Super-delta: A new approach that combines gene expression data normalization and differential expression analysis

Yuhang Liu, Xing Qiu, and Jinfeng Zhang

August 3, 2017

0 Introduction

0.1 Background

Normalization is an important data preparation step in gene expression analyses, designed to remove various systematic noise. Sample variance is greatly reduced after normalization, hence the power of subsequent statistical analyses is likely to increase. On the other hand, variance reduction is made possible by borrowing information across all genes, including differentially expressed genes (DEGs) and outliers, which will inevitably introduce some bias. This bias typically inflates type I error; and can reduce statistical power in certain situations. In this vignette we propose a new differential expression analysis pipeline, dubbed as `super-delta`, that consists of a multivariate extension of the global normalization and a modified t-test. A robust procedure is designed to minimize the bias introduced by DEGs in the normalization step. The modified t-test is derived based on asymptotic theory for hypothesis testing that suitably pairs with the proposed robust normalization.

0.2 Results

We first compared `super-delta` with four commonly used normalization methods: **global, median-IQR, quantile, and cyclic loess normalization** in simulation studies. `Super-delta` was shown to have better statistical power with tighter type I error control than its competitors. In many cases, the performance of `super-delta` is clos to an oracle test in which datasets without **technical noise** were used in analyses. We then applied all methods to a collection of gene expression data sets on breast cancer patients who received neoadjuvant chemotherapy. While there is a substantial overlap of the DEGs identified by all of them, `super-delta` were able to identify comparatively more DEGs than its competitors. Downstream gene set enrichment analysis confirmed that all these methods selected largely consistent pathways. Detailed investigations on the relatively small differences showed that pathways identified by `super-delta` have better connections to breast cancer than other methods.

0.3 Conclusions

As a new pipeline, `super-delta` provides new insights to the area of differential gene expression analysis. Solid theoretical foundation supports its asymptotic unbiasedness and noise-free properties. Implementation on real and simulated datasets demonstrates its decent performance compared with state-of-art procedures. It also has potential of expansion to be incorporated with other data type and/or more general between-group comparison problems.

1 Real Data Analysis

1.1 Read in Data

In this vignette, we will use the microarray expression data already arranged in a matrix, downloadable with the super-delta package. We already filter out a half of genes/probe sets with low variability. Together with the gene expressions are the patient group labels (1 = "pCR", -1 = "RD"), and a mapping file between probe set names and official gene symbols. Other necessary R and Bioconductor packages also need to be loaded for data analysis and plotting purposes.

```
library(genefilter)
library(superdelta)
library(ggplot2)
library(reshape2)
library(limma)
set.seed(5454)
## Set working directory here before loading data:
load("TxA_All.RData")
TxA_sig <- matrix(0,3,7)
adj <- c("none", "BH", "bonferroni")
nor <- c("none", "global", "medIQR", "quant", "cyc_loess", "deltaseq", "superdelta")
rownames(TxA_sig) <- adj; colnames(TxA_sig) <- nor
TxA_teststat <- TxA_rawp <- matrix(0,11141,7)
TxA_result <- list(7)
```

1.2 Normalization and Differential Expression Analysis

The next block of program implements differential expression analysis for all competing methods. `normalize.quantile`, `normalize.global`, and `normalize.medIQR` are functions in the *super-delta* package that incorporate the normalization procedures indicated by their names. The function `log2origin` is also self-written, for the convenience of transforming log2 expression data back to the original scale. Note that in this vignette, we use the definition of fold-change from [1], that is the ratio of mean expression on original scale. An alternative definition can be found in [2]. The basic syntax of our main function *superdelta* is as follows:

```
superdelta <- function(X, classlabel, test="t", side=c("two.sided", "less", "greater"),
methods = "robust", trim = 0.2, baseline = "auto", ...){
...
}
```

In this function, the usage of arguments are detailed as follows:

- `X` is the input expression matrix, where we follow the convention that genes are on rows and samples are on columns.
- `classlabel` is a vector with the group labels. Its length must equal the number of columns of `X`. It can be either numeric, character, or factor, but it must have two unique values.
- `test = "t"` is currently the only test implemented by this package.
- `methods` is method of estimating t-statistics. "robust" is recommended, but other options include "mean" and "median".
- `trim` is the trimming proportion of robust median fold trimmed median estimator. This parameter is only used when `methods` = "robust". The default value is 0.2. In practice, we suggest the range of this parameter to be between 0.1 and 0.4.

- `baseline` is the set of candidate pairing genes. When the total number of genes is smaller than 1,000 or "all" option is on, all genes will be used. When "auto" is on and total number of genes is larger than 1,000, only 1,000 randomly selected genes will make up the candidate set of baseline genes. In practice, using "auto" can speed up the process when the input dataset is large or repeated simulation is required, while the performance will be similar to that of using "all".

```
FC_ratio <- apply(log2origin(TxA_filt[,TxA.response==1]),1,mean)/
apply(log2origin(TxA_filt[,TxA.response==-1]),1,mean)
Data_cycloess <- normalizeCyclicLoess(TxA_filt, method = "fast")
Data_quant <- normalize.quantile(TxA_filt)
rownames(Data_quant) <- rownames(Data_cycloess)
colnames(Data_quant) <- colnames(Data_cycloess)
Data_global <- normalize.global(TxA_filt)
Data_medIQR <- normalize.medIQR(TxA_filt)
TxA_result[[1]] <- t_test_genes(TxA_filt, classlabel = TxA.response, na.rm = TRUE)
TxA_result[[2]] <- t_test_genes(Data_global, classlabel = TxA.response, na.rm = TRUE)
TxA_result[[3]] <- t_test_genes(Data_medIQR, classlabel = TxA.response, na.rm = TRUE)
TxA_result[[4]] <- t_test_genes(Data_quant, classlabel = TxA.response, na.rm = TRUE)
TxA_result[[5]] <- t_test_genes(Data_cycloess, classlabel = TxA.response, na.rm = TRUE)
TxA_result[[6]] <- deltaseq2(TxA_filt, classlabel = TxA.response)
TxA_result[[7]] <- superdelta(TxA_filt, classlabel = TxA.response, test = "t",
                side="two.sided", methods="robust", trim=0.2, baseline="all")
```

After running the DE analyses, we also obtain number of up- and down- regulated genes separately, based on both p-value (raw, Benjamini-Hochberg adjusted, and Bonferroni adjusted) and fold-change cutoff.

```
for (j in 1:7){
  TxA_teststat[,j] <- TxA_result[[j]]$teststat
  TxA_rawp[,j] <- TxA_result[[j]]$rawp
}
colnames(TxA_teststat) <- nor; colnames(TxA_rawp) <- nor
TxA_adjp_BH <- TxA_adjp_Bon <- matrix(0, 11141, 7)
colnames(TxA_adjp_BH) <- colnames(TxA_adjp_Bon) <- nor
for (j in 1:7){
  TxA_adjp_BH[,j] <- p.adjust(TxA_rawp[,j], method = "BH", n = nrow(TxA_filt))
  TxA_adjp_Bon[,j] <- p.adjust(TxA_rawp[,j], method = "bonferroni", n = nrow(TxA_filt))
}
TxA_sig <- matrix(0, nrow = 3, ncol = 7)
for (k in 1:7){
  TxA_sig[1,k] <- sum(TxA_rawp[,k]<=0.05 & abs(log(FC_ratio))>=log(1.25))
  TxA_sig[2,k] <- sum(TxA_adjp_BH[,k]<=0.05 & abs(log(FC_ratio))>=log(1.25))
  TxA_sig[3,k] <- sum(TxA_adjp_Bon[,k]<=0.05 & abs(log(FC_ratio))>=log(1.25))
}
rownames(TxA_sig) <- adj; colnames(TxA_sig) <- nor
rawp_sig_ind <- BH_sig_ind <- Bon_sig_ind <- list(7)
for (k in 1:7){
  rawp_sig_ind[[k]] <- which(TxA_rawp[,k]<=0.05 & abs(log(FC_ratio))>=log(1.25))
  BH_sig_ind[[k]] <- which(TxA_adjp_BH[,k]<=0.05 & abs(log(FC_ratio))>=log(1.25))
  Bon_sig_ind[[k]] <- which(TxA_adjp_Bon[,k]<=0.05 & abs(log(FC_ratio))>=log(1.25))
}
TxA_down <- TxA_up <- matrix(0,3,7)
for (k in 1:7){
  TxA_down[1,k] <- sum(TxA_teststat[,k][rawp_sig_ind[[k]]]<0)
```

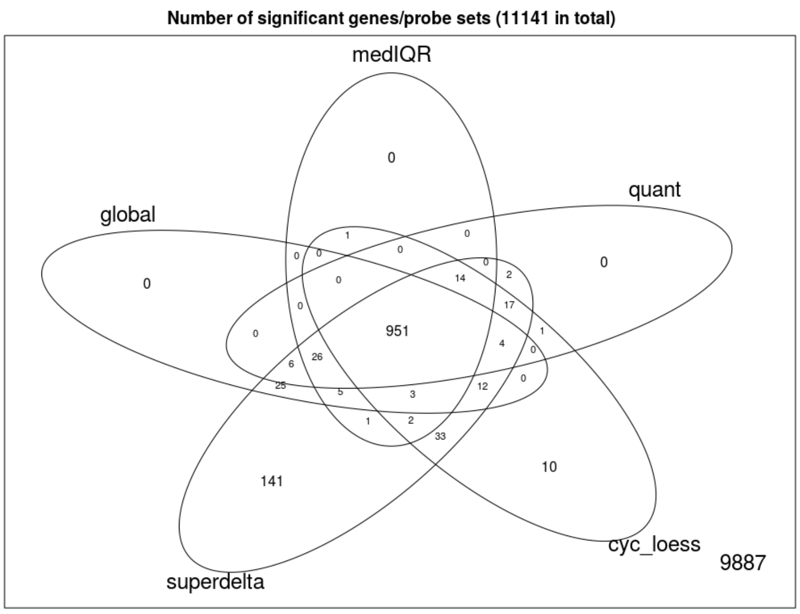

Figure 1: Venn diagram of significant genes (numbers in figure are numbers of genes)

```
  TxA_up[1,k] <- sum(TxA_teststat[,k][rawp_sig_ind[[k]]]>0)
  TxA_down[2,k] <- sum(TxA_teststat[,k][BH_sig_ind[[k]]]<0)
  TxA_up[2,k] <- sum(TxA_teststat[,k][BH_sig_ind[[k]]]>0)
  TxA_down[3,k] <- sum(TxA_teststat[,k][Bon_sig_ind[[k]]]<0)
  TxA_up[3,k] <- sum(TxA_teststat[,k][Bon_sig_ind[[k]]]>0)
}
```

1.3 Summarizing and Demonstrating Results

We want to create a Venn Diagram of the five significant gene lists. We can see that while there is a substantial overlap of the five DE gene lists, `super-delta` is able to identify approximately $10 - 15\%$ more DEGs than its competitors. To justify that these more caught genes are likely true but not false positives, we design a comprehensive simulation study in the next section to investigate the power and Type I error control of `super-delta`.

```
global <- 1:11141 %in% BH_sig_ind[[2]]
medIQR <- 1:11141 %in% BH_sig_ind[[3]]
quant <- 1:11141 %in% BH_sig_ind[[4]]
cyc_loess <- 1:11141 %in% BH_sig_ind[[5]]
superdelta <- 1:11141 %in% BH_sig_ind[[7]]
c3 <- cbind(global, medIQR, quant, cyc_loess, superdelta)
a <- vennCounts(c3)
vennDiagram(a, main = "Number of significant genes/probe sets (11141 in total)")
```

2 Simulation Study

2.1 Basic Rationale

Three related simulation strategies were designed to investigate the power and type I error control of all procedures covered in this study. To achieve verisimilitude of the simulation, we suggest estimating all model parameters from the real biological data. Specifically, we estimated the per-sample random effect term variation $\hat{\eta} = 0.873$ and the random error term variation $\hat{\sigma} = 0.617$.

	Classical					Super-delta		
	oracle	global	medIQR	quantile	cyclic-loess	mean	median	MFTM
SIM1								
Power	88.82(1.02)	88.17(1.08)	87.84(0.99)	87.71(1.00)	87.61(1.01)	88.48(1.04)	88.89(1.13)	88.85(1.00)
Type I error	0.46(0.07)	0.49(0.08)	0.50(0.07)	0.51(0.08)	0.52(0.07)	0.44(0.08)	0.41(0.08)	0.40(0.07)
SIM2								
Power	92.11(0.81)	90.94(0.89)	90.37(0.95)	90.17(0.93)	89.94(0.95)	91.26(1.01)	91.79(0.94)	92.09(0.77)
Type I error	0.93(0.14)	1.08(0.17)	1.25(0.22)	1.26(0.19)	1.36(0.23)	1.03(0.22)	0.85(0.15)	0.83(0.15)
SIM3								
Power	89.18(1.49)	76.67(1.51)	77.28(1.84)	76.20(1.63)	76.51(1.59)	77.16(1.89)	86.05(1.70)	88.55(1.62)
Type I error	0.61(0.11)	1.53(0.19)	1.42(0.20)	1.55(0.20)	1.52(0.20)	1.46(0.26)	0.61(0.14)	0.54(0.12)

Table 1: A summary table of three simulation scenarios. Sample size is 50 for both groups. All p-values are **Benjamini-Hochberg** adjusted; **Power**: Approximate statistical power; **Type I error**: Approximate Type I error rate. All these measurements are calculated by averaging over 50 simulations. Numbers within parentheses are standard deviations. All numbers are pertentage rates.

The signal-to-noise ratio of 1.41 motivates us in the default parameter values when designing the simulation function. We sorted the absolute values of log2 fold changes and selected the first 1000 (363 up and 637 down regulations) to be considered as true signal (mean group difference). The simulation design is presented as follows.

SIM1: Number of genes is $10,000$. True signals are 363 up and 637 down regulations.

SIM2: Number of genes is $5,000$. True signals are 363 up and 637 down regulations. Compared with **SIM1**, the only difference is that total number of genes is reduced by a half, which makes the proportion of DE genes doubled.

SIM3: Number of genes is $5,000$. True signals include only the 637 down regulations and all up regulations are removed. Compared with **SIM2**, **SIM3** has a more extreme and unbalanced DE structure.

For each simulation study, three sample sizes (number of slides in each group) were used: **n=50, 75, 100**. One more scenario of unequal sample size $n_1 = 50, n_2 = 100$ was also included.
Although we strongly encourage the use of MFTM in conjunction with `super-delta`, we include two alternative methods, untrimmed mean and median, of selecting representative statistic in `super-delta` in simulation studies for comparison. Details of these two methods can be found in our companion paper.

2.2 Simulation function for differential expression analysis

The syntax and usage of the *simu* function is as follows:

```
simu <- function(NGENES = 10000, n1 = 50, n2 = 50, bl=NULL, fc = 2^rnorm(NGENES,0,0.5),
SIM = c("SIM1", "SIM2", "SIM3"), eta = 1.5, sigma = 1){
...
}
```

- `NGENES` is number of genes input.
- `n1, n2` are the two group sample sizes.
- `bl` is user input baseline mean log2 expression of one group.
- `fc` is fold-change on raw scale of all input genes ([1]).
- `eta` and `sigma` are model parameters, default signal-to-noise ratio 1.5.

Number of genes used	Mean accuracy(SD) of quantile	Mean accuracy(SD) of `super-delta` + MFTM
10	77.20(7.69)	74.92(7.20)
20	85.10(7.19)	85.48(5.59)
50	88.04(5.89)	95.08(2.88)
100	89.72(3.70)	99.44(0.61)

Table 2: A summary table of prediction accuracy. Sample size is 50 for both groups. Support vector machine is used to build classifier. All numbers in the table are percentages calculated by averaging over 50 simulations. Numbers within parentheses are standard deviations.

2.3 Simulation function for prediction of patient group

The syntax and usage of the *simu_pred* function is as follows. Compared with the previous function, the only thing added is number of top DE genes to be used for prediction. Currently the prediction model implements **support vector machine** (SVM) algorithm.

```
simu_pred <- function(NGENES=10000, n1=50, n2=50, bl=NULL, fc=2^rnorm(NGENES,0,0.5),
SIM=c("SIM1", "SIM2", "SIM3"), TopGenes=50, eta=1.5, sigma=1){
...
}
```

- `NGENES` is number of genes input.
- `n1, n2` are the two group sample sizes.
- `bl` is user input baseline mean log2 expression of one group.
- `fc` is fold-change on raw scale of all input genes ([1]).
- `eta` and `sigma` are model parameters, default signal-to-noise ratio 1.5.
- `TopGenes` is number of most significant genes used to build SVM classifier.

3 Concluding Remarks

In summary, we proposed a differential gene expression analysis pipeline that consists of a multivariate extension of the global normalization method (the δ step) to remove sample-specific variation; an adjusted two sample Welch t-test (the test step) that takes the variation of both genes of interest and their pairs into consideration; and a robust trimming algorithm (the summarizing step) to select one overall statistic to represent the empirical distribution of δs pertain to every gene. Once these representative statistics (t_i^{MFTM}) are calculated, unadjusted and adjusted p-values can be obtained by standard inferential practice.
Traditional normalization procedures calculate the normalized expression levels in one step and they borrow information from both DE and NDE genes to remove sample-specific variation. Such practice can introduce nontrivial bias when the effects of up- and down-regulated genes to normalization are not exactly the same. In comparison, `super-delta` first normalizes every gene by all other genes, which generates thousands of normalized values that can help adjust the t-statistic for this gene. Because we use subtraction as a means to remove per-sample variation, the δ step can be considered as a multivariate extension of the global normalization. In fact, if we take the mean of all $t_{ii'}$, for $i' \neq i$, as the representative statistic for the ith gene, the results are very similar to that of the global normalization. On the other hand, the multivariate nature of the δ step enables us to apply a robust trimming algorithm to filter out certain percentages of extremely large or small statistics that are likely to cause bias to the subsequent inference. This filter reduces false discoveries significantly and improves statistical power at the same time when the gene expression pattern is relatively strong and highly unbalanced (for example, SIM3).

One under-appreciated but important advantage which `super-delta` inherited from `delta-seq` is that we can identify pairing genes (a.k.a. empirical house-keeping genes) by data-driven methods. Biological data analysis shows that these pairing genes either have very broad biological functions thus are good candidate for house-keeping genes; or they play direct or indirect roles in immunity. Further investigations are needed to fully understand the interplay between the main DEGs and those immunity-related pairing genes. Additionally, `super-delta` is a "local" normalization method which makes it especially suitable for real-world applications.
Imaging that for the reason of saving cost and processing time, we are only allowed to use a handful of top DE genes as biomarkers in a commercialized *portable* diagnostic device with very limited computational power. We will not be able to faithfully reproduce the differential expression results as defined by the traditional normalization methods because we don't have enough genes to calculate per-sample mean or median accurately, much less a "reference quantile curve". Commercial diagnostic tools based on gene expression biomarkers usually rely on polymerase chain reaction (PCR) based platforms, which are more economic and convenient, to measure the expressions for a very small set of pre-specified genes. Biomarker discoveries using traditional normalization methods on microarray or next generation sequencing data cannot be directly translated into these PCR based platforms because the same normalization procedure cannot be performed on PCR platforms. This may be an important reason that accounts for the poor success rate when these biomarkers were tested clinically. On the other hand, for `super-delta`, p DE genes need at most p pairing genes in the δ step. Considering that most pairing genes are reused by more than one gene, so the actual number of pairing genes needed to reproduce the results obtained from `super-delta` may be even less than p (hence, total number of genes used less than $2*p$). This is a meaningful direction. Based on this, the simulation in section **2.2.2** is developed to investigate the performance of `super-delta` in terms of prediction of patient subtype.
We believe `super-delta` can be extended to solve other inferential problems such as one-way ANOVA and linear regression. All we need to do is to prove similar asymptotic properties as the **main Theorem** in our companion paper. Another extension to `super-delta` is also possible but may need much more investigation: To create a multivariate-version of the quantile normalization. However, it is not obvious to map the quantile normalization to a local computation just between two genes. One possible solution is to use a first step quantile normalization as a rough guide and then use either weighted linear combination or even a nonlinear transformation for local normalization. A thorough theoretical and empirical study in this direction could be very rewarding in the future.
Finally, we would like to discuss the applicability of `super-delta` to expression data generated by RNA-seq technology [3, 4, 5]. Although raw RNA-seq reads are *discrete* random variables that are generally modeled by non-normal distributions such as negative binomial distribution [6, 7], it is a common practice to apply non-specific filtering to remove genes with very low reads and then use log-transformation to stabilize variance. These pre-processing steps reduce the granularity of the RNA-seq data and make the distribution much more normal. In fact, some recent comparative studies [8, 9, 10] showed that differential expression analysis tools designed for continuous data can achieve comparable, sometimes even slightly better, performance than those based on discrete models. Based on these considerations, we believe that with appropriate adaptations, `super-delta` can be made applicable for pre-processed RNA-seq data. This is a direction we are actively working in.

References

[1] Tusher VG, Tibshirani R, Chu G. Significance analysis of microarrays applied to the ionizing radiation response. Proc Natl Acad Sci U S A. 2001 Apr;98(9):5116–5121.

[2] Guo L, Lobenhofer EK, Wang C, Shippy R, Harris SC, Zhang L, et al. Rat toxicogenomic study reveals analytical consistency across microarray platforms. Nature biotechnology. 2006;24(9):1162.

[3] Garber M, Grabherr MG, Guttman M, Trapnell C. Computational methods for transcriptome annotation and quantification using RNA-seq. Nature methods. 2011;8(6):469–477.

[4] Mortazavi A, Williams BA, McCue K, Schaeffer L, Wold B. Mapping and quantifying mammalian transcriptomes by RNA-Seq. Nature methods. 2008;5(7):621–628.

[5] Wang Z, Gerstein M, Snyder M. RNA-Seq: a revolutionary tool for transcriptomics. Nature Reviews Genetics. 2009;10(1):57–63.

[6] Anders S, Huber W. Differential expression analysis for sequence count data. Genome biology. 2010;11(10):1.

[7] Robinson MD, McCarthy DJ, Smyth GK. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics. 2010 Jan;26(1):139–140.

[8] Law CW, Chen Y, Shi W, Smyth GK. Voom: precision weights unlock linear model analysis tools for RNA-seq read counts. Genome biology. 2014;15(2):1.

[9] Rapaport F, Khanin R, Liang Y, Pirun M, Krek A, Zumbo P, et al. Comprehensive evaluation of differential gene expression analysis methods for RNA-seq data. Genome biology. 2013;14(9):1.

[10] Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, et al. limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic acids research. 2015;1:gkv007.