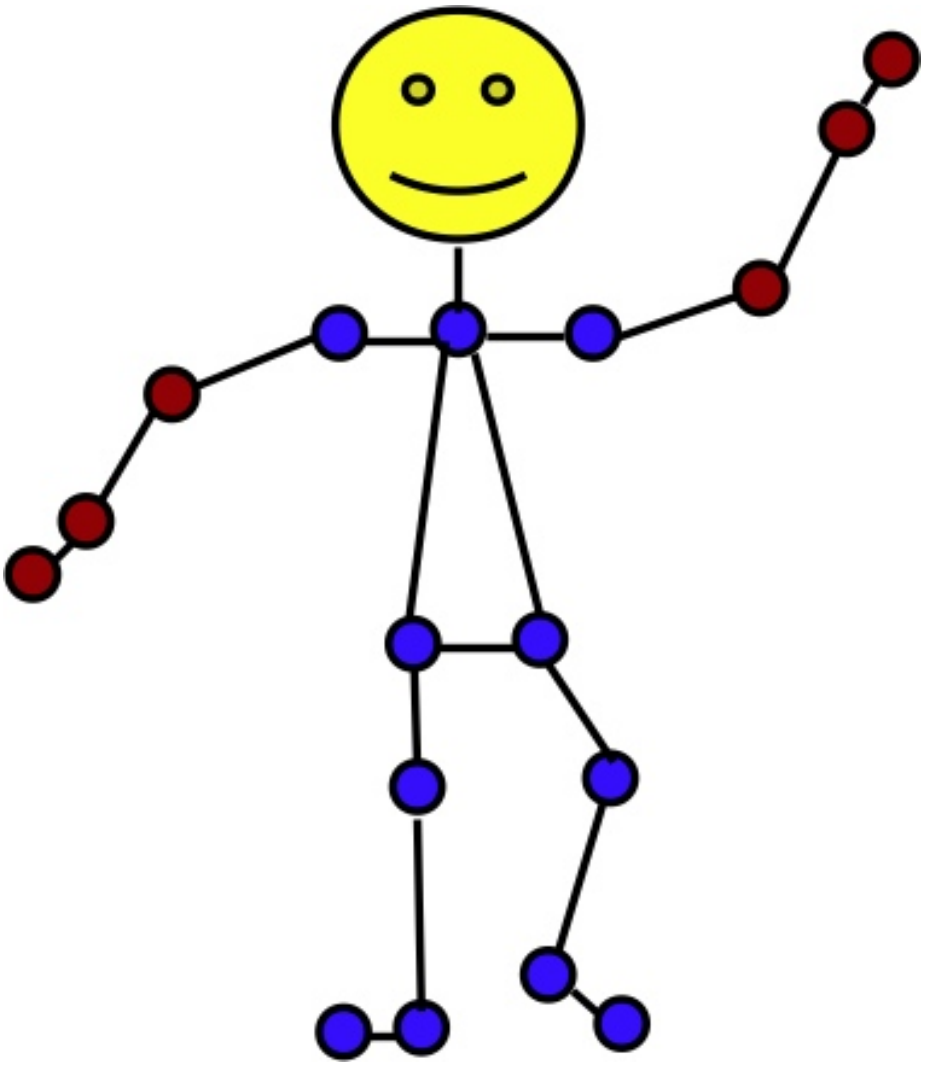
ELBOW – Evaluating foLd change By the lOgit Way



Contents

ELBOW – Evaluating foLd change By the lOgit Way

October 17, 2016

1 FOREWORD

This in depth tutorial about ELBOW is designed for those (especially biologists) who may need to review logit curves, logistic regression, or cluster analysis and their applications in determining biological significance in transcriptomics. If you are already familiar with these concepts, or you wish to start using ELBOW and you are already familiar with R and Bioconductor you may choose to go directly to "Quick Usage" file.

2 What is "ELBOW"?

ELBOW is an improved fold test method for determining values that are biologically significant when using next generation transcriptomics such as microarray and RNA Seq. ELBOW is likely applicable to other large biological datasets.

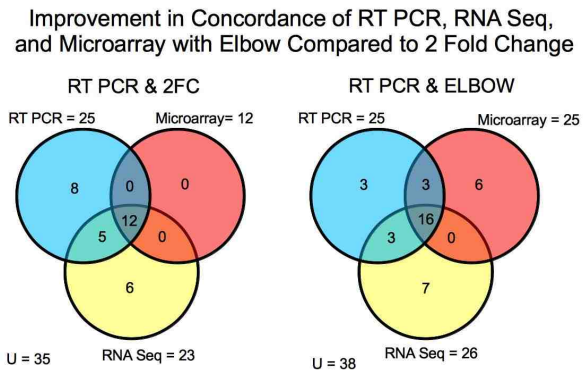
3 How is "ELBOW" better than Fold test?

Fold test is notorious for having high levels of false positives and false negatives.

The "ELBOW" cut off is determined by the variance of the dataset being examined rather than by an arbitrary cut off such as "two fold up and down." This reduces false positives and false negatives.

Unlike simple fold testing, "ELBOW" can be successfully applied to datasets that are not symmetric about zero.

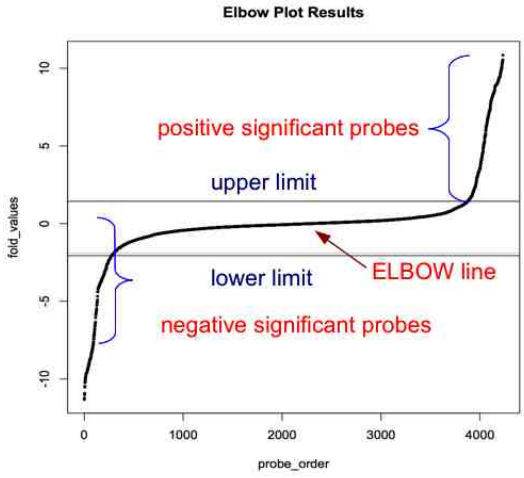
Below are results, one from a microarray and a RNA seq with significance determined by a standard fold test compared to RT PCR and the other with significance determined by ELBOW. ELBOW performed much better with lower rates of false positives and false negatives and improved match to RT PCR results.



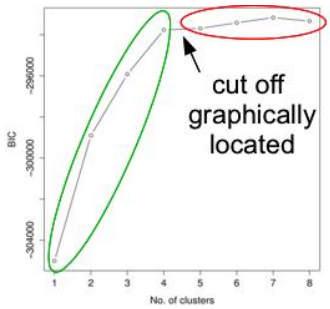
4 How does ELBOW works?

Fold values are calculated for your dataset. The values are then ordered from lowest (most negative) to highest (most positive).

Probe order is plotted on the x axis. Fold values are plotted on the y axis.



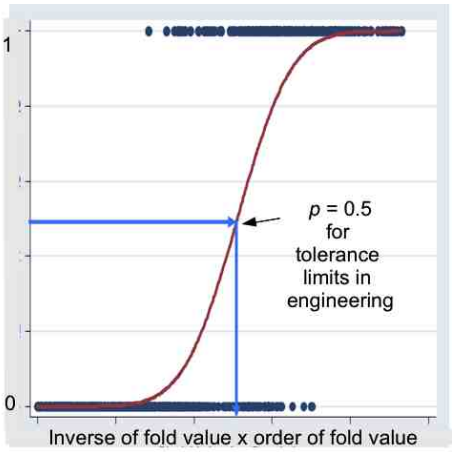
ELBOW generates a graphical representation of the data in the form of a "logit" curve. All values above and below the midpoints of the two "elbows" of the logit curve are considered biologically significant.



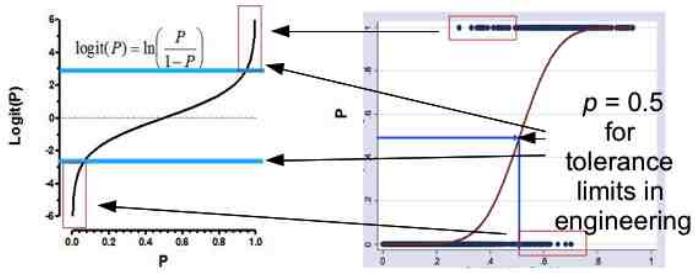
"lowClust: a Bioconductor package for automated gating of flow cytometry data"
Lo K, Hahne F, Brinkman RR, Gottardo R - BMC Bioinformatics (2009)"

ELBOW method came out of our observations about cluster analysis. This example graphical cluster analysis shows that a model to explain a dataset will work well for the first four clusters (green) but because the data levels off, the model will not apply for later clusterings (red). Examining and choosing a point on a curve to divide data is a useful method for many types of analyses. We decided to apply cluster analysis to determining biological significance of transcriptomics.

4.1 Mathematically...



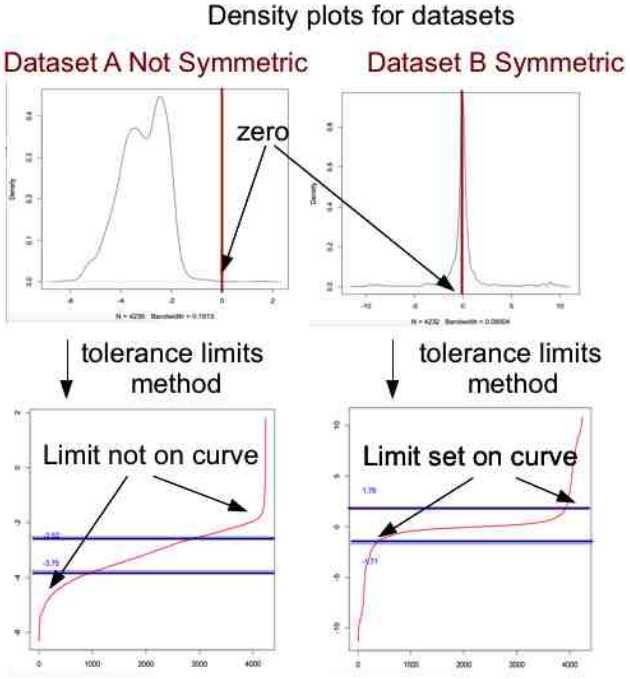
ELBOW method also derives from probability theory and statistics methods used for examining data that gives a yes/no result. In the case of the ELBOW method we could use the inverse of the fold value multiplied by the order of the probes (from lowest to highest fold values) on the x axis and set positive probes = 1 and negative value = 0. The resulting graph would look like this. By applying a logistic curve to the value we can determine a midpoint. The midpoint provides valuable information.



Plotting things like "inverse of fold value x order of fold value" is a bit confusing but we don't actually have to do this because the inverse of the logistic curve is the logit curve. We already found the logit curve directly by the ELBOW method. We can use established methods, like "tolerance limits" from engineering, that are known to apply to logit curves in order to better analyze our data.

4.2 Symmetry About Zero

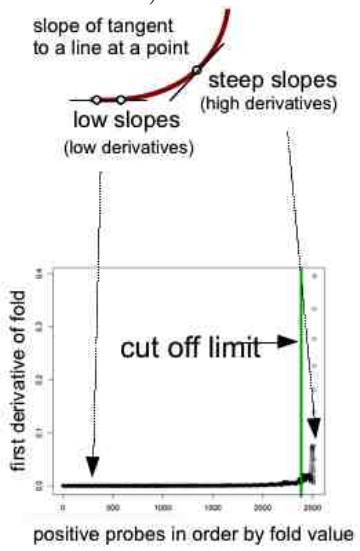
One important limitation to standard logit methods, such as tolerance limits, is they depend on the dataset having a symmetric distribution about zero.



Biological data from transcriptomics is often not symmetric about the midpoint limiting the usefulness of standard methods.

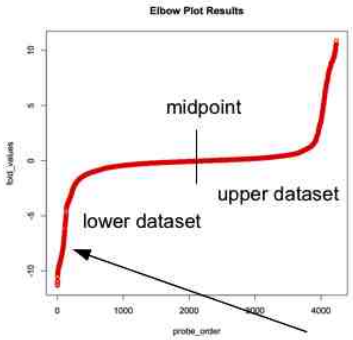
4.3 First Derivative Method

The manual method for finding the cut off limits uses derivatives. A midpoint to the data is found. Data is divided in half at the midpoint to create two separate datasets. Beginning at the midpoint, the change in the value to the next point, (the derivative) is determined. These derivatives are plotted. Since the derivative is the same as the slope of the tangent to the line, a large change in derivative equals a steep slope. The cut off for the manual method is set (by experimentation) where the derivative jumps to 0.03.



The manual method works very well but is time consuming to calculate. We generalized the manual method using principals of pattern recognition for image analysis to automatically determine limits.

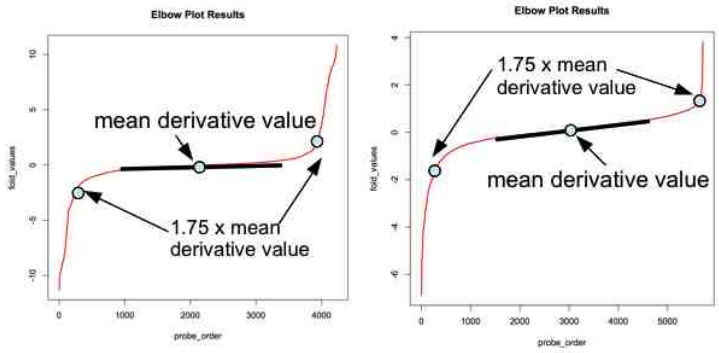
As in the manual method, the data is first divided into two at the midpoint. The two data subsets are smoothed by fitting a cubic smoothing spline. The derivatives for the points on the smoothed curves are determined.



NOTE: how the density of points on the tails of the curve is less than the density at the midpoint.

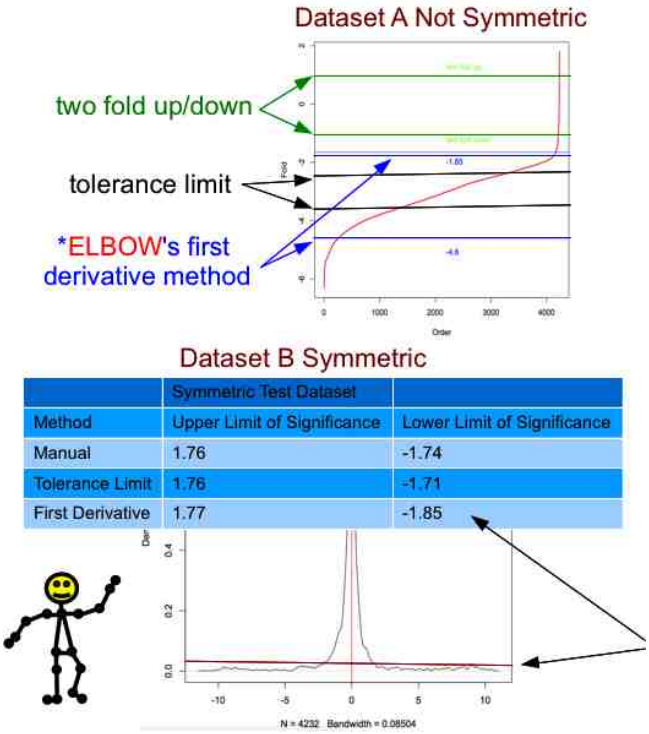
The mean derivative value of both the upper and the lower smoothed curves is calculated. Since most data points are not significant and fall on the flat central line, the mean derivative is approximately the same as the derivative for the midpoint on any dataset we tested.

Further testing showed that the value 1.75 X mean value always corresponded to the midpoint of the curve even on datasets not symmetric about zero or having unequal tails.



4.4 ELBOW Works Best! (comparison on non-symmetrical data)

First derivative method also gives a good match for standard methods in datasets that are symmetric.



First derivative method works even better than our original manual method.

Compare results from Dataset B, a dataset symmetric about zero except for higher values in the positive tail. Only first derivative method detected that subtle difference.

4.5 Error Limits for ELBOW

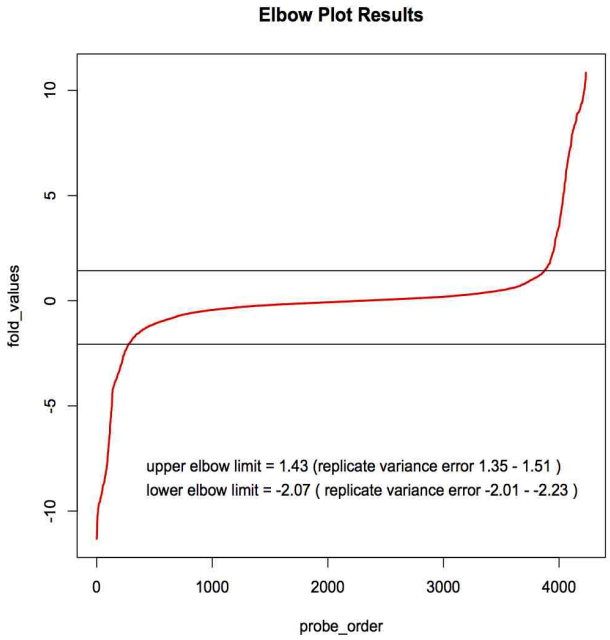
Replicates	Upper ELBOW	Lower ELBOW
A1/B1	1.5	-2
A1/B2	2.1	-2.3
A1/B3	1.7	-1.9
A2/B1	1.2	-2.2
A2/B2	1.6	-1.9
A2/B3	1.4	-1.8
A3/B1	1.1	-2
A3/B2	1.9	-2.1
A3/B3	1.4	-1.5

All possible pairing of initial and final conditions. A1, A2, A3 (initial),

B1,B2,B3 (final) replicates, ELBOW limits averaging all replicates 1.9, -1.8. Upper limit all replicates 1.9; Lower limit all replicates -1.8; Highest Upper Limit 2.1; Lowest Upper Limit 1.2; Lowest Lower Limit -2.3; Highest Lower Limit -1.2; Result Upper Limit 1.9 (2.1 to 1.2); Result Lower Limit -1.8 (-1.5 to -2.3)

ELBOW calculates an upper and lower error bound on the ELBOW limits by taking all possible pairs of replicates in initial condition and final condition and finding the ELBOW limits for all these pairs. A range of limits are determined. The more variance in the replicates, the broader the range.

ELBOW measures the variance of the fold test results using subsets of replicates to determine upper and lower bounds error. These are provided in a text report.



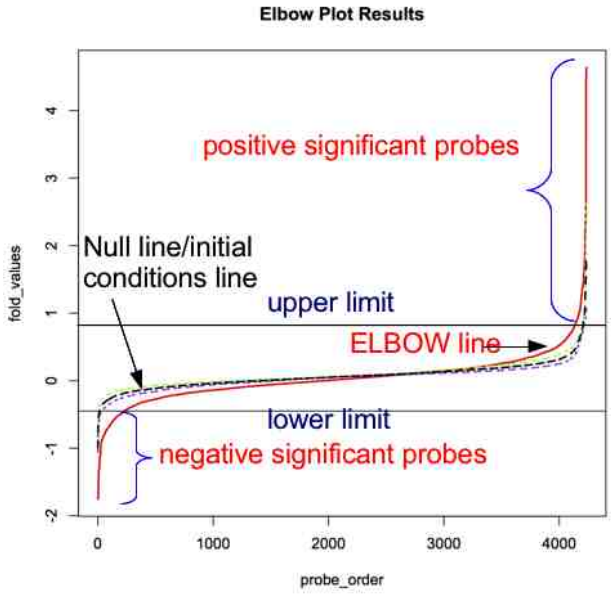
4.6 Null Result for ELBOW

Null is calculated by taking all possible pairs of replicates in initial conditions and finding the fold value difference for all these pairs. The median value for each probe forms the null set. ELBOW also gives high and low values.

A1, A2, A3 replicates in initial condition

	A1/A2	A1/A3	A2/A3	Median/Null	Low	High
Probe 1	1.2	1.3	1.1	1.2	1.1	1.3
Probe 2	1.3	1.4	1.0	1.3	1.0	1.4
Probe 3	1.1	1.0	1.1	1.1	1.0	1.1
Probe 4	0.9	1.0	1.4	1.0	0.8	1.4
Probe 5	0.8	1.2	1.2	1.2	0.8	1.2

(Note that if you start with only two replicates null = high = low and ELBOW will only report the null value.)

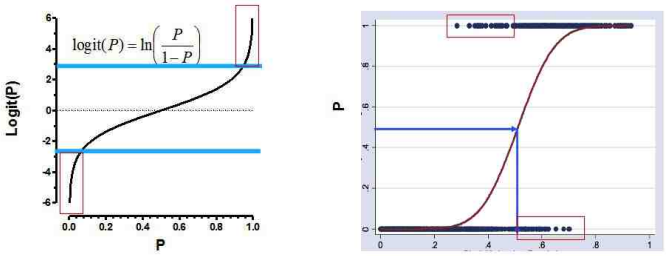


The null line graphically provides information on the quality of your initial condition replicate set and a visual comparison for exploring the magnitude of the change between initial and final conditions.

In addition to the null line there is an upper (green) and lower (purple) null line which is based on the highest and lowest values for the initial conditions. This gives information on the variance of the initial condition replicates.

4.7 P value for the ELBOW result

The entire basis of the ELBOW model is that the dataset creates a logit curve that can be translated back to a logistic curve with a binomial fit . Therefore ELBOW provides a test with a p value result to make sure that the dataset fits the model.



(If the data does not fit, different preparation may be required. This will be explained further in the trouble shooting section.)

ELBOW provides a p value which is the log chi squared p value of the fit to the ELBOW model. ($2^{\text{P value}}$ should be at or near 1 because we expect it to fit.)

If the p value is not provided this means the data does not fit the logit model on which ELBOW is based and the results can not be used to determine biological significance.

NOTE: The $\log \chi^2$ ***P-value displayed by our program is <u>NOT</u> the level of significance for any one probe!***

4.8 One final tip (before you begin using ELBOW)

Standard fold tests are presumed to give more false positive and false negatives that purely statistical methods.

Comparison of ELBOW results with Statistical Analyses of Microarray (SAM) for a six replicate microarray result where that data set was also symmetric about zero do show more positive values with ELBOW. ELBOW did not produce more false negatives on this test.

If you have six replicates in both your initial and final conditions and the dataset is symmetric about zero, then a statistical method such as SAM should give better results than ELBOW.

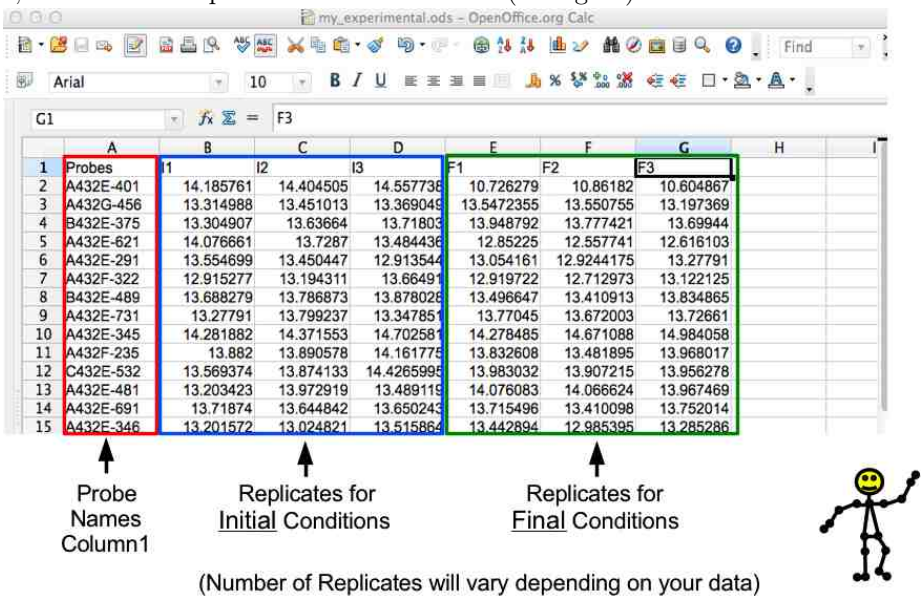
Name of Test	Significant Genes
Statistical Analysis of Microarray	102
Two Fold Test	1019
ELBOW Test	633
T Test	240
Fold Test plus T Test	223
ELBOW plus T Test	205
Fold Test plus SAM	102
ELBOW plus SAM	102

We recommend using ELBOW as the first step to determining biological significance if you have less than six replicates in both initial and final conditions and/or your dataset is not symmetric about zero. You may also choose to add

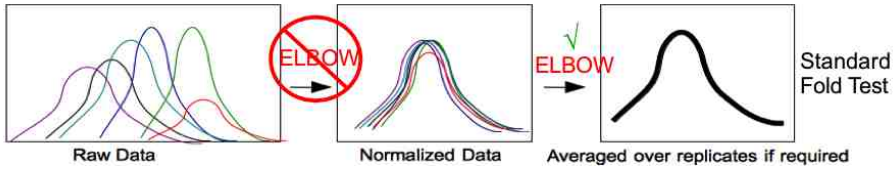
additional methods to refine ELBOW's results such as t testing. Consult with your statistician or bioinformatician for further information.

5 Excel Pipeline

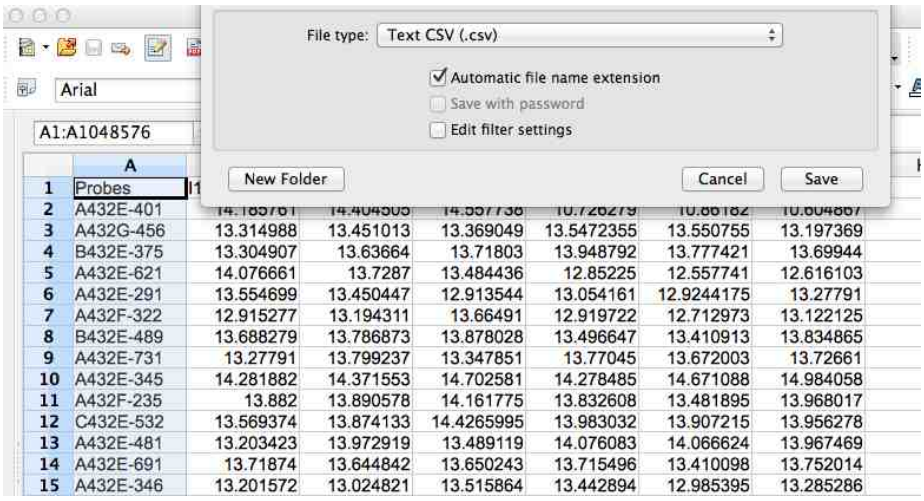
First, place your data in the correct column order. From first to last column: the probe names should be in the first column, the initial replicates should come next, and the final replicates should come last (see figure).



Second, check that your data is normalized. Data from programs like "pliers" usually arrives already "normalized", especially when the data is from a microarray. Normalized means the data has been smoothed by taking a log value (usually 2) for each value. If you are not sure ask the data provider.



Third, save your data in a .csv (comma separated values) format file. Other formats may not work. Be sure to give your file a meaningful filename; the filename will appear at the top of the final elbow output graph.



	A						
1	Probes						
2	A432E-401	14.185761	14.404505	14.557736	10.726279	10.86182	10.804867
3	A432G-456	13.314988	13.451013	13.369049	13.5472355	13.550755	13.197369
4	B432E-375	13.304907	13.63664	13.71803	13.948792	13.777421	13.69944
5	A432E-621	14.076661	13.7287	13.484436	12.85225	12.557741	12.616103
6	A432E-291	13.554699	13.450447	12.913544	13.054161	12.9244175	13.27791
7	A432F-322	12.915277	13.194311	13.66491	12.919722	12.712973	13.122125
8	B432E-489	13.688279	13.786873	13.878028	13.496647	13.410913	13.834865
9	A432E-731	13.27791	13.799237	13.347851	13.77045	13.672003	13.72661
10	A432E-345	14.281882	14.371553	14.702581	14.278485	14.671088	14.984058
11	A432F-235	13.882	13.890578	14.161775	13.832608	13.481895	13.968017
12	C432E-532	13.569374	13.874133	14.4265995	13.983032	13.907215	13.956278
13	A432E-481	13.203423	13.972919	13.489119	14.076083	14.066624	13.967469
14	A432E-691	13.71874	13.644842	13.650243	13.715496	13.410098	13.752014
15	A432E-346	13.201572	13.024821	13.515864	13.442894	12.985395	13.285286

Finally, load your data into R and run your data through ELBOW:

```
>     # load the ELBOW library
>     library("ELBOW")
>     # Set the filename (change everything after the <- to ``csv_file'',
>     # where csv_file is the filename of your CSV data file)
>     filename <- system.file("extdata", "EcoliMutMA.csv", package = "ELBOW")
>     # Read in your CSV file
>     csv_data <- read.csv(filename)
>     # set the number of initial and final condition replicates both to three
>     init_count  <- 3
>     final_count <- 3
>     # Parse the probes, intial conditions and final conditions
>     # out of the CSV file.  Please see: extract_working_sets
>     # for more information.
>     #
>     # init_count should be the number of columns associated with
>     #       the initial conditions of the experiment.
>     # final_count should be the number of columns associated with
>     #       the final conditions of the experiment.
>     working_sets <- extract_working_sets(csv_data, init_count, final_count)
>     probes <- working_sets[[1]]
>     initial_conditions <- working_sets[[2]]
>     final_conditions <- working_sets[[3]]
>     # Uncomment to output the plot to a PNG file (optional)
>     # png(file="output_plot.png")
>
>     # Analyze the elbow curve.
>     sig <- analyze_elbow(probes, initial_conditions, final_conditions)
```

```
[1] "rowsums"
[1] "fold"
[1]  0.06873333 -0.08933333  0.34013333  0.19313333  0.00940000 -0.02596667
[1] "bound data"
    ID_REF        fold
1 1001_115  0.06873333
2  1002_33 -0.08933333
3 1003_942  0.34013333
4 1004_552  0.19313333
5 1005_657  0.00940000
6 1006_393 -0.02596667
[1] "firsta_data"
    ID_REF        fold
1 1001_115  0.06873333
2  1002_33 -0.08933333
3 1003_942  0.34013333
4 1004_552  0.19313333
5 1005_657  0.00940000
6 1006_393 -0.02596667
[1] "sorted"
[1] "headers"
[1] "rowsums"
[1] "fold"
[1]  0.2138 -0.2044  0.1604  0.1214 -0.1342 -0.0477
[1] "bound data"
    ID_REF    fold
1 1001_115  0.2138
2  1002_33 -0.2044
3 1003_942  0.1604
4 1004_552  0.1214
5 1005_657 -0.1342
6 1006_393 -0.0477
[1] "firsta_data"
    ID_REF    fold
1 1001_115  0.2138
2  1002_33 -0.2044
3 1003_942  0.1604
4 1004_552  0.1214
5 1005_657 -0.1342
6 1006_393 -0.0477
[1] "sorted"
[1] "headers"
[1] "rowsums"
[1] "fold"
[1]  0.2071  0.0046  0.0323  0.3172  0.0230 -0.0754
[1] "bound data"
```

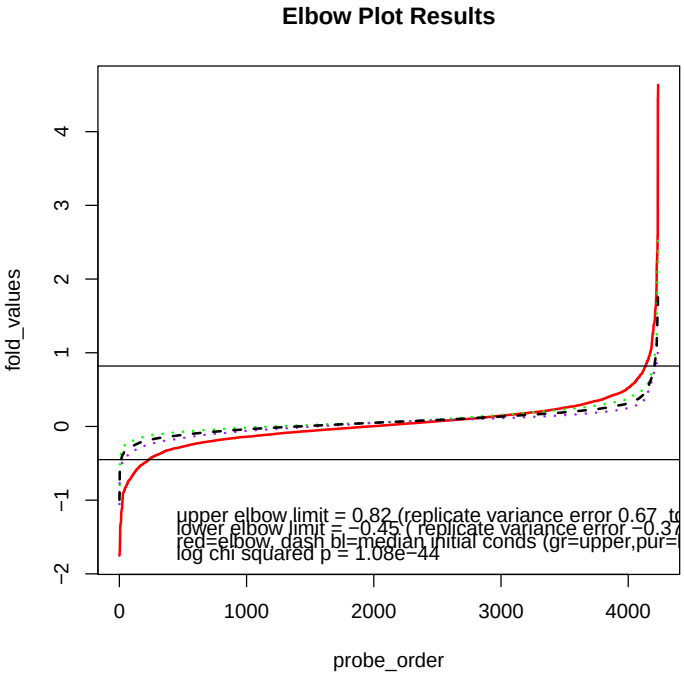
```
    ID_REF    fold
1 1001_115  0.2071
2  1002_33  0.0046
3 1003_942  0.0323
4 1004_552  0.3172
5 1005_657  0.0230
6 1006_393 -0.0754
[1] "firsta_data"
    ID_REF    fold
1 1001_115  0.2071
2  1002_33  0.0046
3 1003_942  0.0323
4 1004_552  0.3172
5 1005_657  0.0230
6 1006_393 -0.0754
[1] "sorted"
[1] "headers"
[1] "rowsums"
[1] "fold"
[1]  0.2611  0.0195  0.0503  0.3531 -0.0252 -0.0441
[1] "bound data"
    ID_REF    fold
1 1001_115  0.2611
2  1002_33  0.0195
3 1003_942  0.0503
4 1004_552  0.3531
5 1005_657 -0.0252
6 1006_393 -0.0441
[1] "firsta_data"
    ID_REF    fold
1 1001_115  0.2611
2  1002_33  0.0195
3 1003_942  0.0503
4 1004_552  0.3531
5 1005_657 -0.0252
6 1006_393 -0.0441
[1] "sorted"
[1] "headers"
[1] "rowsums"
[1] "fold"
[1]  0.0346 -0.2222  0.5502  0.0479 -0.0125  0.0560
[1] "bound data"
    ID_REF    fold
1 1001_115  0.0346
2  1002_33 -0.2222
3 1003_942  0.5502
```

```
4 1004_552  0.0479
5 1005_657 -0.0125
6 1006_393  0.0560
[1] "firsta_data"
    ID_REF    fold
1 1001_115  0.0346
2  1002_33 -0.2222
3 1003_942  0.5502
4 1004_552  0.0479
5 1005_657 -0.0125
6 1006_393  0.0560
[1] "sorted"
[1] "headers"
[1] "rowsums"
[1] "fold"
[1]  0.0279 -0.0132  0.4221  0.2437  0.1447  0.0283
[1] "bound data"
    ID_REF    fold
1 1001_115  0.0279
2  1002_33 -0.0132
3 1003_942  0.4221
4 1004_552  0.2437
5 1005_657  0.1447
6 1006_393  0.0283
[1] "firsta_data"
    ID_REF    fold
1 1001_115  0.0279
2  1002_33 -0.0132
3 1003_942  0.4221
4 1004_552  0.2437
5 1005_657  0.1447
6 1006_393  0.0283
[1] "sorted"
[1] "headers"
[1] "rowsums"
[1] "fold"
[1] 0.0819 0.0017 0.4401 0.2796 0.0965 0.0596
[1] "bound data"
    ID_REF   fold
1 1001_115 0.0819
2  1002_33 0.0017
3 1003_942 0.4401
4 1004_552 0.2796
5 1005_657 0.0965
6 1006_393 0.0596
[1] "firsta_data"
```

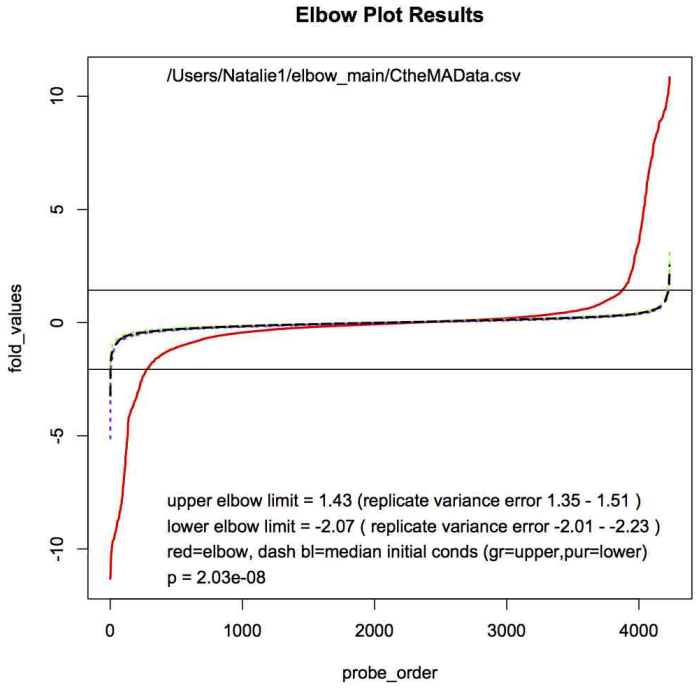
```
    ID_REF   fold
1 1001_115 0.0819
2  1002_33 0.0017
3 1003_942 0.4401
4 1004_552 0.2796
5 1005_657 0.0965
6 1006_393 0.0596
[1] "sorted"
[1] "headers"
[1] "rowsums"
[1] "fold"
[1] -0.0828 -0.2743  0.5480 -0.0174 -0.0913 -0.0621
[1] "bound data"
    ID_REF    fold
1 1001_115 -0.0828
2  1002_33 -0.2743
3 1003_942  0.5480
4 1004_552 -0.0174
5 1005_657 -0.0913
6 1006_393 -0.0621
[1] "firsta_data"
    ID_REF    fold
1 1001_115 -0.0828
2  1002_33 -0.2743
3 1003_942  0.5480
4 1004_552 -0.0174
5 1005_657 -0.0913
6 1006_393 -0.0621
[1] "sorted"
[1] "headers"
[1] "rowsums"
[1] "fold"
[1] -0.0895 -0.0653  0.4199  0.1784  0.0659 -0.0898
[1] "bound data"
    ID_REF    fold
1 1001_115 -0.0895
2  1002_33 -0.0653
3 1003_942  0.4199
4 1004_552  0.1784
5 1005_657  0.0659
6 1006_393 -0.0898
[1] "firsta_data"
    ID_REF    fold
1 1001_115 -0.0895
2  1002_33 -0.0653
3 1003_942  0.4199
```

```
4 1004_552  0.1784
5 1005_657  0.0659
6 1006_393 -0.0898
[1] "sorted"
[1] "headers"
[1] "rowsums"
[1] "fold"
[1] -0.0355 -0.0504  0.4379  0.2143  0.0177 -0.0585
[1] "bound data"
    ID_REF    fold
1 1001_115 -0.0355
2  1002_33 -0.0504
3 1003_942  0.4379
4 1004_552  0.2143
5 1005_657  0.0177
6 1006_393 -0.0585
[1] "firsta_data"
    ID_REF    fold
1 1001_115 -0.0355
2  1002_33 -0.0504
3 1003_942  0.4379
4 1004_552  0.2143
5 1005_657  0.0177
6 1006_393 -0.0585
[1] "sorted"
[1] "headers"
[1] "upper elbow limit = 0.82 (replicate variance error 0.67  to  1.05 )"
[1] "lower elbow limit = -0.45 ( replicate variance error -0.37  to  -0.63 )"
[1] "log chi squared p = 1.08e-44"

>
>     # uncomment to write the significant probes to 'signprobes.csv'
>     #write.table(sig,file="signprobes.csv",sep=",",row.names=FALSE)
```



Your plot will appear. You can also ask R to report the results. a csv file with the significant probes and their fold values. After you enter the command below and it will appear in your R folder.



6 Limma Pipeline

The steps below, for setting up the Microarray data and loading it into R, were adapted from the tutorial: `http://bioinformatics.knowledgeblog.org/2011/06/20/analysing-microarray-data-in-bioconductor`

First, load your Microarray data into R (in the below example, we will load an expression set from NCBI's GEO database). Note, the data must be normalized (see the last line of the code block below).

```
>         ####
>         # Uncomment the lines below to install the necessary libraries:
>         ####
>         #source("http://www.bioconductor.org/biocLite.R")
>         #biocLite("GEOquery")
>         #biocLite("simpleaffy")
>         #biocLite("limma")
>         #biocLite("affyPLM")
>         #biocLite("RColorBrewer")
>         ####
>
>         # retrieve the data from GEO
```

```
>         library("GEOquery")
>         getGEOSuppFiles("GSE20986")
>         untar("GSE20986/GSE20986_RAW.tar", exdir="data")
>         cels <- list.files("data/", pattern = "[gz]")
>         sapply(paste("data", cels, sep="/"), gunzip)

data/GSM524662.CEL.gz data/GSM524663.CEL.gz data/GSM524664.CEL.gz
             13555726              13555055              13555639
data/GSM524665.CEL.gz data/GSM524666.CEL.gz data/GSM524667.CEL.gz
             13560122              13555663              13557614
data/GSM524668.CEL.gz data/GSM524669.CEL.gz data/GSM524670.CEL.gz
             13556090              13560054              13555971
data/GSM524671.CEL.gz data/GSM524672.CEL.gz data/GSM524673.CEL.gz
             13554926              13555042              13555290

>         # reading in the CEL data into R (w/simpleaffy)
>         library("simpleaffy")
>         pheno_data <- data.frame(
+                 c(    # Name
+                         "GSM524662.CEL", "GSM524663.CEL", "GSM524664.CEL",
+                         "GSM524665.CEL", "GSM524666.CEL", "GSM524667.CEL",
+                         "GSM524668.CEL", "GSM524669.CEL", "GSM524670.CEL",
+                         "GSM524671.CEL", "GSM524672.CEL", "GSM524673.CEL"),
+                 c(    # FileName
+                         "GSM524662.CEL", "GSM524663.CEL", "GSM524664.CEL",
+                         "GSM524665.CEL", "GSM524666.CEL", "GSM524667.CEL",
+                         "GSM524668.CEL", "GSM524669.CEL", "GSM524670.CEL",
+                         "GSM524671.CEL", "GSM524672.CEL", "GSM524673.CEL"),
+                 c(    # Target
+                         "iris", "retina", "retina",
+                         "iris", "retina", "iris",
+                         "choroid", "choroid", "choroid",
+                         "huvec", "huvec", "huvec")
+                 )
>         colnames(pheno_data) <- c("Name", "FileName", "Target")
>         write.table(pheno_data, "data/phenodata.txt", row.names=FALSE, quote=FALSE)
>         celfiles <- read.affy(covdesc="phenodata.txt", path="data")
>         # normalizing the data
>         celfiles.gcrma <- gcrma(celfiles)

Adjusting for optical effect.....Done.
Computing affinities[1] "Checking to see if your internet connection works..."
.Done.
Adjusting for non-specific binding.....Done.
Normalizing
Calculating Expression
```

Perform quality control checks:

- Chip level QC:

```
> ########
> # quality control checks (chip-level)
> ########
> # load colour libraries
> library("RColorBrewer")
> # set colour palette
> cols <- brewer.pal(8, "Set1")
> # plot a boxplot of unnormalised intensity values
> boxplot(celfiles, col=cols)
> # plot a boxplot of normalised intensity values, affyPLM is required to inter
> library("affyPLM")
> boxplot(celfiles.gcrma, col=cols)
> # the boxplots are somewhat skewed by the normalisation algorithm
> # and it is often more informative to look at density plots
> # Plot a density vs log intensity histogram for the unnormalised data
> hist(celfiles, col=cols)
> # Plot a density vs log intensity histogram for the normalised data
> hist(celfiles.gcrma, col=cols)
> ########
```

- Probe level QC:

```
>                 ########
>                 # quality control checks (probe-level)
>                 ########
>                 # Perform probe-level metric calculations on the CEL files:
>                 celfiles.qc <- fitPLM(celfiles)
>                 # Create an image of GSM24662.CEL:
>                 image(celfiles.qc, which=1, add.legend=TRUE)
>                 # Create an image of GSM524665.CEL
>                 # There is a spatial artifact present
>                 image(celfiles.qc, which=4, add.legend=TRUE)
>                 # affyPLM also provides more informative boxplots
>                 # RLE (Relative Log Expression) plots should have
>                 # values close to zero. GSM524665.CEL is an outlier
>                 RLE(celfiles.qc, main="RLE")
>                 # We can also use NUSE (Normalised Unscaled Standard Errors).
>                 # The median standard error should be 1 for most genes.
>                 # GSM524665.CEL appears to be an outlier on this plot too
>                 NUSE(celfiles.qc, main="NUSE")#'
>                 ########
```

- Sample vs. sample QC:

```
>         ########
>         # quality control checks (sample vs. sample)
>         ########
>         eset <- exprs(celfiles.gcrma)
>         distance <- dist(t(eset),method="maximum")
>         clusters <- hclust(distance)
>         plot(clusters)
>         ########
```

Next, filter the data.

```
>         ########
>         # data filtering
>         ########
>         celfiles.filtered <- nsFilter(celfiles.gcrma, require.entrez=FALSE, remove.du
>         # What got removed and why?
>         celfiles.filtered$filter.log

$numLowVar
[1] 27307

$feature.exclude
[1] 62
```

Then, extract the differentially expressed probes.

```
>         ########
>         # Finding differentially expressed probesets
>         ########
>         samples <- celfiles.gcrma$Target
>         # check the results of this
>         #samples
>
>         # convert into factors
>         samples <- as.factor(samples)
>         # check factors have been assigned
>         #samples
>
>         # set up the experimental design
>         design <- model.matrix(~0 + samples)
>         colnames(design) <- c("choroid", "huvec", "iris", "retina")
>
>         # inspect the experiment design
>         #design
```

And, feed the data into limma:

```
>         ########
>         # feed the data into limma for analysis
>         ########
>         library("limma")
>         # fit the linear model to the filtered expression set
>         fit <- lmFit(exprs(celfiles.filtered$eset), design)
>         # set up a contrast matrix to compare tissues v cell line
>         contrast.matrix <- makeContrasts(huvec_choroid = huvec - choroid, huvec_retin
>         # Now the contrast matrix is combined with the per-probeset linear model fit.
>         huvec_fits <- contrasts.fit(fit, contrast.matrix)
>         huvec_ebFit <- eBayes(huvec_fits)
```

Finally, run the dataset through the ELBOW method.

```
>         # load the ELBOW library
>         library("ELBOW")
>         # find the ELBOW limits
>         get_elbow_limma(huvec_ebFit)
```

```
                           up_limit low_limit
huvec_choroid              1.586583 -1.797899
huvec_retina               1.576926 -1.746454
huvec_iris <- huvec - iris 1.664171 -1.821341
```

7 DESeq Pipeline

The steps below, for loading the RNA-seq data into R, were adapted from the tutorial: `http://cgrlucb.wikispaces.com/Spring+2012+DESeq+Tutorial`

Load the RNA-seq bam file into R:

```
>         # load the ELBOW library
>         library("ELBOW")
>         # install the DESeq libraries
>         #source("http://www.bioconductor.org/biocLite.R")
>         #biocLite("DESeq")
>
>         ## download the table
>         library("DESeq")
>         # the following bam file dataset was obtained from:
>         # http://cgrlucb.wikispaces.com/file/view/yeast_sample_data.txt
>         # it has been downloaded into this package for speed convenience.
>         filename <- system.file("extdata", "yeast_sample_data.txt", package = "ELBOW"
>         count_table <- read.table(filename, header=TRUE, sep="\t", row.names=1)
>         expt_design <- data.frame(row.names = colnames(count_table), condition = c("W
>         conditions = expt_design$condition
```

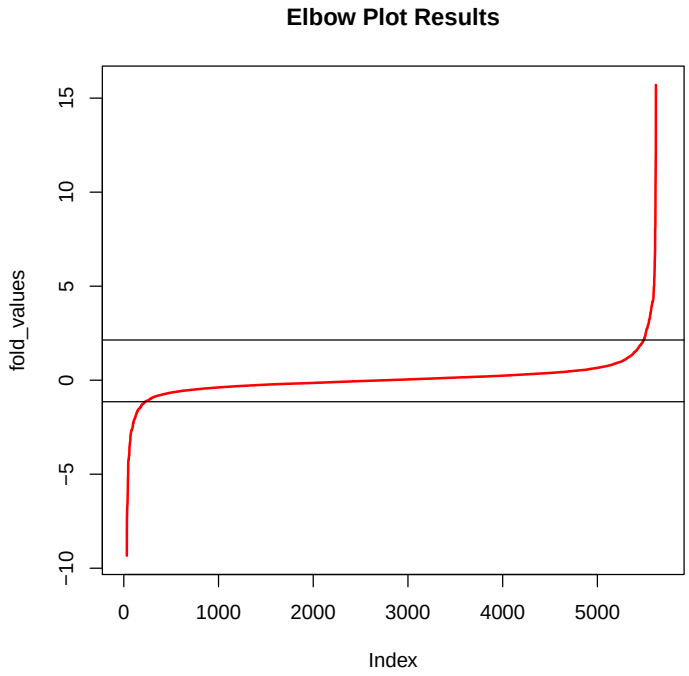
```
>         rnadata <- newCountDataSet(count_table, conditions)
>         rnadata <- estimateSizeFactors(rnadata)
>         rnadata <- as(rnadata, "CountDataSet")
>         ## rnadata <- estimateVarianceFunctions(rnadata)
>         rnadata <- estimateDispersions(rnadata)
>         # this next step is essential, but it takes a long time...
>         results <- nbinomTest(rnadata, "M", "WE")
```

Then, run an elbow analysis on the data:

```
>         limits <- do_elbow_rnaseq(results)
```

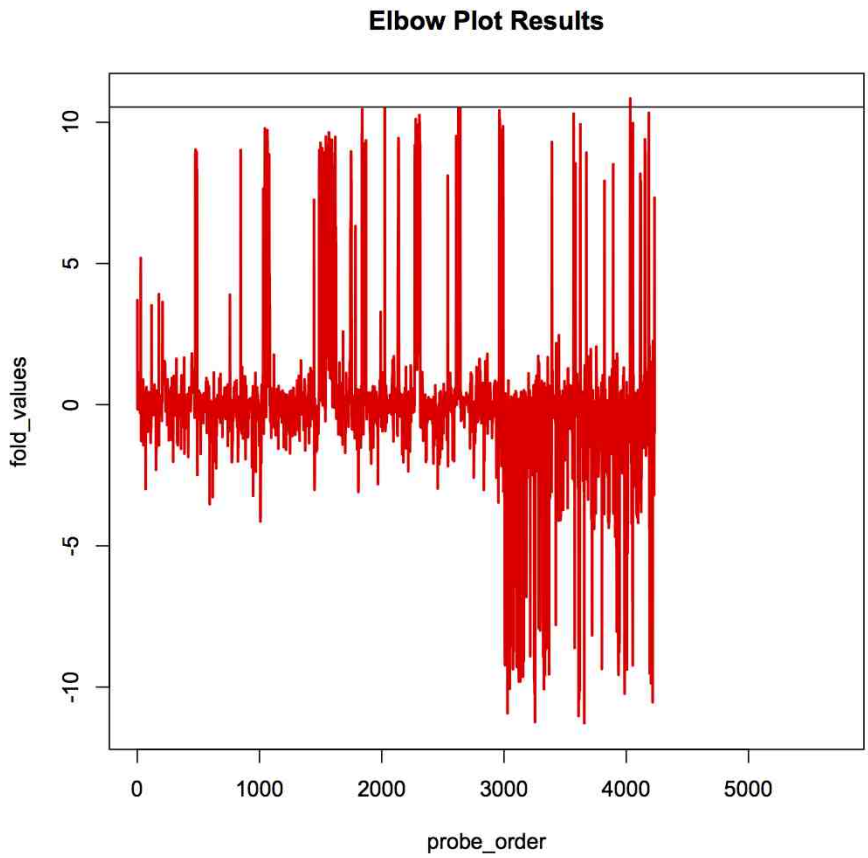
Additionally, you can plot the data:

```
>         # plot the results w/elbow
>         plot_dataset(results, "log2FoldChange", limits$up_limit, limits$low_limit)
```



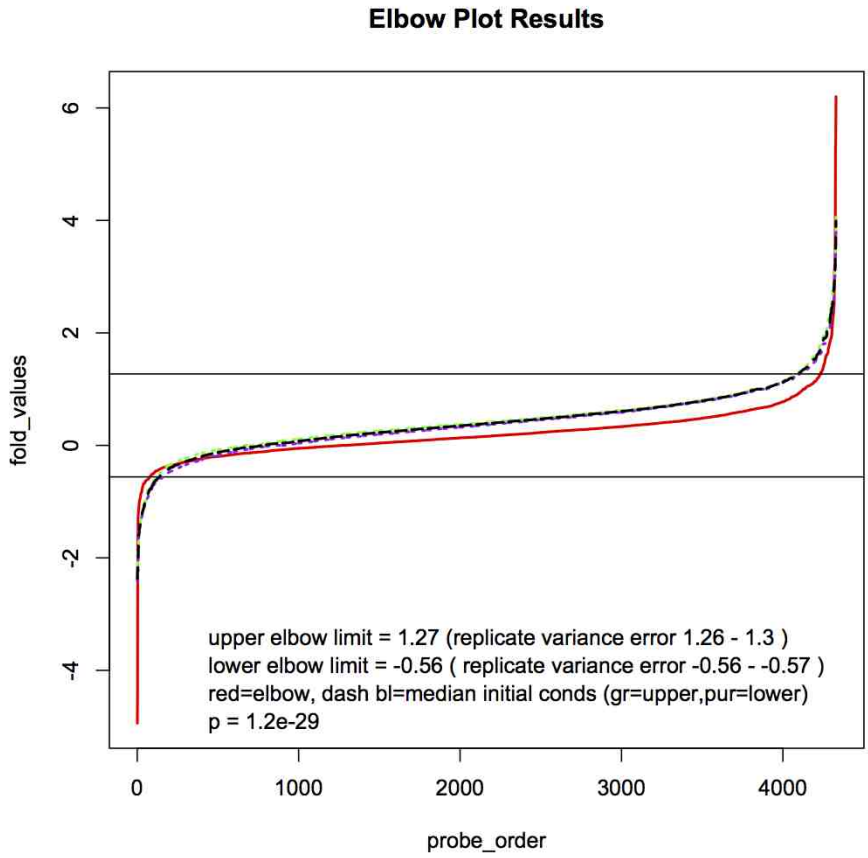
8 Troubleshooting

8.1 "I don't see anything like an ELBOW!"



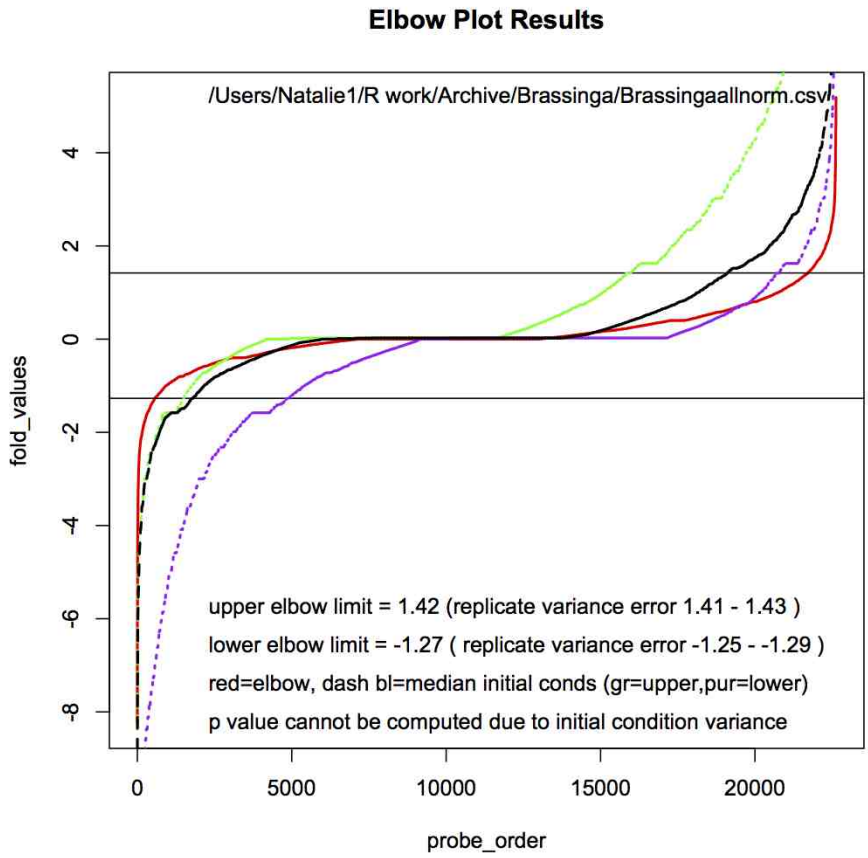
This kind of result is typical when data has not been properly formatted. Check that you have probes in the first column, initial conditions in the middle set of columns and final conditions in the last set of columns. Also be certain your file has been saved in proper comma separated values (.csv) and you have entered the correct number of replicates when you ran the program.

8.2 The final cuve is flatter than the curve of initial conditions



This result simply means there is more variance in the initial conditions than the final conditions. However, it is within limits for the ELBOW test. You may see this in certain situations such as downward metabolism or reduced expression overall in the final conditions. Because ELBOW has provided a p value, the limits are still valid to determine biological significance.

8.3 "The ELBOW curve is flatter that the curve of initial conditions and there is no p value."

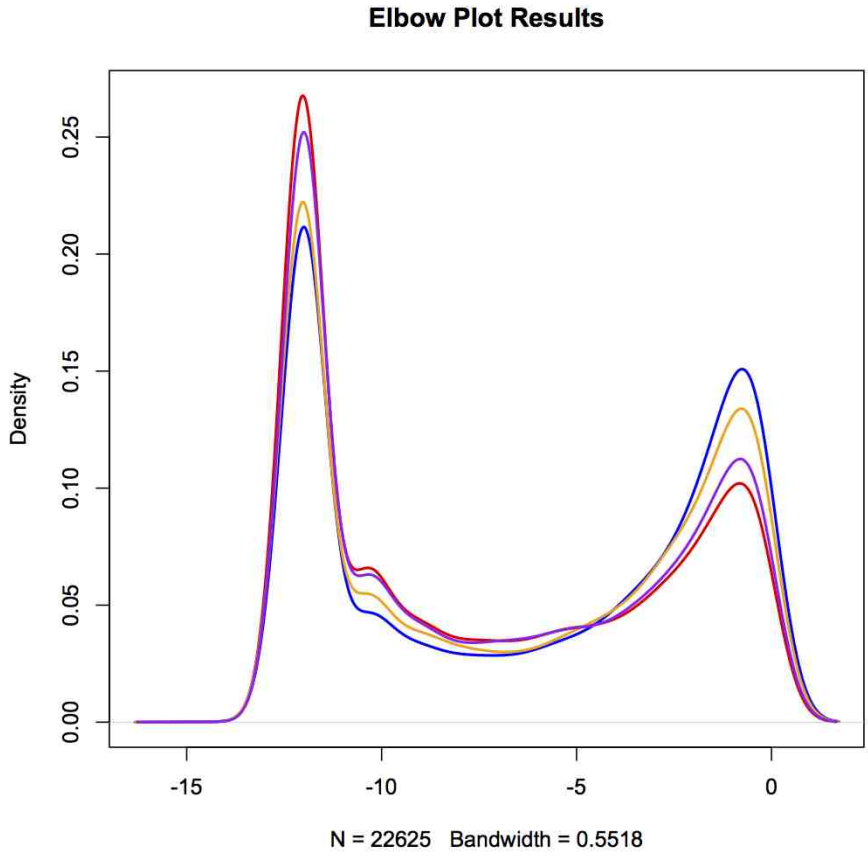


This result means there is high replicate variance in the initial conditions. It also means that the final replicate set does not meet the model conditions. The wide variance in upper and lower null lines also indicates a problem. The overall effect is that because of the variance in the initial replicates, ELBOW will not work and the results are not useful. It is often possible to troubleshoot this type of dataset to find the cause.

Try using R's density plot to plot the density of each replicate.

8.4 Density plot

Try using R's density plot to plot the density of each replicate.



Using "head" will yield the column names of the dataset:

```
> head(csv_data)
```

Add each additional replicate using:

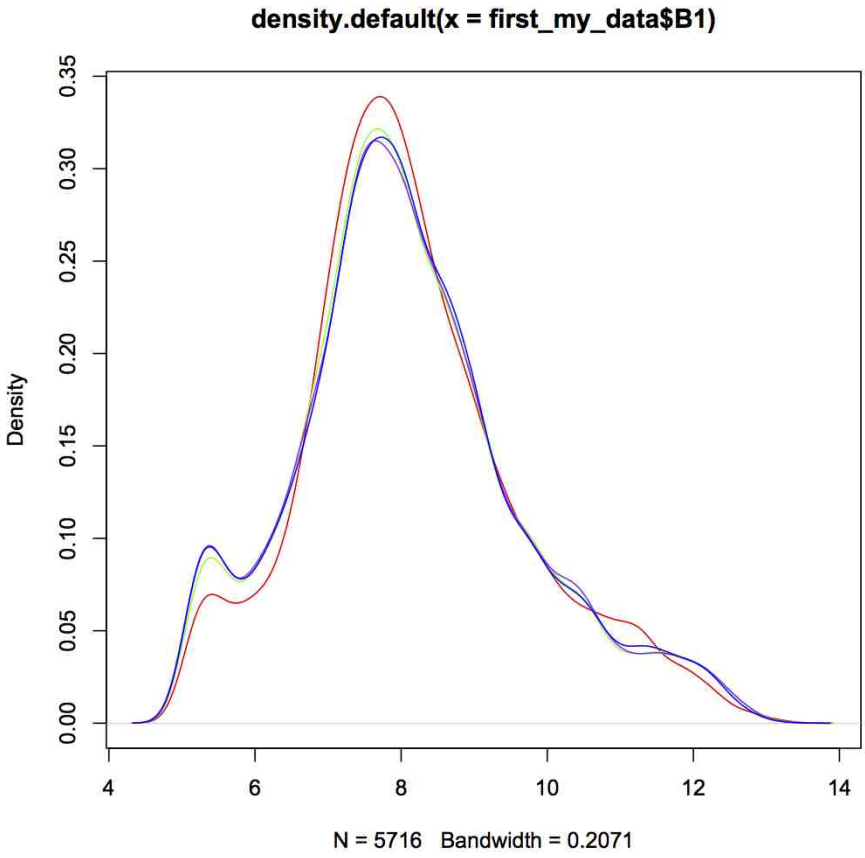
```
> plot(density(csv_data$y1))
```

(where y1 is the name of your first column)

Add the rest of the replicates on the same plot using

```
> lines(density(csv_data$y2))
> lines(density(csv_data$y3))
```

Each line should approximately overlap. In this case, the replicates do not overlap but are the same shape. It may be possible to adjust data until the lines overlap and then apply ELBOW.



In this case, the density plot indicates that one replicate (red) is very different from the others. The others match well and overlap each other.

ELBOW may work after the one very different replicate is removed from the dataset.

If you are unable to get your datasets to generate a p value you cannot use ELBOW and should consult with your data provider.