Introduction to analyzing NanoString nCounter data using the NanoStringNorm package

Daryl Waggott

December 11, 2017

Contents

1 Getting started **2**

2 Importing nCounter Data **2**

3 Standard Normalization Methods **3**

4 Other Normalization Methods **5**

5 Static Plotting **7**

6 Interactive Plotting **19**

7 Best Normalization Method **22**

8 Study Design Related Issues **24**

1 Getting started

NanoStringNorm is a suite of tools used to pre-process, run diagnostics, and visualize NanoString nCounter expression data. nCounter data has some unique features as compared to traditional intensity based arrays. Specifically, it uses direct digital detection requiring minimal sample intervention. The 'direct' refers to the process of counting individual tagged nucleic acids without any need for amplification and the 'digital' nature refers to the capacity to use absolute and specific quantification, independent of relative measures like intensity or amplification cycles. As such, the platform is useful for an array of study designs (multiplexed samples, joint miRNA - mRNA code sets) and sample types (FFPE, plasma, whole lysate).

The consequence of this flexibility is that not all pre-processing methods are valid with all tissues and code sets. For example, housekeeping genes are a fundamental part of mRNA standardization, however their appropriateness is not clear for miRNA data. NanoStringNorm emphasizes normalization diagnostics and visualization of results in order to provide the best data, void of technical artifacts for downstream analysis. Moreover, the package is designed to be fully extensible in order to support and evaluate new pre-processing methods.

The following vignette describes the general workflow for using NanoStringNorm. In the simplest case, one can run the NanoStringNorm package on the raw counts without specifying any normalization methods. The output will be of class *NanoStringNorm* and can be subsequently plotted for summary review and sample diagnostics.

```
> require('NanoStringNorm');
> data("NanoString");

> # example 1
> NanoString.mRNA.raw = NanoStringNorm(NanoString.mRNA);
> pdf('my_first_nsn_plot.pdf');
> Plot.NanoStringNorm(NanoString.mRNA.raw, plot.type = 'all');
> dev.off();
```

2 Importing nCounter Data

The input data usually comes in the form of a structured Excel spreadsheet. You can export the raw count data from Excel as a delimited text file for use with R. Start by opening the *raw* worksheet in a blank Excel page for editing. Copy the count data (row 23) for each sample including the first 3 annotation columns (Code.Class, Name and Accession) to a separate worksheet or text file. Don't forget to add the sample IDs (row 5), and remove any incomplete rows or columns. The resulting tabular data can be saved as a tab delimited file for import into R.

Alternatively, you can import data directly from xls format into R using the function `read.xls.RCC` based on core functionality in the *gdata* package.

```
> # directly import the nCounter output
> path.to.xls.file <- system.file("extdata", "RCC_files", "RCCCollector1_rat_tcdd.xls",
+                                 package = "NanoStringNorm");
> NanoString.mRNA <- read.xls.RCC(x = path.to.xls.file, sheet = 1);

You have chosen to import worksheet 1 named RCC Collection. Does that sound correct?
The other sheet names are:
1:RCC Collection
2:Reporter Order
3:__VBA__0

There were 25 samples imported.
Note that spaces in sample names will be replaced by dots.
The first and last 3 sample names found in the dataset are:
X20100728_Watson1-1_HW1D-a_01 X20100728_Watson1-1_HW2B-b_02 X20100728_Watson1-1_HW3B-a_03 X20100728_Watson1

There were 68 genes imported with the following Code Class breakdown:
```

```
Endogenous   Negative   Positive
        54          8          6
```

```
> # only keep the counts and not the header
> NanoString.mRNA <- NanoString.mRNA$x;
```

3 Standard Normalization Methods

The current normalization procedure recommended by NanoString involves a set of simple aggregations and adjustments using control genes. There are many potential combinations of these parameters which presents a challenge to end users. In preliminary work no single combination has proven to work best in all situations or study designs. Therefore, a reasonable approach is to start with one of the better performing combinations and calibrate it if necessary. The *tweaking* is based on reviewing the diagnostics and paying close attention to the experimental design caveats discussed at the end of this document.

The following example is based on a subset of data from an mRNA study observing the response of different Rat strains to dioxins. The first step is to add some housekeeping genes. In some code sets, specifically miRNAs, these are already included.

```
> # specify housekeeping genes in annotation
> data(NanoString);
> NanoString.mRNA[NanoString.mRNA$Name %in%
+         c('Eef1a1','Gapdh','Hprt1','Ppia','Sdha'),'Code.Class'] <- 'Housekeeping';
```

Next, run a typical normalization. The following is reasonable and in many situations will be sufficient to handle technical variation. Note the use of the 'mean' option for Background correction. This is the least conservative method of summarizing negative controls and will increase the number of observed false positives due to elevated uncertainty at lower expression values.

By default the output is a list of sample and gene summary data.frames. In the following example only the matrix of normalized values is output.

```
> # example 2
> # normalize mRNA and output a matrix of normalized counts
> NanoString.mRNA.norm <- NanoStringNorm(
+         x = NanoString.mRNA,
+         anno = NA,
+         CodeCount = 'sum',
+         Background = 'mean',
+         SampleContent = 'housekeeping.sum',
+         round.values = FALSE,
+         take.log = FALSE,
+         return.matrix.of.endogenous.probes = TRUE
+         );

##############################
### NanoStringNorm v1.2.1 ###
##############################

There are 25 samples and 49 Endogenous genes

Background: The following samples have an estimated background greater than
         3 standard deviations from the mean.

         background.zscore
LE4D51.a              3.71

Background: After correction 25 samples and 49
```

```
        Endogenous genes have less than 90% missing.

SampleContent: The following samples have sample/rna content greater than
        3 standard deviations from the mean.

         rna.zscore
LE4D51.a       3.36
```

NanoStringNorm prints informative diagnostic messages to the screen. Specifically, samples that have normalization factors three standard deviations from the mean are flagged for review. These large values could reflect a technical problem and dramatically influence all normalization.

In the following example the geometric mean is used to summarize the CodeCount (positive) and SampleContent (housekeeping) controls. This minimizes the impact of outlier values. Also, a stringent background correction is applied (mean + 2 standard deviations) which removes a large proportion of false positives and therefore increases specificity at the expense of some sensitivity. For preliminary analysis it can be easier to focus on high confidence results first. For these reasons, **the following model is recommended** as a first step. Adjustments, can be made based on review of diagnostics.

Background correction normalizes data on the assumption that negative control probe summaries should be transformed to 0. Similarly, CodeCount and SampleContent corrections normalize on the assumption that positive and housekeeping control probe summaries should each be equalized to some target value across all samples. These target values vary by experimental design; by default, they are calculated as average summary values across all samples. One implication of this default behaviour is that the CodeCount and SampleContent normalized data of a single sample are dependent on other samples being normalized. An alternative option is to specify expected target summary values using the 'CodeCount.summary.target' and 'SampleContent.summary.target' parameters, which allows samples to be normalized independently.

For miRNA data the use of mRNA or RNU's can be problematic, so change the SampleContent method to 'top.geo.mean'.

```
> # example 3
> # this is the recommended method!
> NanoString.mRNA.norm <- NanoStringNorm(
+         x = NanoString.mRNA,
+         CodeCount = 'geo.mean',
+         Background = 'mean.2sd',
+         SampleContent = 'housekeeping.geo.mean',
+         round.values = TRUE,
+         take.log = TRUE
+         );

##############################
### NanoStringNorm v1.2.1 ###
##############################

There are 25 samples and 49 Endogenous genes

Background: After correction 25 samples and 49
        Endogenous genes have less than 90% missing.

SampleContent: The following samples have sample/rna content greater than
        3 standard deviations from the mean.

         rna.zscore
LE4D51.a       3.13

log: Setting values less than 1 to 1 in order to calculate the log in positive space.
```

In order to provide more information for diagnostic and preliminary results one can supply binary trait data. Ideally, one would include a mix of design-related covariates such as RNA quality/quantity, FFPE age in addition to potential biological confounders such as tissue, age, gender, and ethnicity. Results will be presented as differential expression and correlations with normalization factors.

In this case we are looking at three *strains* of Rat. Data can be dichotomized using the median, extremes, kmeans etc. Values must be 1, 2 or NA. For categorical data with greater than two classes, you can convert them to a set of indicator variables i.e. plate1 vs. all other plates. This has the advantage of simple interpretation of effect estimates.

```
> # setup a binary trait using rat strain
> sample.names <- names(NanoString.mRNA)[-c(1:3)];
> strain1 <- rep(1, times = (ncol(NanoString.mRNA)-3));
> strain1[grepl('HW',sample.names)] <- 2;
> strain2 <- rep(1, times = (ncol(NanoString.mRNA)-3));
> strain2[grepl('WW',sample.names)] <- 2;
> strain3 <- rep(1, times = (ncol(NanoString.mRNA)-3));
> strain3[grepl('LE',sample.names)] <- 2;
> trait.strain <- data.frame(
+         row.names = sample.names,
+         strain1 = strain1,
+         strain2 = strain2,
+         strain3 = strain3
+         );

> # You can also input the gene annotation separately to allow flexibility
> NanoString.mRNA.anno <- NanoString.mRNA[,c(1:3)];
> NanoString.mRNA.data <- NanoString.mRNA[,-c(1:3)];
> #NanoString.mRNA.anno$cool.genes <- vector.of.cool.genes;

> # example 3
> # include a trait for differential expression and batch effect evaluation
> NanoString.mRNA.norm <- NanoStringNorm(
+         x = NanoString.mRNA.data,
+         anno = NanoString.mRNA.anno,
+         CodeCount = 'geo.mean',
+         Background = 'mean.2sd',
+         SampleContent = 'top.geo.mean',
+         round.values = TRUE,
+         take.log = TRUE,
+         traits = trait.strain
+         );

##############################
### NanoStringNorm v1.2.1 ###
##############################

There are 25 samples and 49 Endogenous genes

Background: After correction 25 samples and 49
        Endogenous genes have less than 90% missing.

log: Setting values less than 1 to 1 in order to calculate the log in positive space.
```

4 Other Normalization Methods

As an alternative to the standard normalization methods proposed by NanoString there is facility to apply an expanding set of additional methods. Some basic strategies include transforming each sample to zscores, the standard

normal distribution based on ranks, and an empirical derived normal distribution. The first two methods have advantages when comparing results across batches or platforms in the case of joint or meta-analysis. Quantile normalization on the other hand has the advantage of scaling samples within the same pre-processing batch to a common empirically derived distribution.

Variance Stabilizing Normalization is also implemented via functionality in the popular *vsn* package (available for installation via Bioconductor).

Summary diagnostics and plotting are still available for these methods.
Several examples include:

```
> # z-value transformation.
> # scale each sample to have a mean 0 and sd
> # by default all the other normalization methods are 'none'
> # you cannot apply a log because there are negative values
> # good for meta-analysis and cross platform comparison abstraction of effect size
> NanoString.mRNA.norm <- NanoStringNorm(
+         x = NanoString.mRNA,
+         OtherNorm = 'zscore',
+         return.matrix.of.endogenous.probes = TRUE
+         );

> # inverse normal transformation.
> # use quantiles to transform each sample to the normal distribution
> NanoString.mRNA.norm <- NanoStringNorm(
+         x = NanoString.mRNA,
+         OtherNorm = 'rank.normal',
+         return.matrix.of.endogenous.probes = TRUE
+         );

> # quantile normalization.
> # create an empirical distribution based on the median gene counts at the same
> # rank across sample. then transform each sample to the empirical distribution.
> NanoString.mRNA.norm <- NanoStringNorm(
+         x = NanoString.mRNA,
+         OtherNorm = 'quantile',
+         return.matrix.of.endogenous.probes = FALSE
+         );

> # vsn.
> # apply a variance stabilizing normalization.
> # fit and predict the model using 'all' genes i.e. 'controls' and
> # 'endogenous' (this is the default)
> # note this is just a wrapper for the vsn package
> # you could even add strata for the controls vs. the endogenous to review
> # systematic differences
> NanoString.mRNA.norm <- NanoStringNorm(
+         x = NanoString.mRNA,
+         OtherNorm = 'vsn',
+         return.matrix.of.endogenous.probes = FALSE,
+         genes.to.fit = 'all',
+         genes.to.predict = 'all'
+         );

> # vsn.
> # this time generate the parameters (fit the model) on the 'controls'
> # and apply (predict) on the 'endogenous'
> # alternatively you may want to use the endogenous probes for both fitting and predicting
```

```
> NanoString.mRNA.norm <- NanoStringNorm(
+         x = NanoString.mRNA,
+         OtherNorm = 'vsn',
+         return.matrix.of.endogenous.probes = FALSE,
+         genes.to.fit = 'controls',
+         genes.to.predict = 'endogenous'
+         );

> # vsn.
> # apply standard NanoString normalization strategies as an alternative to
> # the vsn affine transformation.
> # this effectively applies the glog2 variance stabilizing transformation
> # to the adjusted counts
> NanoString.mRNA.norm <- NanoStringNorm(
+         x = NanoString.mRNA,
+         CodeCount = 'sum',
+         Background = 'mean',
+         SampleContent = 'top.geo.mean',
+         OtherNorm = 'vsn',
+         return.matrix.of.endogenous.probes = FALSE,
+         genes.to.fit = 'endogenous',
+         genes.to.predict = 'endogenous',
+         calib = 'none'
+         );
```

5 Static Plotting

There are a number of descriptive, diagnostic and results plots. The following code snippets show how to generate them.

```
null device
          1

> # plot all the plots as PDF report
> pdf('NanoStringNorm_Example_Plots_All.pdf');
> Plot.NanoStringNorm(
+         x = NanoString.mRNA.norm,
+         label.best.guess = TRUE,
+         plot.type = 'all'
+         );
> dev.off();

> # publication quality tiff volcano plot
> tiff('NanoStringNorm_Example_Plots_Volcano.tiff', units = 'in',  height = 6,
+         width = 6, compression = 'lzw', res = 1200, pointsize = 10);
> Plot.NanoStringNorm(
+         x = NanoString.mRNA.norm,
+         label.best.guess = TRUE,
+         plot.type = c('volcano'),
+         title = FALSE
+         );
> dev.off();

> # all plots as separate files output for a presentation
> png('NanoStringNorm_Example_Plots_%03d.png', units = 'in',  height = 6,
+         width = 6, res = 250, pointsize = 10);
> Plot.NanoStringNorm(
```

```
+         x = NanoString.mRNA.norm,
+         label.best.guess = TRUE,
+         plot.type = c('cv', 'mean.sd', 'RNA.estimates', 'volcano', 'missing',
+                 'norm.factors', 'positive.controls', 'batch.effects')
+         );
> dev.off();

> # user specified labelling with optimal resolution for most digital displays
> png('NanoStringNorm_Example_Plots_Normalization_Factors.png', units = 'in', height = 6,
+         width = 6, res = 250, pointsize = 10);
> Plot.NanoStringNorm(
+         x = NanoString.mRNA.norm,
+         label.best.guess = FALSE,
+         label.ids = list(genes = rownames(NanoString.mRNA.norm$gene.summary.stats.norm),
+                 samples = rownames(NanoString.mRNA.norm$sample.summary.stats)),
+         plot.type = c('norm.factors')
+         );
> dev.off();
```

What is the distribution of gene expression? Existing or potential Housekeeping genes should have modest to high expression with very low variation. Estimates of the standard deviation of genes can be used for power calculations.

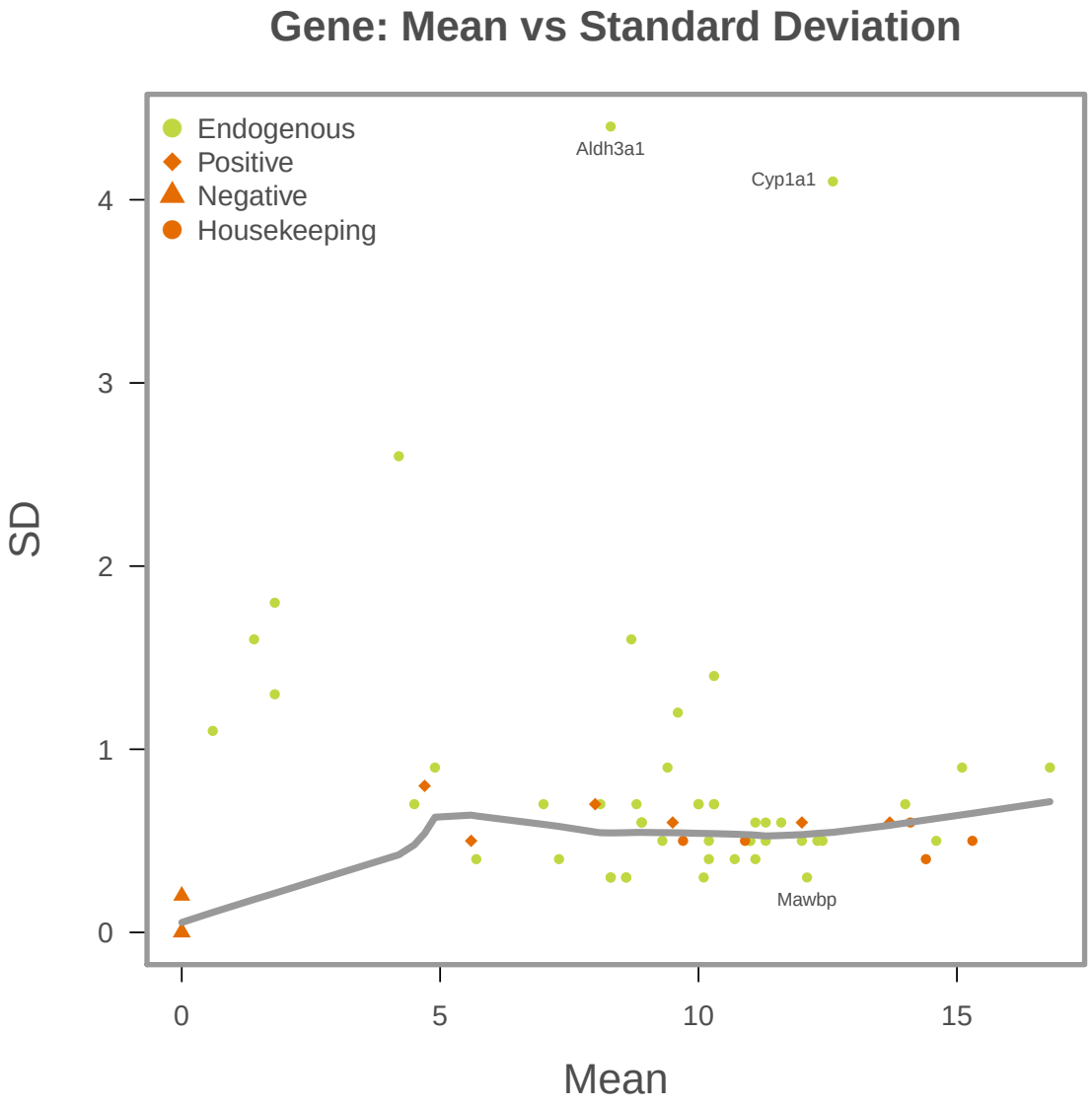
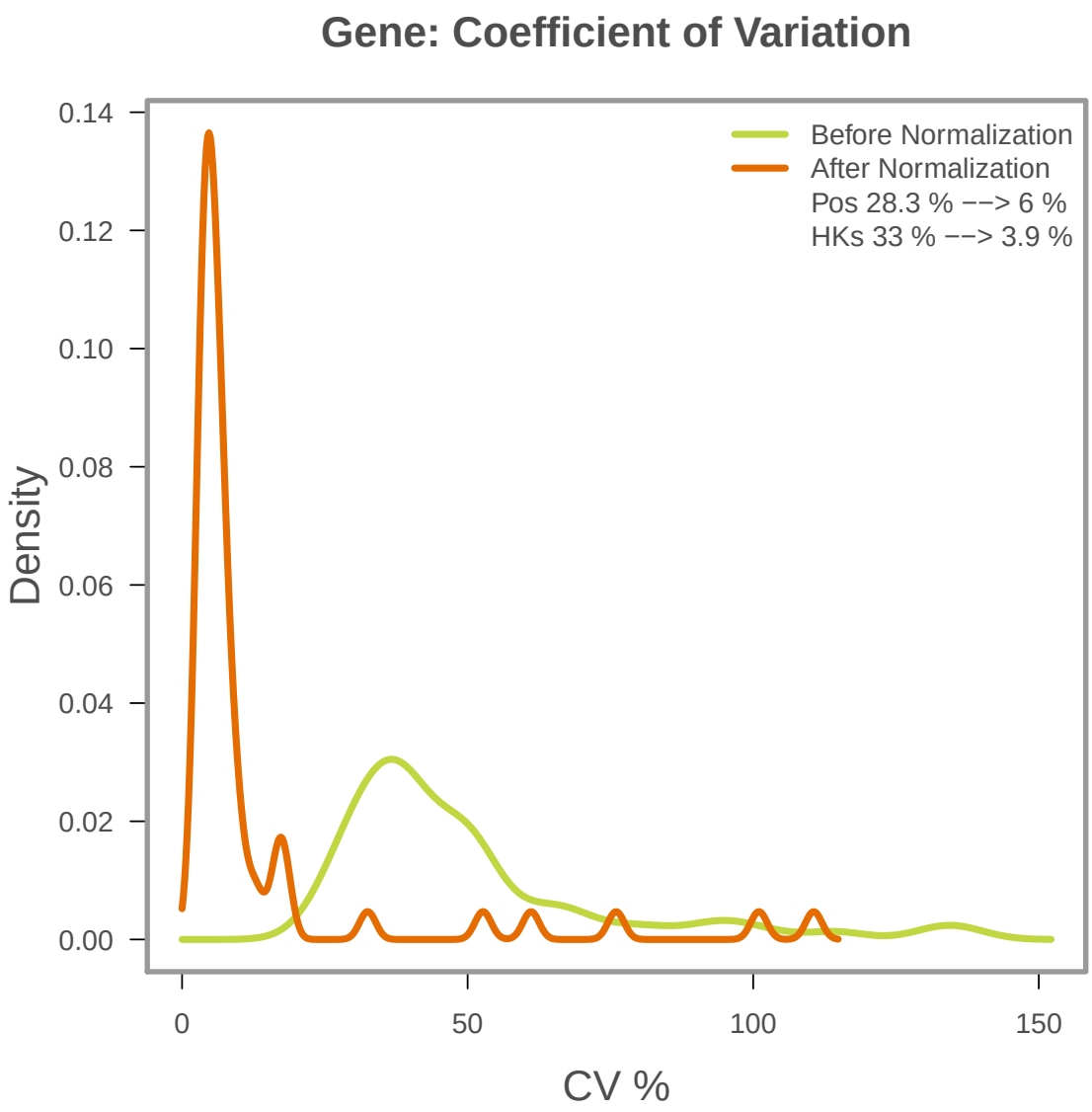


Figure 1: The mean vs the standard deviation. Green dots are endogenous genes and orange are controls, with the symbol indicating the specific type. The grey line is a best fit loess curve through the data. Generally, good housekeeping genes should have high mean and low sd i.e. bottom right of plot.

A comparison of global gene level variation pre- and post-normalization. The post-normalization curve should be shifted to the left if the method is properly accounting for technical variation.



Global gene level variation before and after normalization.

Figure 2: Global gene level variation before and after normalization.

Is there differential expression in design traits? Do you see known or expected signals among biologically relevant contrasts? Are there any profiles associated with biological covariates such as age, gender, trx, ethnicity, or bmi? Including these traits as covariates in a model could dampen any true positive signal if the same genes are associated with the outcome. Conversely, not including the traits could result in confounded results. Models should be evaluated carefully.

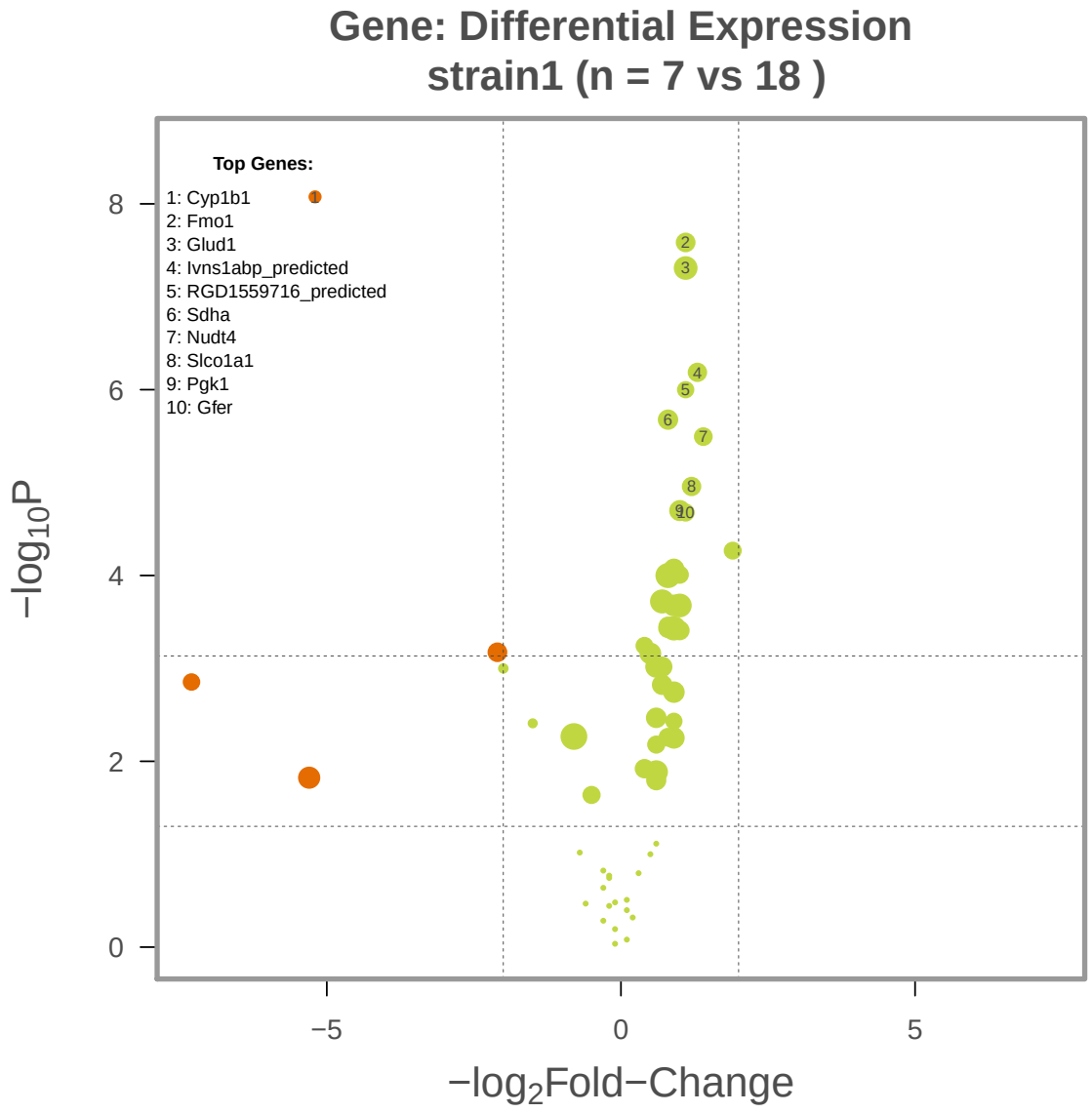


Figure 3: Volcano plot of Rat strain 1 vs. all other rats. The y-axis is -log10 p-values and the x-axis is fold change. The size of the points is related to mean expression and the orange colour indicates fold change less than negative two. The two dashed horizontal lines are drawn at p-value = 0.05 and a Bonferroni corrected level.

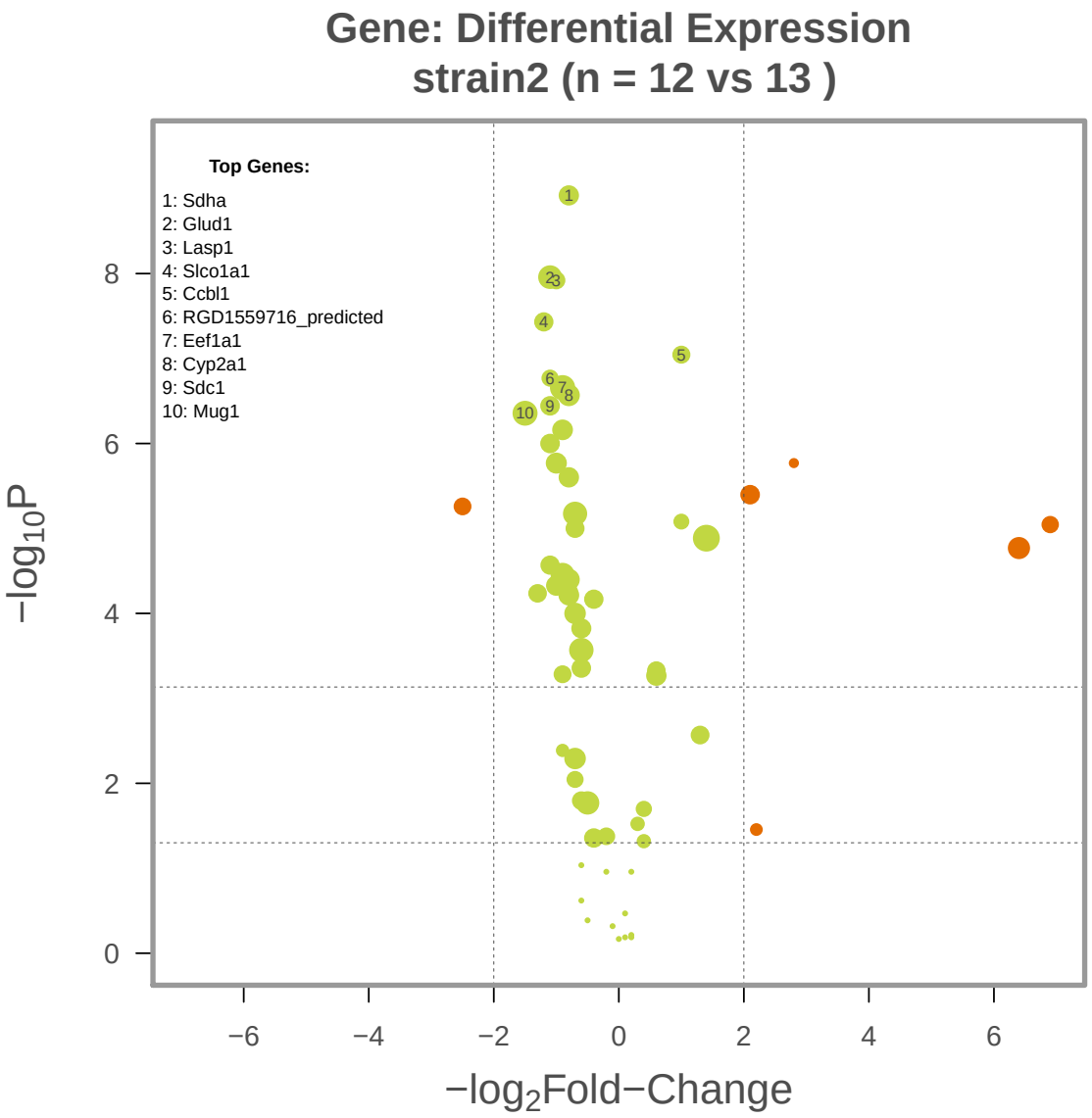


Figure 4: Volcano plot of Rat strain 2 vs. all other rats. The y-axis is -log10 p-values and the x-axis is fold change. The size of the points is related to mean expression and the orange colour indicates fold change greater than two. The two dashed horizontal lines are drawn at p-value = 0.05 and a Bonferroni corrected level.

Are there any samples with many missing values? Missing values could be caused by a technical failure or as a result of too little input material.

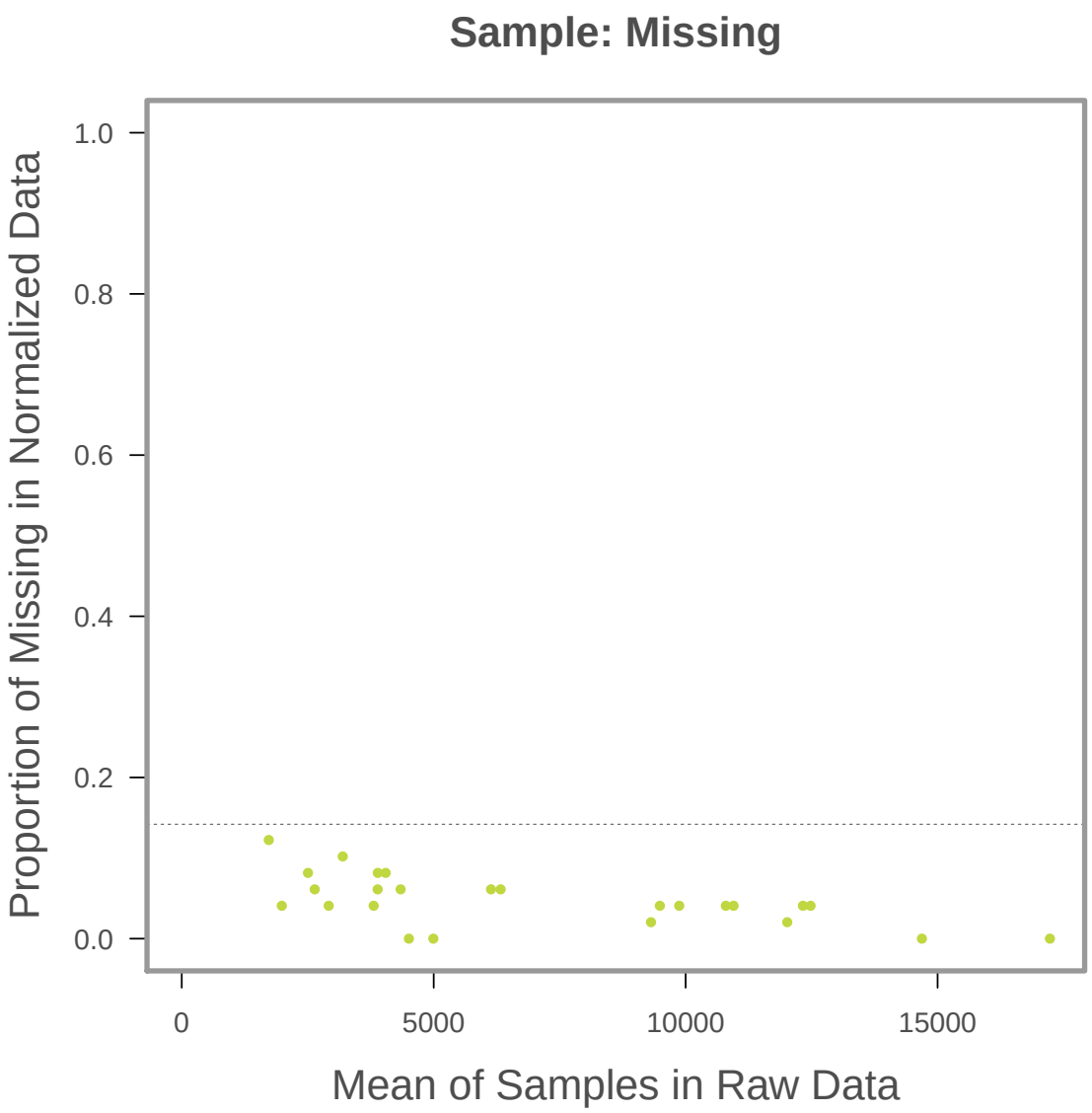


Figure 5: Proportion of missing values vs. mean expression by sample.
Proportion of missing values vs. mean expression by sample.

In normalizing for sample content you can use Housekeeping genes or an aggregate score of a larger set of non-Housekeeping genes. Under ideal conditions, both methods should be equivalent. Samples with large deviations from the best fit line should be investigated. When looking at miRNA code sets a large difference could reflect contamination by ligation inhibitors. All the miRNA genes undergo ligation, however the mRNA housekeeping genes do not. If inhibitors are present, this could also affect levels of the other controls and consequently magnify inter-sample differences.

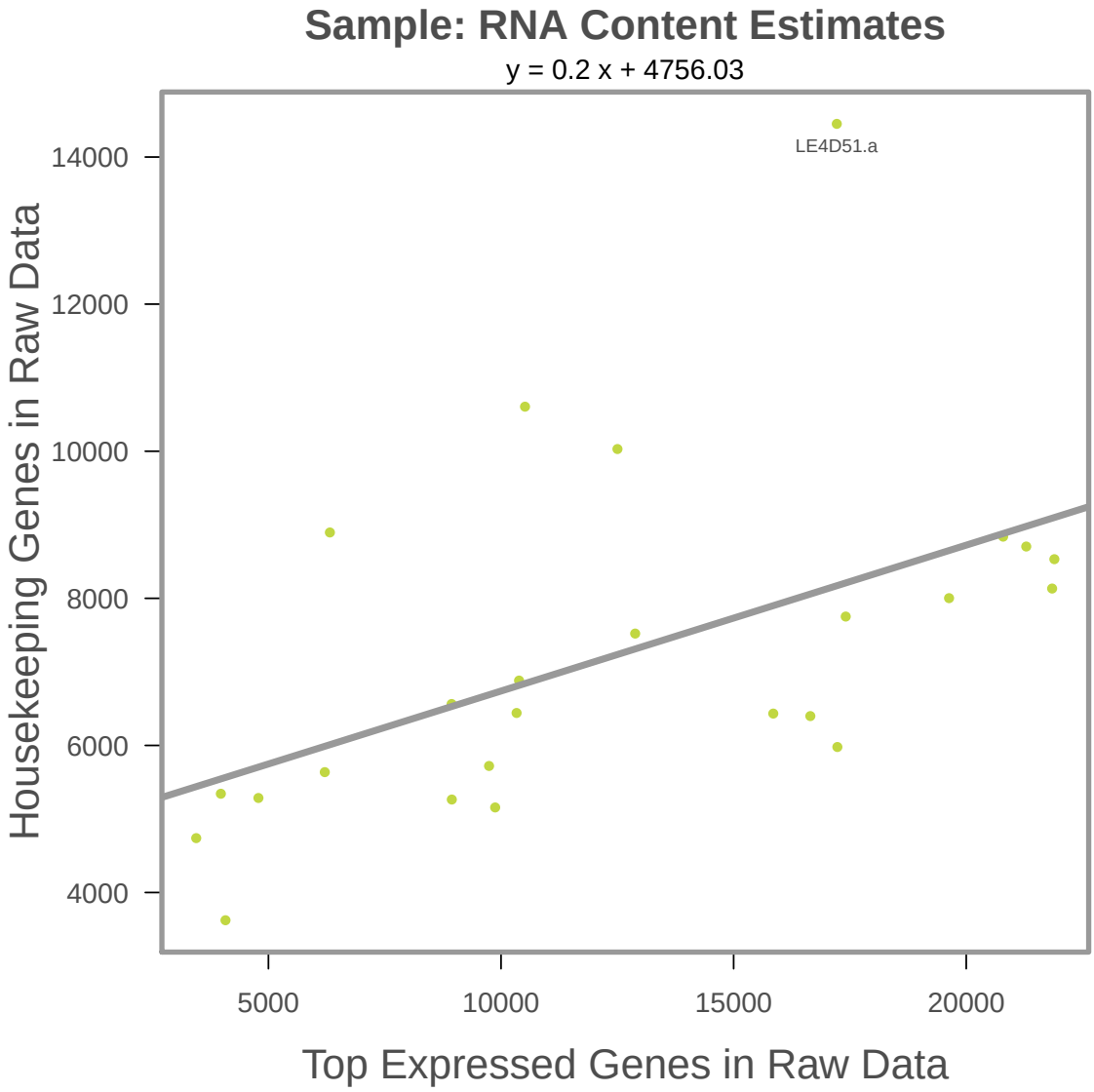


Figure 6: Estimates of input RNA or sample content. Each point represents a sample. The x-axis is the RNA estimate for each sample based on the geometric mean of the top 75 expressed genes, and the y-axis is the RNA estimate based on the geometric mean housekeeping genes.

Correlation of technical covariates with summary gene features. If still present after normalization, correlation could indicate residual technical variation or batch effects. This could be problematic if the study has an unbalanced design, which could result in a gene vs. trait model being confounded by a third technical covariate. For example, cartridge 1 has a high mean expression level and it contains a large proportion of females. Looking for differential expression between males and females will be confounded by females being over represented on an outlier cartridge.

In some instances a correlation of design covariate with a summary gene feature could be indicative of the hidden biological substructure in the data. Biological rationale for observed technical relationships is preferred if possible. For example, the last cartridge in your dataset is associated with low RNA content and elevated levels of missing values. You go back and check your lab notes and realize that you prioritized samples based on quality/quantity for running on the nCounter, so all the poor samples were left to the end.

These correlations can be described as batch effects and can be somewhat tricky to identify due to wide variations in experimental design and codesets. Generally, a batch effect is the situation when there is a systematic technical bias that normalization ignores or inadvertantly exacerbates. The following plots try and identify these situations by showing how sample level summaries (i.e. mean of positive controls) are a associated with specified traits or technical features. Mean, SD and missing are generally related to the biology and therefore are mostly descriptive. The sample level summaries for the controls, however should be in most cases, independent of features of the data. Therefore, any significant difference could be an observed bias. Normalization will take care of minor fluctuations but large differences i.e. large green dots should be flagged or at least reconciled. For example, if one biologial subgroup of interest (i.e. tumour tissue vs normal tissue) has much higher RNA content than the other, then the resulting normalization factor could over-correct and end up actually removing the true signal. Similarly, if cartridges, dates, or scanners have different levels of positive controls then the standard normalization techniques are unlikely to fully compensate resulting in bias when evaluating differential expression.

Correlation of biological covariates with summary gene features. While not strictly 'batch effects', any association could introduce a bias. For example, a large proportion of your tumour samples have high levels of missing values. This could be biologically plausible, or it could reflect a technical artifact (e.g. the tumour samples were taken from very old and degraded FFPE blocks).

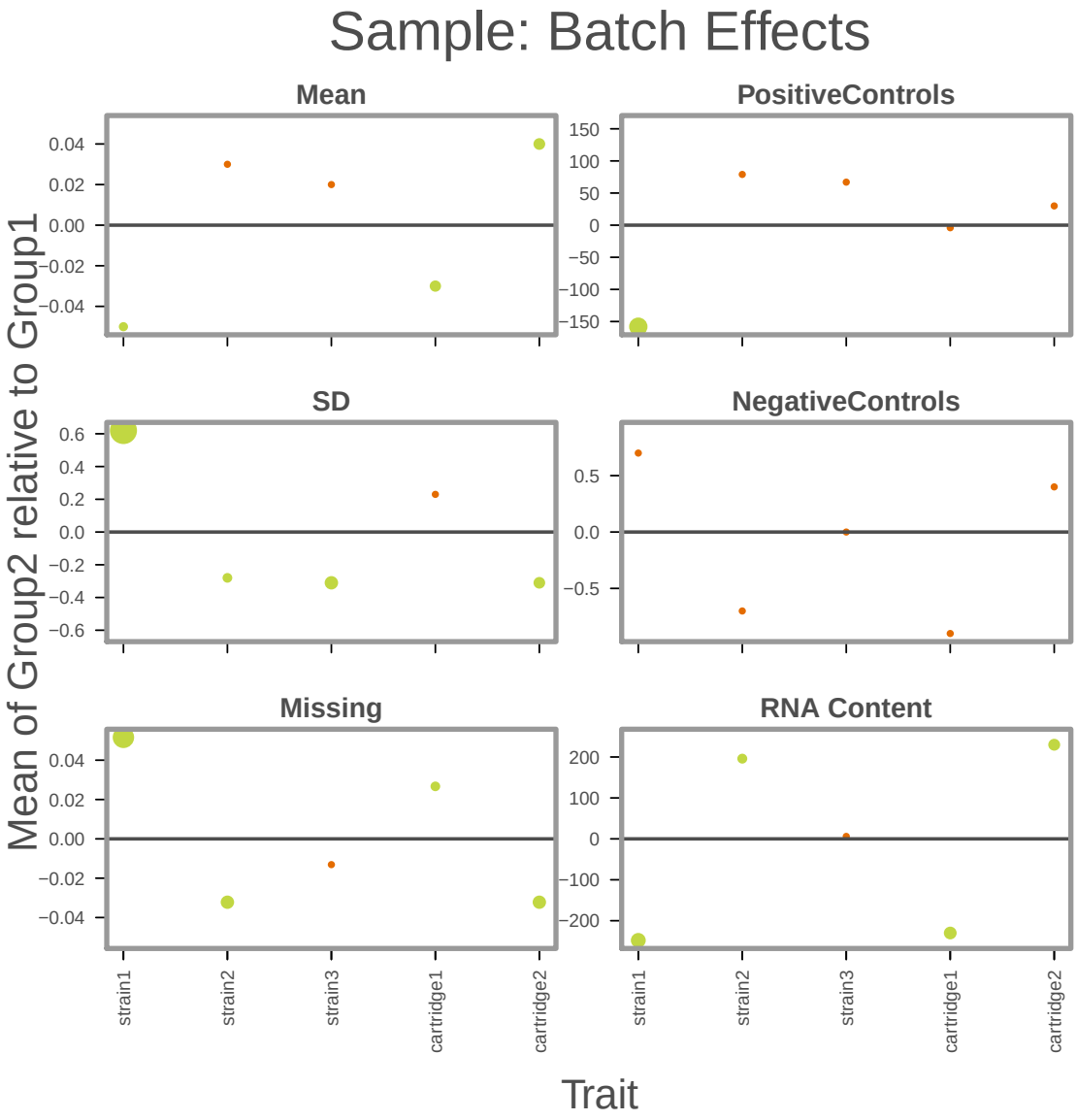


Figure 7: Batch effects and potential confounding. The x-axis lists binary traits and design variables. The y-axis is the difference between between the two levels of the trait for the specified sample summary feature. Usually, the trait label on the x-axis reflects the group being compared. i.e. strain1 vs the rest or cartridge2 vs the rest. Sample summaries can be mean counts, sd, proportion of missing (0 counts) or related to the controls probes. The location of the dots relative to the horizontal line show how different that group is relative to the rest. Green dots indicate that the difference is signicant based on a t-test and the size is proportional to the level of significance. For example strain2 (column 2) has a much larger mean count than all the other strains and also significantly less missing. Note that summaries for the controls are generated sequentially according to what was specified in the NanoStringNorm call. Therefore, the summarization of the PositiveControls is based on the raw counts while the RNA Content is based on the data adjusted for Positive and Negative controls.

It is important to review the quality and quantity of the normalization factors. The batch effects above show how these factors relate to groups of samples, while in contrast, the normalization factors plot shows these summaries at the sample level. A perceived batch effect could be the result of one or two outlier or influential samples. Samples that fluctuate beyond 100% from the mean are labelled for review.

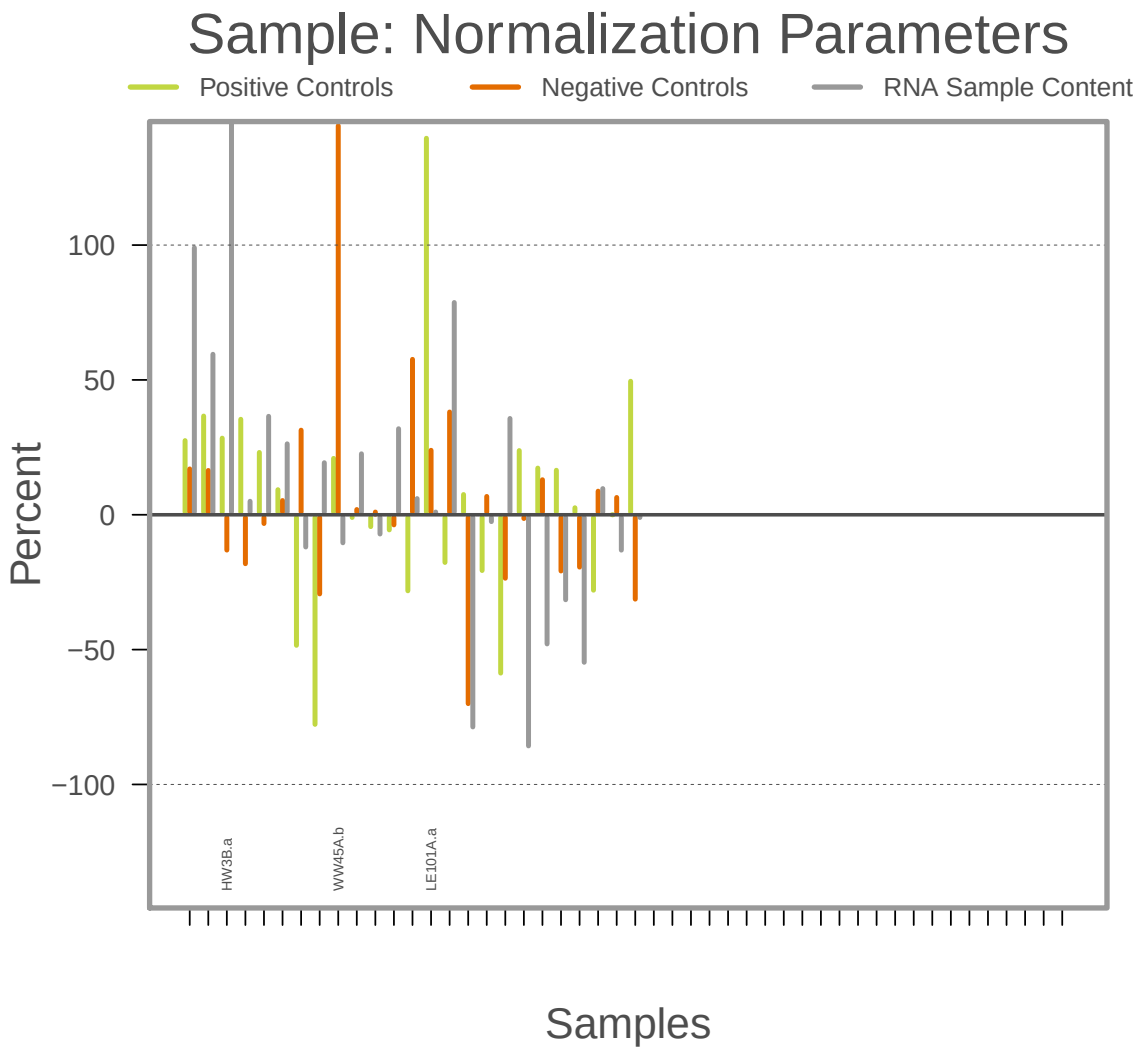


Figure 8: Barplot of normalization factors used to correct the data. For each sample the CodeCount (positive controls), Background (negative controls) and RNA sample content (housekeeping) normalization parameter is expressed as percent difference from the mean. Samples extending beyond 100% are annotated for being potential outliers. Any sample or group of samples that have very high or low bars could be influential outliers.

The following show the distribution of the observed versus expected positive and negative controls. The negative controls should be tightly clustered and the positive controls should show a strong linear trend. If any sample has a slope or intercept that strongly deviates from other samples then it should be used carefully. If any sample is flagged for any reason then these plots are a good place to start in the diagnostic process.

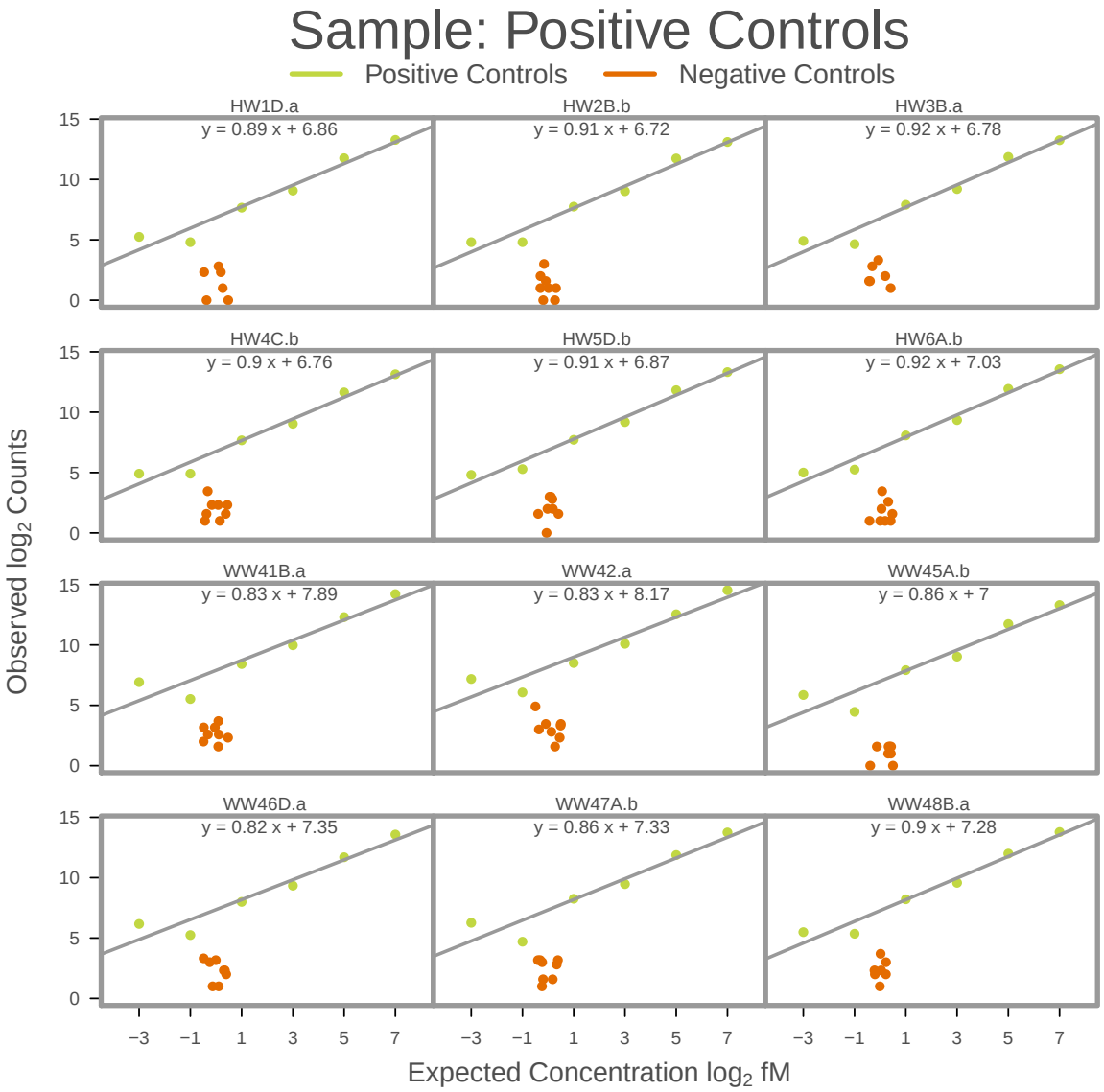


Figure 9: Scatterplot of positive and negative control counts. The green points show the expected concentration in fM of spiked in positive controls vs. the observed normalized counts. A best fit line is added where the intercept reflects sensitivity. Orange dots represent the negative controls.

6 Interactive Plotting

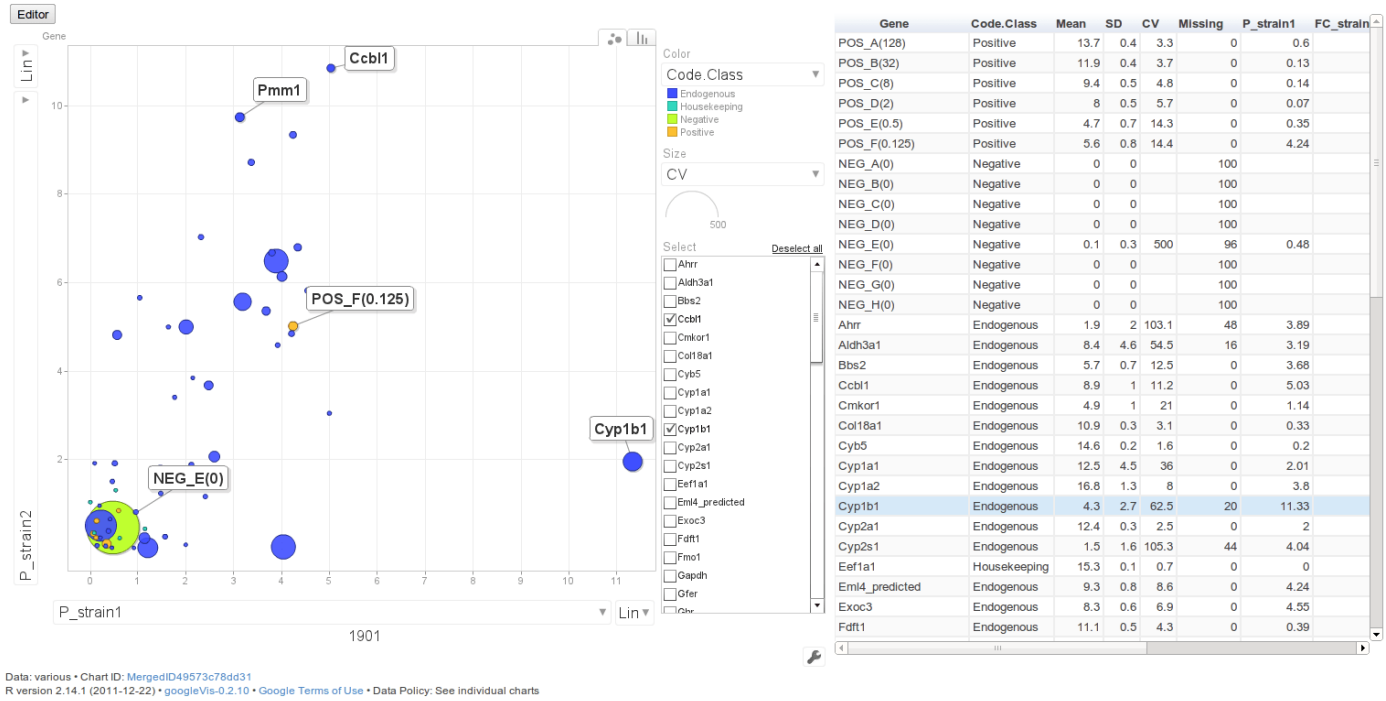
A set of Google Motion charts can be produced based on functionality in the *googleVis* package. These are highly interactive plots based on sample and gene summary output. The plots are very useful for the purposes of presentations and user directed data exploration. For example, one can compare the differential expression of uncorrelated traits to find unexpected overlaps.

By default the plots are rendered in your browser using a built in web server. For distribution and later use you will need to save the resulting html files. The use of a web server is required to display the results, and therefore the program attempts to download *mongoose* embedded web server http://code.google.com/p/mongoose/ (License MIT). After starting the binary/executable you can access the plots at http://127.0.0.1:8080.

The first example displays the plots in your browser the second saves them for later use.

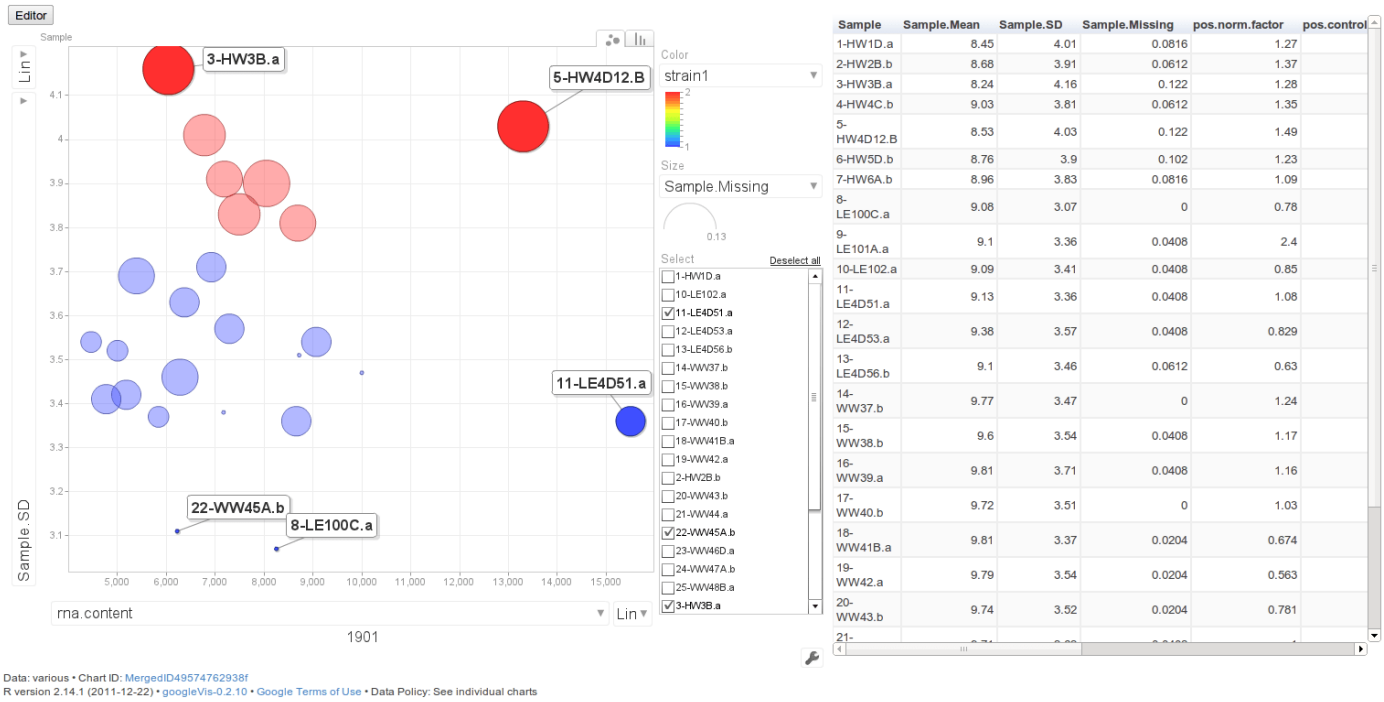
```
> # plot the sample summaries to your browser
> Plot.NanoStringNorm.gvis(
+         x = NanoString.mRNA.norm,
+         plot.type = c('gene.norm', 'sample'),
+         save.plot = FALSE
+         );

> # plot the gene summaries to a directory for distribution and later viewing
> # note.  if you save.plot = TRUE the default for path to mongoose is "web" i.e. it tries to download fro
> # alternatively, you can specify a path to the binary or use "none" as below.
> Plot.NanoStringNorm.gvis(
+         x = NanoString.mRNA.norm,
+         plot.type = c('gene.norm', 'sample'),
+         save.plot = TRUE,
+         path.to.mongoose = 'none',
+         output.directory = "NanoStringNorm_Interactive_Plot"
+         );
```



Gene	Code.Class	Mean	SD	CV	Missing	P_strain1	FC_strain
POS_A(128)	Positive	13.7	0.4	3.3	0	0.6	
POS_B(32)	Positive	11.9	0.4	3.7	0	0.13	
POS_C(8)	Positive	9.4	0.5	4.8	0	0.14	
POS_D(2)	Positive	8	0.5	5.7	0	0.07	
POS_E(0.5)	Positive	4.7	0.7	14.3	0	0.35	
POS_F(0.125)	Positive	5.6	0.8	14.4	0	4.24	
NEG_A(0)	Negative	0	0		100		
NEG_B(0)	Negative	0	0		100		
NEG_C(0)	Negative	0	0		100		
NEG_D(0)	Negative	0	0		100		
NEG_E(0)	Negative	0.1	0.3	500	96	0.48	
NEG_F(0)	Negative	0	0		100		
NEG_G(0)	Negative	0	0		100		
NEG_H(0)	Negative	0	0		100		
Ahrr	Endogenous	1.9	2	103.1	48	3.89	
Aldh3a1	Endogenous	8.4	4.6	54.5	16	3.19	
Bbs2	Endogenous	5.7	0.7	12.5	0	3.68	
Ccbl1	Endogenous	8.9	1	11.2	0	5.03	
Cmkor1	Endogenous	4.9	1	21	0	1.14	
Col18a1	Endogenous	10.9	0.3	3.1	0	0.33	
Cyb5	Endogenous	14.6	0.2	1.6	0	0.2	
Cyp1a1	Endogenous	12.5	4.5	36	0	2.01	
Cyp1a2	Endogenous	16.8	1.3	8	0	3.8	
Cyp1b1	Endogenous	4.3	2.7	62.5	20	11.33	
Cyp2a1	Endogenous	12.4	0.3	2.5	0	2	
Cyp2s1	Endogenous	1.5	1.6	105.3	44	4.04	
Eef1a1	Housekeeping	15.3	0.1	0.7	0	0	
Eml4_predicted	Endogenous	9.3	0.8	8.6	0	4.24	
Exoc3	Endogenous	8.3	0.6	6.9	0	4.55	
Fdft1	Endogenous	11.1	0.5	4.3	0	0.39	

Figure 10: A screen capture of the interactive Google chart for genes. Axis, colour and point size can be altered for any gene or for trait level summaries.



Sample	Sample.Mean	Sample.SD	Sample.Missing	pos.norm.factor	pos.control
1-HW1D.a	8.45	4.01	0.0816	1.27	
2-HW2B.b	8.68	3.91	0.0612	1.37	
3-HW3B.a	8.24	4.16	0.122	1.28	
4-HW4C.b	9.03	3.81	0.0612	1.35	
5-HW4D12.B	8.53	4.03	0.122	1.49	
6-HW5D.b	8.76	3.9	0.102	1.23	
7-HW6A.b	8.96	3.83	0.0816	1.09	
8-LE100C.a	9.08	3.07	0	0.78	
9-LE101A.a	9.1	3.36	0.0408	2.4	
10-LE102.a	9.09	3.41	0.0408	0.85	
11-LE4D51.a	9.13	3.36	0.0408	1.08	
12-LE4D53.a	9.38	3.57	0.0408	0.829	
13-LE4D56.b	9.1	3.46	0.0612	0.63	
14-WW37.b	9.77	3.47	0	1.24	
15-WW38.b	9.6	3.54	0.0408	1.17	
16-WW39.a	9.81	3.71	0.0408	1.16	
17-WW40.b	9.72	3.51	0	1.03	
18-WW41B.a	9.81	3.37	0.0204	0.674	
19-WW42.a	9.79	3.54	0.0204	0.563	
20-WW43.b	9.74	3.52	0.0204	0.781	
21-					

Figure 11: A screen capture of the interactive Google chart for samples. Axis, colour and point size can be altered for any gene or for trait level summaries.

7 Best Normalization Method

Different code sets and study designs are better suited towards alternative normalization strategies. This is an area of active research and development. In this section I'll outline several basic strategies for evaluating how well a particular method is doing at removing technical variation. The function `norm.comp` is a simple interface for looping over a set of methods and summarizing comparisons. Options include:

1. Signal to noise or *Coefficient of Variation* (CV) for control genes. The CV of the Positive controls is proportional to the technical variability introduced by the nCounter platform. The CV for the Housekeeping controls is proportional to the confounding biological variation due to sample input. The mean Endogenous CV shows the global noise of experimentally observed genes. Methods that minimize the CV are generally considered better. Be careful in comparing methods that change the scale of the measured counts i.e. zscores and vsn.

2. Correlation between replicates. If multiple groups of replicates are assayed then the *Intra Class Correlation* (ICC) is calculated. ICC is defined as the ratio of *between* group variation to the *total* variation. Variation between groups is relative to the correlation within groups. Replicates can be purely technical or reflect experimental design influences. For example titrations of different input RNA, samples run on different dates or even well chosen biological replicates.

3. Trait Discrimination. A method that maximizes trait difference i.e. Tumour - Normal status, could be considered good at removing technical variation. Similarly, any method that maximizes the predictive accuracy between two datasets would be desirable. This is not yet implemented.

Ideally, for large projects and new code sets, a pilot study will be run in order to optimize the normalization choice. For past projects we have chosen triplicate titrations of four samples and triplicate technical replicates for two tumour and two normal samples. Both designs require a twelve samples or a single cartridge.

```
> # compare normalization methods using coefficient of variation
> # of controls (CV) and correlation of replicates (ICC)
>
> # specifiy housekeeping genes in annotation
> NanoString.mRNA[NanoString.mRNA$Name %in%
+   c('Eef1a1','Gapdh','Hprt1','Ppia','Sdha'),'Code.Class'] <- 'Housekeeping';
> # strain x experimental condition i.e. replicate.
> # this is only a small subset of the original data  used for the plot
> biological.replicates <- c("HW_1.5_0","HW_1.5_0","HW_1.5_0","HW_1.5_100","HW_1.5_100","HW_1.5_100",
+                       "HW_6_100","HW_6_100","HW_3_100","HW_3_100","HW_3_100","HW_3_100",
+                       "LE_19_0","LE_19_0","LE_19_0","LE_96_0","LE_96_0","LE_96_0","HW_10_100",
+                       "HW_10_100","HW_10_100","HW_10_100","HW_6_100","HW_6_100","HW_96_0");
> norm.comp.results.test <- norm.comp(
+   x = NanoString.mRNA,
+   replicates = biological.replicates,
+   CodeCount.methods = 'none',
+   Background.methods = 'none',
+   SampleContent.methods = c('none','housekeeping.sum', 'housekeeping.geo.mean',
+                             'top.mean', 'top.geo.mean'),
+   OtherNorm.methods = 'none',
+   verbose = FALSE
+   );
```

In this example zscore normalization is ranked the highest based on ICC and interestingly sample content is better adjusted using total RNA and not the Housekeeping gene.

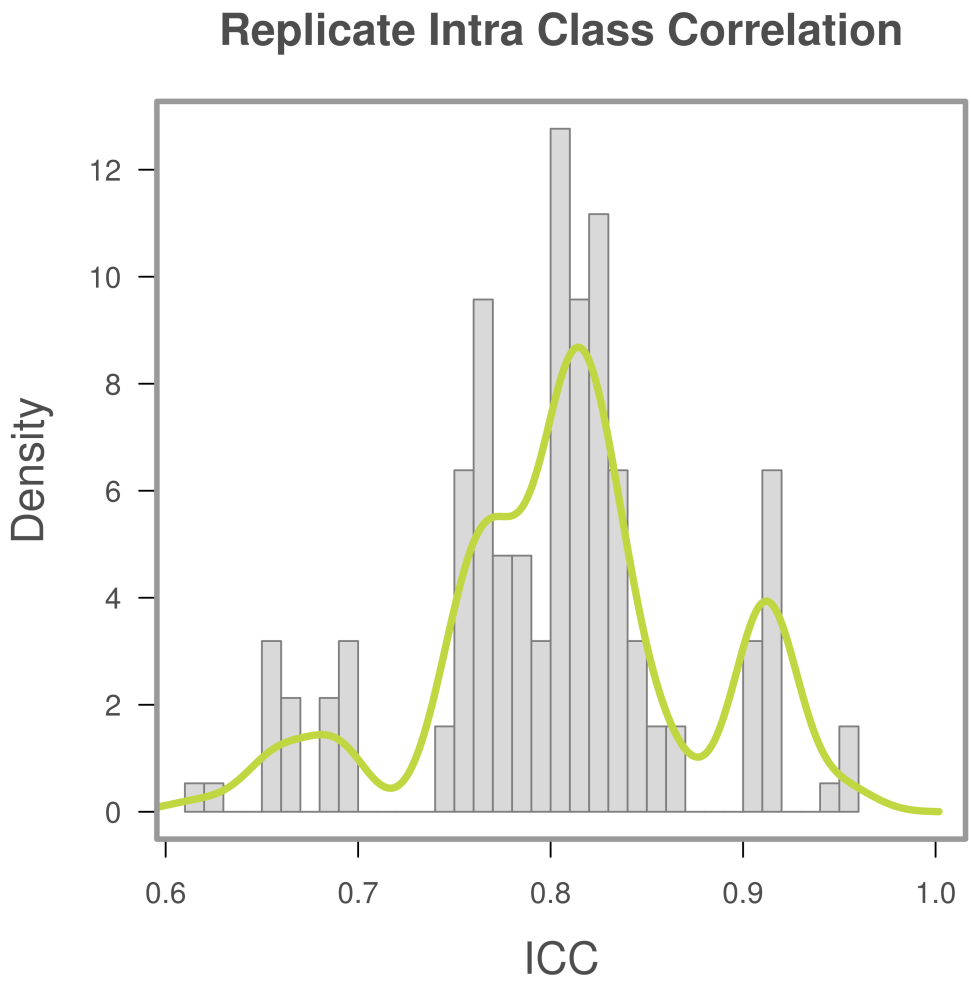


Figure 12: A histogram of ICC for all possible normalization methods on the rat strain data.

8 Study Design Related Issues

Based on the analysis of a number of miRNA and mRNA datasets, we've noticed several situations that tend present analytical challenges:

- Mixed tissues. Jointly pre-processing different tissues with very different expression distributions (i.e. presence of inhibitor's or T/N). This can introduce technical artifacts which increase false positive differences.
- Mixed samples. Including both frozen and FFPE sometimes produces dramatic differences even between replicates. Large variation in fixation process and date can also influence RNA integrity. This has been flagged in several studies.
- Housekeeping gene choice. Some common genes have been shown to be associated with phenotypes. Also, the validity of using mRNA in miRNA studies is unclear. In NanoString code sets the mRNA probes do not undergo ligation and therefore are under different technical influences.
- Using global or top normalization when a large proportion of genes are associated with phenotype (i.e. EBV and NPC).
- Plasma or circulating profiles. Very specific care needs to be taken on concise input volumes and quantification. Moreover, additional spike-in controls are needed. Inhibitors have been shown to be an issue.
- nCounter batch variability. While not as of yet quantified, date is a contributing factor. Jointly normalizing and analyzing data across multiple batches should be done carefully.

At this time, adaptable methods are being developed which better account for experimental design and technical variability. This document will be updated as more information becomes available.