[celvarnames[[i]]] =
  Import[celfilenames[[i]]], {i, 1, Length[celfilenames]}];
chipsize = chipdimensions[microarray[celvarnames[[1]]]];

SetDirectory[affycdflocation];
cdffilenames = FileNames[];
cdffile =
  Import[
   Flatten[StringCases[cdffilenames, ___ ~~ ".cdf" ~~ ___]][[
    1]]];
ginfile =
  Import[
   Flatten[StringCases[cdffilenames, ___ ~~ ".gin" ~~ ___]][[
    1]]];

probesetids = cdffile[[All, 1, 2]];
dispatchrules =
  Dispatch[
   Thread[probesetids → Range[Length[probesetids]]]];
ginreorderinfo =
  Ordering[(ginfile[[All, 4]] /. dispatchrules)];
reorderedgin = ginfile[[ginreorderinfo]];
```

```
experimentchips = Map[ToString, experimentchips];
controlchips = Map[ToString, controlchips];

pmindexes =
 Table[Select[cdffile[[i, 3, 2]], #[[5]] ≠ #[[6]] &][[
   All, 4]], {i, 1, Length[cdffile]}];
mmapmindices =
 Map[Transpose,
  Thread[affyindextoMMAindices[pmindexes, chipsize]]];
pmtemp =
 Table[Extract[microarray[celvarnames[[i]]],
   mmapmindices[[j]]], {i, 1, Length[celvarnames]},
  {j, 1, Length[mmapmindices]}];
Table[pmsignalraw[celvarnames[[i]]] = pmtemp[[i]],
  {i, 1, Length[celvarnames]}];
Clear[pmtemp];
```

The `chipdimensions` and `affyindextoMMAindices` modules are designed to establish the number of rows and columns of probe data on the microarray chips, as well as to convert the probe position coordinates as assigned by Affymetrix to equivalent *Mathematica* indices.

For example, the chips used here (Human Genome U133 Plus 2.0) happen to be square, with 1,164 rows of information and 1,164 columns of information.

```
chipsize
```

```
{1164, 1164}
```

Affymetrix uses a single-number index (present on the Affymetrix CDF file) created from (x, y) coordinates of the probes present on their microarray chips. The single-number index is derived from a formula used by Affymetrix that assumes an (x, y) index of $(0, 0)$ refers to the uppermost-leftmost position of the chip. To successfully load the probeset data into usable groups in *Mathematica*, it is necessary to shift the $(0, 0)$ coordinates used by Affymetrix into `[[1, 1]]` indexing used by *Mathematica*.

Here is an example of the data contained within the Affymetrix CDF file.

```
cdffile[[3576]]
```

```
{ProbeSetName → 203987_at,
 CellLabel → {IndexPosition, X, Y, IndexPosition,
   SubstitutionBase, TargetBase},
 CellData → {{0, 993, 202, 236 121, A, A},
   {0, 993, 201, 234 957, T, A}, {1, 897, 537, 625 965, C, G},
   {1, 897, 538, 627 129, G, G}, {2, 954, 782, 911 202, A, A},
   {2, 954, 781, 910 038, T, A}, {3, 221, 821, 955 865, A, T},
   {3, 221, 822, 957 029, T, T}, {4, 224, 654, 761 480, A, A},
   {4, 224, 653, 760 316, T, A}, {5, 937, 927, 1 079 965, C, G},
   {5, 937, 928, 1 081 129, G, G}, {6, 118, 932, 1 084 966, C, C},
   {6, 118, 931, 1 083 802, G, C}, {7, 76, 894, 1 040 692, A, A},
   {7, 76, 893, 1 039 528, T, A}, {8, 561, 547, 637 269, C, G},
   {8, 561, 548, 638 433, G, G}, {9, 548, 715, 832 808, C, G},
   {9, 548, 716, 833 972, G, G}, {10, 144, 152, 177 072, A, A},
   {10, 144, 151, 175 908, T, A}}, Direction → anti-sense}
```

The AffyDGED algorithm parses out the single-number indices for the perfect match probes of each probeset and converts that positional information into usable indices for *Mathematica*. The mismatch probes are purposely ignored in the AffyDGED algorithm because they often produce signals higher than the perfect match probes, which is an indication that the mismatch probes are not performing as originally intended by Affymetrix engineers.

```
pmindexes[[3576]]
```

```
{234 957, 625 965, 910 038, 955 865, 760 316,
 1 079 965, 1 083 802, 1 039 528, 637 269, 832 808, 175 908}
```

There are 11 individual numbers, referring to the (x, y) positions (in Affymetrix coordinates) of 11 perfect match probes of a single probeset used to measure the expression of a specific gene.

Here is the same positional information, now expressed in *Mathematica*'s indexing system.

```
mmapmindices[[3576]]
```

```
{{202, 994}, {538, 898}, {782, 955},
 {822, 222}, {654, 225}, {928, 938}, {932, 119},
 {894, 77}, {548, 562}, {716, 549}, {152, 145}}
```

There are now 11 groups of (x, y) *Mathematica* indices referring to the same data and accessible by conventional *Mathematica* indexing.

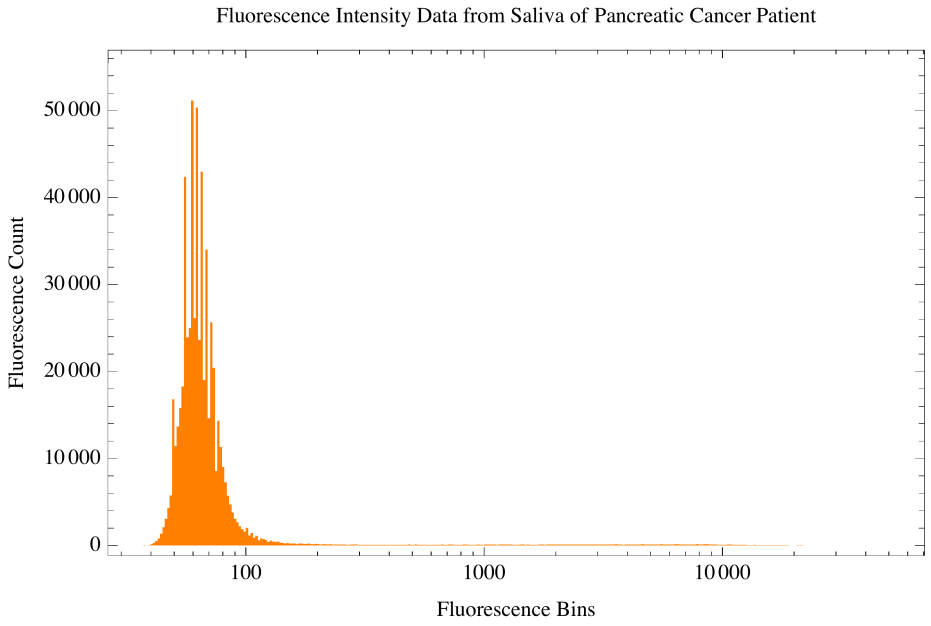
We can now take a look at the raw data from a single chip used in the pancreatic cancer study that has been loaded. Notice the extreme range of values typically seen in microarray experiments.

```
{Min[Flatten[pmsignalraw["GSM356796"]]],
 Max[Flatten[pmsignalraw["GSM356796"]]]}
```

```
{31., 60 050.}
```

For this reason, we look at a histogram of the data using a logarithmic scale (Figure 2).

```
Histogram[Flatten[pmsignalraw["GSM356796"]], "Log",
 ChartStyle → Orange, Frame → True,
 FrameLabel → {{"Fluorescence Count", ""},
   {"Fluorescence Bins",
    "Fluorescence Intensity Data from Saliva of
      Pancreatic Cancer Patient"}}]
```



▲ **Figure 2.** A histogram of the raw fluorescence intensity data (log scale) contained in the microarray chip GSM356796, used to measure the expression of genes in the saliva of a pancreatic cancer patient.

The majority of the data has an approximately Gaussian appearance (on a log scale), while still harboring extreme values on the rightward tail (look carefully along the x axis). This shape is very characteristic of microarray data.

The next steps in processing include transforming the raw data to a $\log_2$ scale, determining the probeset size specific for the chip in use, and performing quantile normalization.

```
quantilenorm[alldata_] :=
 Module[{datarows, datasort, datasortrows1, datasortrows2,
   datasortmean, datasortprime, datanorm1, datanorm2,
   finaldata},

  datarows = MapThread[List,
    {alldata, Range[Length[alldata]]}];
  datarows =
   Table[Thread[datarows[[i, 1]] → datarows[[i, 2]]],
    {i, 1, Length[datarows]}];
  datasort = Table[Sort[datarows[[All, i]]],
    {i, 1, Length[datarows[[1]]]}];

  datasortrows1 = Transpose[datasort];
  datasortrows2 = Table[datasortrows1[[i, All, 1]],
    {i, 1, Length[datasortrows1]}];
  datasortmean = Map[Mean, datasortrows2] // N;

  datasortprime =
   Table[ReplacePart[datasortrows1[[i]],
     Thread[List[Range[Length[datasortrows1[[i]]]], 1]] →
      datasortmean[[i]]], {i, 1 Length[datasortrows1]}];
  datanorm1 = Transpose[datasortprime];

  quicksort[sortlist_, element_] :=
   sortlist[[Ordering[sortlist[[All, element]]]]];
  datanorm2 = Table[quicksort[datanorm1[[i]], 2],
    {i, 1, Length[datanorm1]}];
  finaldata =
   Flatten[Table[List[datanorm2[[i, All, 1]]],
     {i, 1, Length[datanorm2]}], 1]]
```

```
Table[pmsignallog[celvarnames[[i]]] =
  Log2[pmsignalraw[celvarnames[[i]]]],
 {i, 1, Length[celvarnames]}];
commonpssize =
 Sort[
   Tally[
    Map[Length, RandomChoice[pmsignallog[celvarnames[[1]]],
      1000]]], (#1[[2]] < #2[[2]]) &][[-1, 1]];
oddballs = Position[pmsignallog[celvarnames[[1]]],
  (x_ /; Length[x] ≠ commonpssize), {1}, Heads → False];
minicdffile = Delete[cdffile, oddballs];
miniginfile = Delete[reorderedgin, oddballs];

If[qnormversion === all,

  (allchipsrectangular =
    Transpose[
     Table[Flatten[Delete[pmsignallog[celvarnames[[i]]],
        oddballs]], {i, 1, Length[celvarnames]}]];
   quantnormtemp = quantilenorm[allchipsrectangular];
   Table[quantnorm[celvarnames[[i]]] =
     Partition[quantnormtemp[[i]], commonpssize],
    {i, 1, Length[celvarnames]}];
   Clear[quantnormtemp];) ,

  (allexprectangular =
    Transpose[
     Table[
      Flatten[Delete[pmsignallog[experimentchips[[i]]],
        oddballs]], {i, 1, Length[experimentchips]}]];
   allcontrectangular =
    Transpose[
     Table[Flatten[Delete[pmsignallog[controlchips[[i]]],
        oddballs]], {i, 1, Length[controlchips]}]];

   expquantnormtemp = quantilenorm[allexprectangular];
   contquantnormtemp = quantilenorm[allcontrectangular];

   Table[quantnorm[experimentchips[[i]]] =
     Partition[expquantnormtemp[[i]], commonpssize],
    {i, 1, Length[experimentchips]}];
   Table[quantnorm[controlchips[[i]]] =
     Partition[contquantnormtemp[[i]], commonpssize],
    {i, 1, Length[controlchips]}];
   Clear[expquantnormtemp, contquantnormtemp];)];
```

The convention of transforming raw microarray data by $\log_2$ is almost universally used in the microarray community, because it performs two useful functions. First, it makes the distribution of raw data more Gaussian (although certainly not perfectly so), and it aids interpretation of gene expression ratios for the end user, because it is easier to appreciate that a ratio of $+2$ and -2 on a log scale indicates the same degree of "up" and "down" regulation for a gene, as opposed to $+4$ and $+0.25$ on an absolute scale.

In the example shown here, this is the probeset size (the number of probes making up each probeset).

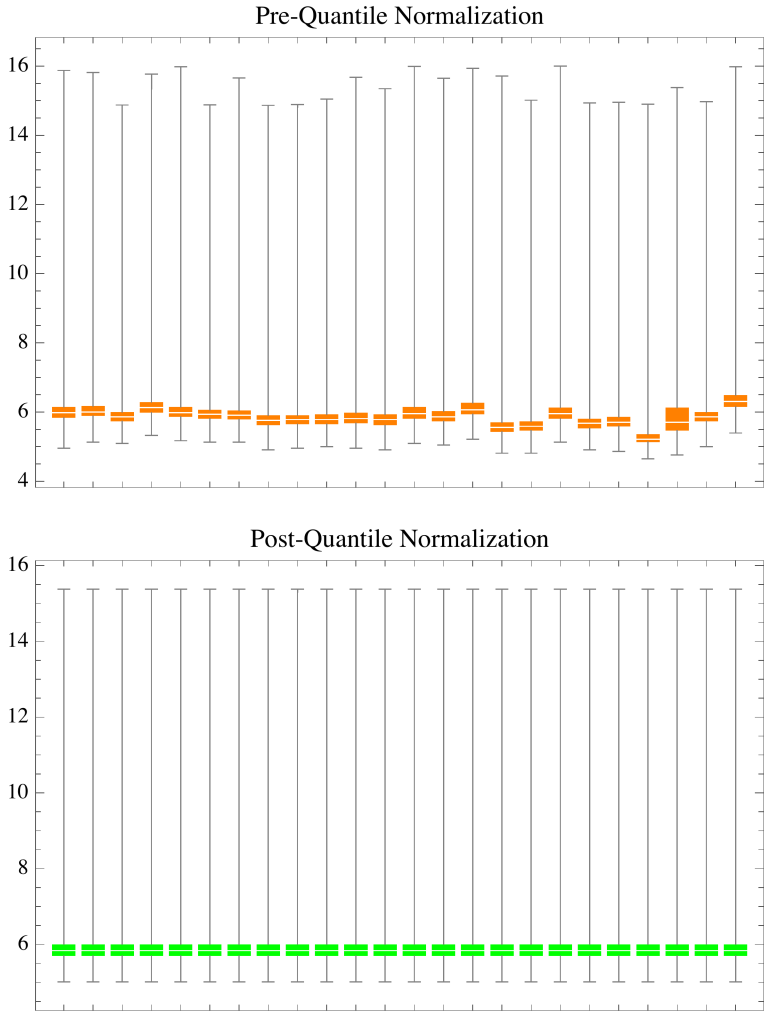
```
commonpssize
```

```
11
```

Quantile normalization is performed to place each dataset of the experiment on a common scale so the datasets can be appropriately compared to each other.

Using `BoxWhiskerChart`, this shows the difference between the pre-quantile normalized data and the post-quantile normalized data (Figure 3).

```
preQnorm = Table[Flatten[pmsignallog[celvarnames[[i]]]],
   {i, 1, Length[celvarnames]}];
postQnorm = Table[Flatten[quantnorm[celvarnames[[i]]]],
   {i, 1, Length[celvarnames]}];
temp1 = BoxWhiskerChart[preQnorm, ChartStyle → Orange,
   PlotLabel → "Pre-Quantile Normalization",
   PerformanceGoal → "Speed"];
temp2 = BoxWhiskerChart[postQnorm, ChartStyle → Green,
   PlotLabel → "Post-Quantile Normalization",
   PerformanceGoal → "Speed"];
GraphicsColumn[{temp1, temp2}]
```



▲ **Figure 3.** A box-and-whisker comparison of all 24 microarray chips used to compare gene expression between the saliva of pancreatic cancer and healthy patients, before and after quantile normalization.

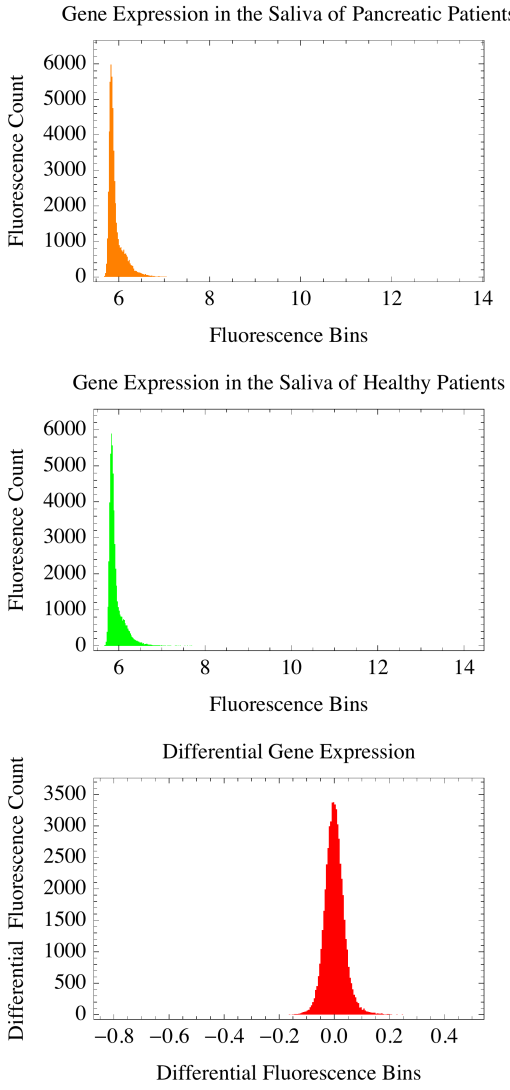
Following quantile normalization, the probesets are summarized (i.e., a single measure of gene expression is generated) by first performing median polish and then taking the mean of the polished values for all probes within a probeset. After the probesets are summarized, the differential expression of each gene is obtained by subtracting the expression of a gene in the control condition from the expression of the gene in the experimental condition. For example, if the expression of a gene in the saliva of pancreatic cancer patients (the "experimental" condition) is 2.1 and the expression of the same gene in the saliva of healthy patients (the "control" condition) is 1.4, then the differential expression of the gene is (2.1 – 1.4) = 0.7.

```
summarization[data_] :=
 Module[{data1, polishresult, summary, fitdata},

 data1 = Transpose[data];

 polishresult = medianpolish[data1];
 fitdata = data1 - polishresult;

 Map[Mean, Partition[Flatten[fitdata],
     (Length[fitdata[[1]]] * commonpssize)]]


medianpolish[data_] :=
 Module[{data1, startrowmedians, endrowmedians,
    startcolmedians, endcolmedians, rowmedians,
    overalleffect, columnmedians, medianofcolmedian,
    cumroweffects, cumcoleffects, cumgrandeffect},

 data1 = data;
 startrowmedians = {10.5};
 endrowmedians = {5.5};
 startcolmedians = {10.5};
 endcolmedians = {5.5};

 cumroweffects = 0.0;
 cumcoleffects = 0.0;
 cumgrandeffect = 0.0;


  While[
   (Abs[Total[Map[Abs, startrowmedians]] -
        Total[Map[Abs, endrowmedians]]] >= 0.25) ⋁

    (Abs[Total[Map[Abs, startcolmedians]] -
        Total[Map[Abs, endcolmedians]]] >= 0.25),

 rowmedians = Map[Median, data1];
 startrowmedians = rowmedians;
 startcolmedians =
    Map[Median, MapThread[List, data1]] // N;

 data1 = data1 - rowmedians;
 overalleffect = Median[rowmedians] // N;
 cumgrandeffect = cumgrandeffect + overalleffect;
 rowmedians = rowmedians - overalleffect;
 cumroweffects = cumroweffects + rowmedians;
```

```
 columnmedians = Map[Median, MapThread[List, data1]] // N;
 data1 = Transpose[Transpose[data1] - columnmedians];
   medianofcolmedian = Median[columnmedians];
 cumgrandeffect = cumgrandeffect + medianofcolmedian;
 columnmedians = columnmedians - medianofcolmedian;
 cumcoleffects = cumcoleffects + columnmedians;

 endrowmedians = Map[Median, data1];
 endcolmedians = columnmedians;];

 data1]

exparrays = Table[Flatten[quantnorm[experimentchips[[i]]]],
  {i, 1, Length[experimentchips]}];
controlarrays = Table[Flatten[quantnorm[controlchips[[i]]]],
  {i, 1, Length[controlchips]}];

expsummary = summarization[exparrays];
contsummary = summarization[controlarrays];
diffexp = expsummary - contsummary;
```

Here are the distributions of gene expression in the saliva of pancreatic cancer and healthy patients, as well as the distribution of differentially expressed genes between the two states (Figure 4).

```
temp3 = Histogram[expsummary, ChartStyle → Orange,
   Frame → True,
   FrameLabel → {{"Fluorescence Count", ""},
     {"Fluorescence Bins",
      "Gene Expression in the Saliva of Pancreatic
        Patients"}}];
temp4 = Histogram[contsummary, ChartStyle → Green,
   Frame -> True,
   FrameLabel → {{"Fluoresence Count", ""},
     {"Fluorescence Bins",
      "Gene Expression in the Saliva of Healthy
        Patients"}}];
temp5 = Histogram[diffexp, ChartStyle → Red, Frame → True,
   FrameLabel → {{"Differential Fluorescence Count", ""},
     {"Differential Fluorescence Bins",
      "Differential Gene Expression"}}];
GraphicsColumn[{temp3, temp4, temp5}]
```



▲ **Figure 4.** Histograms of the summarized (post median polish) gene expression values contained in the saliva of pancreatic cancer and healthy patients, as well as the difference in gene expression between those two biological groups.

From these results, simulations are performed to calculate probability p-values, which answer the question, If we were to repeat this experiment many times, under the same experimental conditions as this study, how often would we find results as (or more) extreme than we have observed for each gene in the current study? The simulations performed to calculate the p-values use the corrected fluorescent signal values contained within the experimental and control datasets and thus make the assumptions that the corrections are valid and that the data is now a good representative of the biological "truth" being studied. If these assumptions are correct, the probability values reported help the end user to gauge how rare a gene's measure of differential expression is, but not—as is traditionally used in the microarray community—to make a decision about statistical or biological significance.

Following this, the algorithm organizes the output into a more human-readable table.

```
expdist = FindDistributionParameters[expsummary,
   NormalDistribution[mean, stdev]];
controldist = FindDistributionParameters[contsummary,
   NormalDistribution[mean, stdev]];

simulateddiff =
  RandomVariate[NormalDistribution[expdist[[1, 2]],
     expdist[[2, 2]]], 100 000] -
   RandomVariate[NormalDistribution[controldist[[1, 2]],
     controldist[[2, 2]]], 100 000];

fastSelect := Compile[{{realde, _Real}, {simde, _Real, 1}},
   If[Negative[realde],
    (Length[Select[simde, # ≤ realde &]] / Length[simde]),
    (Length[Select[simde, # ≥ realde &]] / Length[simde])]];

pvalues =
  ParallelTable[fastSelect[diffexp[[i]], simulateddiff],
    {i, 1, Length[diffexp]}] // N;

transcriptids = minicdffile[[All, 1, 2]];
ginoutput = miniginfile[[All, 8 ;; 10]];
alldatafinal =
  MapThread[List, {expsummary, contsummary, diffexp,
    pvalues, transcriptids, ginoutput}];
```

Here is a random sample of our current results.

The output columns are as follows:

Column 1: the signal-corrected $\log_2$ fluorescence intensity of gene expression in the experimental condition

Column 2: the signal-corrected $\log_2$ fluorescence intensity of gene expression in the control condition

Column 3: the measure of differential expression, obtained by subtracting column 2 from column 1

Column 4: the *p*-value obtained through simulation

Column 5: the probeset name as assigned by Affymetrix

Column 6: descriptive information for the probeset including the genbank accession number, gene name, and gene product information

Due to the length of descriptive data in column 6, the output here has been purposefully rearranged to print column 6 underneath each data point's first five column entries.

```
sample1 = alldatafinal[[500 ;; 505]][[All, 1 ;; 5]];
sample2 = alldatafinal[[500 ;; 505]][[All, 6]];
sample2[[1, 3]] =
  "stress-associated endoplasmic reticulum protein
    1\nribosome-associated membrane protein 4";
sample2[[5, 3]] =
  "palmitoyl-protein thioesterase 1
    \n(ceroid-lipofuscinosis, neuronal 1, infantile)";
sample2[[6, 3]] =
  "Tax 1 (human T-cell leukemia virus type I) \nbinding
    protein 1";
Text[Grid[Riffle[sample1, sample2]]]
```

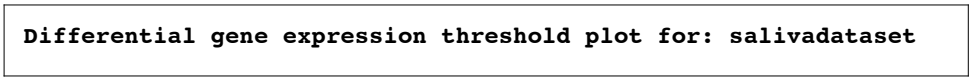
5.8704	5.87253	−0.00213572	0.49547	200971_s_at
gb:NM_014445.1	SERP1	stress−associated endoplasmic reticulum protein 1 ribosome−associated membrane protein 4		
5.8283	5.79987	0.0284325	0.46083	200972_at
gb:BC000704.1	None	tetraspan 3		
5.76611	5.78402	−0.017911	0.47338	200973_s_at
gb:NM_005724.1	TSPAN−3	tetraspan 3		
5.77746	5.8305	−0.0530355	0.42262	200974_at
gb:NM_001613.1	ACTA2	alpha 2 actin		
6.45964	6.43239	0.0272519	0.46244	200975_at
gb:NM_000310.1	PPT1	palmitoyl−protein thioesterase 1 (ceroid−lipofuscinosis, neuronal 1, infantile)		
5.90178	5.91914	−0.0173576	0.47419	200976_s_at
gb:NM_006024.2	TAX1BP1	Tax 1 (human T−cell leukemia virus type I) binding protein 1		

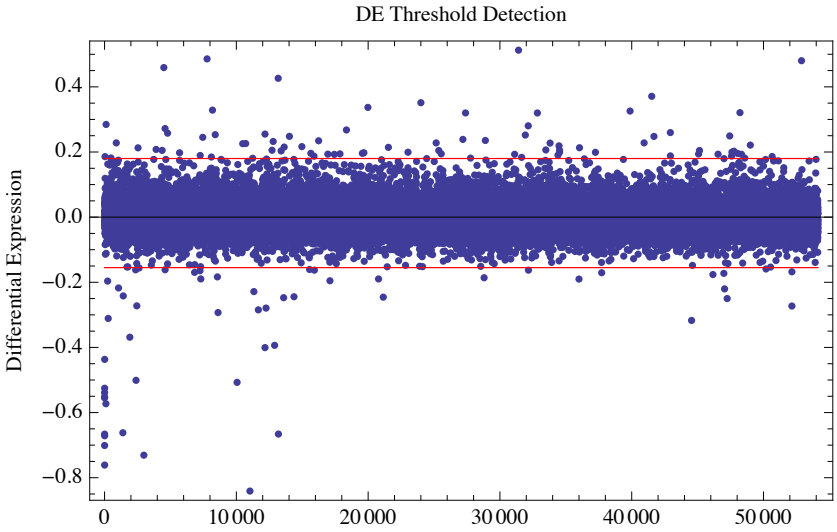
The final steps in processing the data include establishing the upper and lower thresholds for determining when a gene is considered up-regulated (turned "on") in the experimental condition versus the control condition, and when a gene is considered down-regulated (turned "off"). Further, the algorithm saves the final results and provides a summary report of the completed analysis.

```
uplowiter = 0.0;
upmiditer = 0.005;
uphighiter = 0.01;
upbottomface = Select[diffexp, uplowiter ≤ # < upmiditer &];
uptopface = Select[diffexp, upmiditer ≤ # < uphighiter &];

While[(Length[uptopface] / Length[upbottomface] > 0.50),

 uplowiter = upmiditer; upmiditer = uphighiter;
 uphighiter = (uphighiter + 0.005);
 upbottomface = Select[diffexp, uplowiter ≤ # < upmiditer &];
 uptopface = Select[diffexp, upmiditer ≤ # < uphighiter &];]

downlowiter = 0.0; downmiditer = -0.005; downhighiter = -0.01;
downbottomface =
 Select[diffexp, downlowiter ≥ # > downmiditer &];
downtopface =
 Select[diffexp, downmiditer ≥ # > downhighiter &];

While[(Length[downtopface] / Length[downbottomface] > 0.5),
 downlowiter = downmiditer; downmiditer = downhighiter;
 downhighiter = (downhighiter - 0.005);
 downbottomface =
  Select[diffexp, downlowiter ≥ # > downmiditer &];
 downtopface =
  Select[diffexp, downmiditer ≥ # > downhighiter &];]

dedatafinal = Select[alldatafinal,
   #[[3]] ≥ upmiditer ⋁ #[[3]] ≤ downmiditer &];
```

```
date = DateString[];
date = DateString[date,
  {"Month", "Day", "Year", "Hour", "Minute", "Second"}];
foldername = StringJoin[ToString[studyname], " - ", date];
SetDirectory[savelocationroot];
savelocationfinal = CreateDirectory[foldername];
SetDirectory[savelocationfinal];

Put[alldatafinal, StringJoin[ToString[studyname],
   " - allgenes"]];
Put[dedatafinal, StringJoin[ToString[studyname],
  " - degenes"]];
Export[StringJoin[ToString[studyname], " - allgenes.csv"],
 alldatafinal];
Export[StringJoin[ToString[studyname], " - degenes.csv"],
  dedatafinal];

dethresh1 = ListPlot[diffexp,
   FrameLabel → {{"Differential Expression", ""},
     {"", "DE Threshold Detection"}}, PlotRange → Full,
   Frame → True, ImageSize → Medium];
dethresh2 = Plot[upmiditer, {x, 0, Length[diffexp]},
   PlotStyle → Directive[Red]];
dethresh3 = Plot[downmiditer, {x, 0, Length[diffexp]},
   PlotStyle → Directive[Red]];
dethresh4 = Show[dethresh1, dethresh2, dethresh3];

Print[];
Print[];
Print[
 Style["Differential gene expression threshold plot for: ",
  Bold], Style[studyname, Bold]]
Print[];
dethresh4
Print[];
Print[];
Print[Style["Report summary for study: ", Bold],
 Style[studyname, Bold]]
Print[];
Print["1. All data saved to: ", savelocationfinal];
Print[];
```

```
Print[
 "2. Number of transcripts interrogated by algorithm: ",
 Length[alldatafinal]]

If[qnormversion === all,
  Print[
   "        A: Quantile normalization: all chips at once"],
  Print[
   "        A: Quantile normalization: by condition"]];

Print[
  "        B. Median polish: rows contain probe data;
    columns contain chip data"];

Print[];
Print[
 "3. Number of differentially expressed transcripts
   detected: ", Length[dedatafinal]]
Print[
 "        A. Cutoff threshold for UP regulated
   transcripts: ", upmiditer]
Print[
 "        B. Cutoff threshold for DOWN regulated
   transcripts: ", downmiditer]
Print[];
endtime = AbsoluteTime[] - starttime;
Print["4: Computational time: ", endtime, " seconds"];
```





Report summary for study: salivadataset

1. All data saved to:
C:\Users\Wookie\Desktop\salivadataset - 10262013183835

2. Number of transcripts interrogated by algorithm: 54 130

A: Quantile normalization: all chips at once

B. Median polish: rows
contain probe data; columns contain chip data

```
3. Number of differentially expressed transcripts detected: 136
```

```
        A. Cutoff threshold for UP regulated transcripts: 0.18
```

```
        B. Cutoff threshold for DOWN regulated transcripts:
 -0.155
```

```
4: Computational time: 567.8564795 seconds
```

AffyDGED saves all the data and a list of differentially expressed genes to the location defined by the user in the template above. Further, both lists of data are saved in two formats, one convenient for users to continue to explore the data in *Mathematica* and the other conveniently readable in Microsoft Excel.

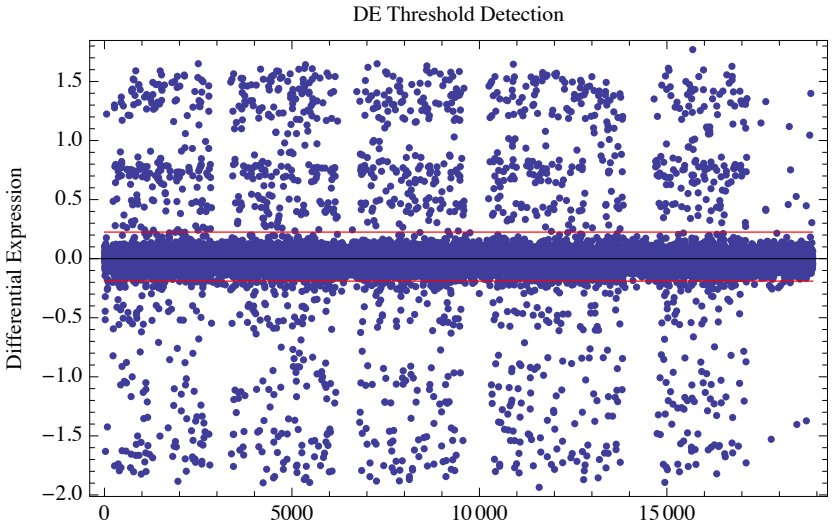
Support Files for AffyDGED

Affymetrix gene expression array technology utilizes a suite of library files containing important annotation information about the layout and content of its microarray products. AffyDGED can analyze any Affymetrix gene expression experiment as long as the .cdf (chip description file) and .gin (gene information) library files associated with the specific type of microarray chip being used are provided. These files are freely available to the public at www.affymetrix.com/support/technical/libraryfilesmain.affx. The variable "affycdflocation", defined in the user template, holds the directory location of where the user has stored the .cdf file, and AffyDGED expects the .gin file to also be stored in the same location. Having the .cdf and .gin files together happens naturally, as they are packaged together by Affymetrix and unzip to the same location upon download.

How Accurate Are AffyDGED Results?

As mentioned previously, there is no universally correct algorithm for processing microarray data in all experimental circumstances. AffyDGED was developed under the guidance of [2] because the exact expression of genes in this study was precisely controlled and therefore is a useful gauge of how effectively a microarray analysis algorithm is performing.

Supplemental file 5 in [2] describes the 1,944 differentially expressed genes and 3,426 non-differentially expressed genes that were purposefully "spiked-in" to the microarray experiment. When this dataset is analyzed by the AffyDGED algorithm described here, the thresholds for determining "up" and "down" gene expression are calculated to be 0.225 and −0.19, respectively (the red lines in Figure 5). Establishing the thresholds for determining differential expression relies on the observation illustrated in Figure 5, that a plot of the processed data always reveals a tight clustering of data about the line $y = 0$. As the reader scans above and below that axis, note how the density of data noticeably separates from the cluster along that line. This observation was used to develop code that scans vertically up and down in small increments and establishes a breakpoint in each direction any time the density of data at a vertical position is 50% less than it was at the previous increment. These breakpoints become the thresholds for determining differentially expressed up and down genes.



▲ **Figure 5.** The resulting differential expression threshold determination plot by AffyDGED when processing the data in [2] (accession number GSE21344).

The AffyDGED algorithm identifies 1,832 genes as differentially expressed in [2]. Of these, 1,591 genes overlap with the true 1,944 differentially expressed genes for a true positive rate of (1591 / 1944) = 0.818. This means AffyDGED was unable to correctly identify 18% of the true list of differentially expressed genes. While perhaps surprising to readers unfamiliar with microarray analysis, this places AffyDGED's performance among the top performers of leading algorithms identified by [2]. State of the art at this time in microarray analysis means accepting a 1 out of 5 "miscall" rate in differential expression detection. Because of the complexity of microarray technology and of all the steps that occur prior to the actual algorithmic analysis of the data, it is important to realize that at least some of the miscalls by any microarray algorithm are really due to factors that are poorly controlled for by the underlying technology, and do not indicate a weakness in the algorithm itself [8].

Performance Timings on Different Datasets

To gauge the performance of AffyDGED, several publicly available datasets of different sizes and complexity were profiled. The first row of Table 1 shows the series accession number for each dataset available at NCBI's Gene Expression Omnibus. All timings were obtained using quantile normalization with all chips (i.e. qnormversion set to "all"). Timings were acquired running *Mathematica* 9.0 under Windows 7 (64 bit) using an Intel Core i5-2500K processor overclocked to 4.48 Ghz.

Series accession #	GSE14245	GSE21344	GSE40693	GSE11899	GSE31660
Chip Type	Human	Drosophila	E. coli	Mouse	Grape
# of chips	24	18	8	10	7
# of genes processed per chip	53 130	18 890	10 119	45 032	16 294
Timing (seconds)	437.7	126.2	49.3	213.4	103.5

▲ **Table 1.** Performance timings of AffyDGED using five different publicly available datasets.

AffyDGED performs very well, in all cases completing its analysis in less than seven and a half minutes. The largest chip in this comparison is the Human chip, where each of the processed files contains 13.2 Mb of data. AffyDGED is able to process the combined (24 chips × 13.2 Mb per chip) = 316.8 Mb worth of data in a practically usable time frame.

Conclusion

Microarray technology continues to be heavily used by the biomedical and basic science research communities throughout the world. AffyDGED brings a contemporary algorithm useful in the real world to the *Mathematica* user community interested in exploring fundamental biology questions with their favorite computational tool chest.

References

[1] M. Schena, D. Shalon, R. W. Davis, and P. O. Brown, "Quantitative Monitoring of Gene Expression Patterns with a Complementary DNA Microarray," *Science*, **270**(5235), 1995 pp. 467–470. www.jstor.org/stable/2889064.

[2] Q. Zhu, J. C. Miecznikowsk, and M. S. Halfon, "Preferred Analysis Methods for Affymetrix GeneChips. II. An Expanded, Balanced, Wholly-Defined Spike-in Dataset," *BMC Bioinformatics*, **11**(285), 2010. doi:10.1186/1471-2105-11-285.

[3] B. M. Bolstad, R. A. Irizarry, M. Åstrand, and T. P. Speed, "A Comparison of Normalization Methods for High Density Oligonucleotide Array Data Based on Variance and Bias," *Bioinformatics*, **19**(2), 2003 pp. 185–193. doi:10.1093/bioinformatics/19.2.185.

[4] J. W. Tukey, *Exploratory Data Analysis*, Reading, MA: Addison-Wesley, 1977.

[5] C. A. Harrington, C. Rosenow, and J. Retief, "Monitoring Gene Expression Using DNA Microarrays," *Current Opinion in Microbiology*, **3**(3), 2000 pp. 285–291. doi:10.1016/S1369-5274(00)00091-6.

[6] J. Lovén, D. A. Orlando, A. A. Sigova, C. Y. Lin, P. B. Rahl, C. B. Burge, D. L. Levens, T. I. Lee, and R. A. Young, "Revisiting Global Gene Expression Analysis," *Cell*, **151**(3), 2012 pp. 476–482. doi:10.1016/j.cell.2012.10.012.

[7] L. Zhang, J. J. Farrell, H. Zhou, D. Elashoff, D. Akin, N.-H. Park, D. Chia, and D. T. Wong, "Salivary Transcriptomic Biomarkers for Detection of Resectable Pancreatic Cancer," *Gastroenterology*, **138**(3), 2010 pp. 949–957.e7. doi:10.1053/j.gastro.2009.11.010.

[8] R. A. Rubin, "A First Principles Approach to Differential Expression in Microarray Data Analysis," *BMC Bioinformatics*, **10**(292), 2009. doi:10.1186/1471-2105-10-292.

T. Allen, "Detecting Differential Gene Expression Using Affymetrix Microarrays," *The Mathematica Journal*, 2013. dx.doi.org/doi:10.3888/tmj.15-11.

About the Author

Todd Allen is an associate professor of biology at HACC-Lancaster. His interest in computational biology using *Mathematica* took shape during his postdoctoral research years at the University of Maryland, where he developed a custom cDNA microarray chip to study gene expression changes in the fungal pathogen *Cryphonectria parasitica*.

Todd D. Allen, Ph.D.
Harrisburg Area Community College (Lancaster Campus)
East 206 R
1641 Old Philadelphia Pike
Lancaster, PA 17602
tdallen@hacc.edu