Linnorm User Manual

Shun H. Yip[1,2], Pak Chung Sham[1,3,4], Junwen Wang[1]

[1] Centre for Genomic Sciences, LKS Faculty of Medicine,
The University of Hong Kong, Hong Kong SAR, China;
[2] School of Biomedical Sciences, LKS Faculty of Medicine,
The University of Hong Kong, Hong Kong SAR, China;
[3] Department of Psychiatry, LKS Faculty of Medicine,
The University of Hong Kong, Hong Kong SAR, China;
[4] State Key Laboratory in Cognitive and Brain Sciences,
The University of Hong Kong, Hong Kong SAR, China.

March 16, 2016

Abstract

Linnorm is an R package for the analysis of RNA-seq expression data. Its main function is to normalize RNA-seq expression data for parametric tests. Examples of such tests include using *limma* for differential expression analysis or calculating Pearson Correlation Coefficient for gene correlation study. Compared to previous methods, Linnorm has great compatibility and is able to normalize raw count, CPM, RPKM, FPKM and TPM units in positive real numbers. Additionally, Linnorm provides the RnaXSim function for the simulation of RNA-seq raw counts for the evaluation of differential expression analysis methods. RnaXSim can simulate RNA-seq dataset in Gamma, Log Normal, Negative Binomial or Poisson distributions.

Contents

1 Introduction **3**
1.1 Linnorm . . . 3
1.1.1 Input Format . . . 3
1.1.2 Compatible Upstream Expression Quantification Tools . . . 3
1.1.3 Expression Types . . . 3
1.1.4 Data Types and Format . . . 3
1.2 RnaXSim . . . 4
1.2.1 Inputs . . . 4

2 Examples with Source Codes **5**
2.1 Before we get started . . . 5
2.2 Linnorm . . . 5
2.2.1 Linnorm-limma Differential Expression Analysis . . . 5
2.2.1.1 Analysis procedure . . . 5
2.2.1.2 Data analysis . . . 6
2.2.2 Linnorm Normlization for Parametric Tests . . . 8
2.2.2.1 Linnorm Normalization . . . 8
2.2.2.2 Gene Correlation Analysis . . . 8
2.2.2.3 Calculate Fold Change . . . 9
2.3 RnaXSim . . . 12
2.3.1 RNA-seq Raw Count Simulation . . . 12
2.3.1.1 Default . . . 12
2.3.1.2 Advanced . . . 12

3 Frequently Asked Questions **13**
3.1 How do I convert Linnorm Normalized dataset back to CPM/TPM? . . . 13
3.2 Can I use Linnorm Normalized dataset to calculate Fold Change? . . . 14
3.3 I only have 1 replicate for each sample set. Can I perform Linnorm Normalization? . . . 14
3.4 I only have 1 replicate for each sample set. Can I perform Differential Expression Analysis with Linnorm and limma? . . . 14
3.5 There are a lot of fold changes with INF values in Linnorm.limma output. Can I convert them into numerical units like those in the voom-limma pipeline? . . . 14

4 Bug Reports, Questions and Suggestions **14**

1 Introduction

The Linnorm R package contains a variety of functions for RNA-seq data analysis. Currently, it has two main functions.

- RNA-seq Expression Normalization (Linnorm)
  - Differential Expression Analysis
  - Gene Correlation Analysis
- RNA-seq Raw Count Simulation (RnaXSim)

1.1 Linnorm

1.1.1 Input Format

Linnorm accepts any RNA-seq Expression data, including but not limited to

- Raw Count
- Count per Million (CPM)
- Reads per Kilobase per Million reads sequenced (RPKM)
- expected Fragments Per Kilobase of transcript per Million fragments sequenced (FPKM)
- Transcripts per Million (TPM)

1.1.2 Compatible Upstream Expression Quantification Tools

Linnorm inputs can accept raw counts generated by simple count methods like HTSeq and *Rsubread*. Expression matrices generated by supervised learning algorithms, such as RSEM, Sailfish, Cufflinks and etc are also compatible.

1.1.3 Expression Types

Linnorm can accept any RNA-seq expression data, including but not limited to

- Gene
- Exon
- Isoform
- miRNA
- smRNA

1.1.4 Data Types and Format

Linnorm accepts matrix as its data type. Data frames are also accepted and will be automatically converted into the matrix format before analysis. Each column in the matrix should be a sample or replicate. Each row should be a Gene/Exon/Isoform/etc.

Example:

	Sample 1	Sample 2	Sample 3	...
Gene1	1	2	1	...
Gene2	3	0	4	...
Gene3	10.87	11.56	12.98	...
Gene4	21.23	0	56.44	...
...	...	...	...	...
...	...	...	...	...

Please note that undefined values such as NA, NaN, INF, etc are **NOT** supported.

1.2 RnaXSim

The Linnorm package provides the RnaXSim function for the simulation of RNA-seq expression data given a distribution. Supported distributions include:

- Gamma Distribution
- Log Normal Distribution
- Negative Binomial Distribution
- Poisson Distribution

1.2.1 Inputs

Acceptable input format, expression types and data types are the same as Linnorm's section.

However, each sample in the data matrix are assumed to be replicates of each other. One good source of such data is the *seqc* package.

2 Examples with Source Codes

2.1 Before we get started

- SEQC dataset
  - In our examples, we will use RNA-seq data from *seqc*.
- limma package
  - *limma* is imported with Linnorm. Please cite it separately if you use limma for differential expression analysis in published work.

2.2 Linnorm

2.2.1 Linnorm-limma Differential Expression Analysis

2.2.1.1 Analysis procedure In this example, we are going to randomly choose 6 replicates of RNA-seq expression data from SEQC's sample A and 8 replicates from Sample B. Then, we will perform differential expression analysis on it.

1. Get RNA-seq data from Sample A and B.

```
library(seqc)

SampleA <- ILM_aceview_gene_BGI[,grepl("A_",colnames(ILM_aceview_gene_BGI))]
rownames(SampleA) <- ILM_aceview_gene_BGI[,2]

SampleB <- ILM_aceview_gene_BGI[,grepl("B_",colnames(ILM_aceview_gene_BGI))]
rownames(SampleB) <- ILM_aceview_gene_BGI[,2]
```

2. Randomly extract replicates from each sample. We are choosing 6 replicates from Sample A and 8 replicates from Sample B.

```
set.seed(12345)
SampleA3 <- SampleA[,sample(1:80,6)]
SampleB3 <- SampleB[,sample(1:80,8)]
```

3. Combine them into one matrix.

```
datamatrix <- cbind(SampleA3,SampleB3)
```

4. Create limma design matrix

```
design <- c()
for (i in 1:6) {
    design <- c(design, 1)
}
for (i in 1:8) {
    design <- c(design, 2)
}
design <- model.matrix(~ 0+factor(design))
colnames(design) <- c("group1", "group2")
rownames(design) <- colnames(datamatrix)
```

5. Linnorm-limma Normalization + Differential Expression Analysis

- Please note that there is no need to filter low count genes with Linnorm.
- This function is created for users because voom and limma expect raw counts as input. So, input Linnorm normalized data into the limma pipeline will result in wrong log 2 fold change and expression values.

– This function will correct this incompatibility.

a. This is an example that outputs only Differential Expression Analysis Results.

```
library(Linnorm)
DEG_Results <- Linnorm.limma(datamatrix,design)
#The DEG_Results matrix contains DEG analysis results.
```

b. To output both DEG analysis results and the transformed matrix:

```
library(Linnorm)
BothResults <- Linnorm.limma(datamatrix,design,output="Both")

#To separate results into two matrices for analysis:
DEG_Results <- BothResults[[1]]
#Or
DEG_Results <- BothResults$DEResults
#The DEG_Results matrix now contains DEG analysis results.

TransformedMatrix <- BothResults[[2]]
#Or
TransformedMatrix <- BothResults$TMatrix
#The TransformedMatrix matrix now contains a Linnorm Normalized dataset.
```

2.2.1.2 Data analysis

With the results from the previous section, we are going to demonstrate common procedures in analyzing RNA-seq data.

1. Write out the results to a tab delimited file.

```
write.table(DEG_Results, "DEG_Results.txt", quote=F, sep="\t", col.names=TRUE,
row.names=TRUE)
```

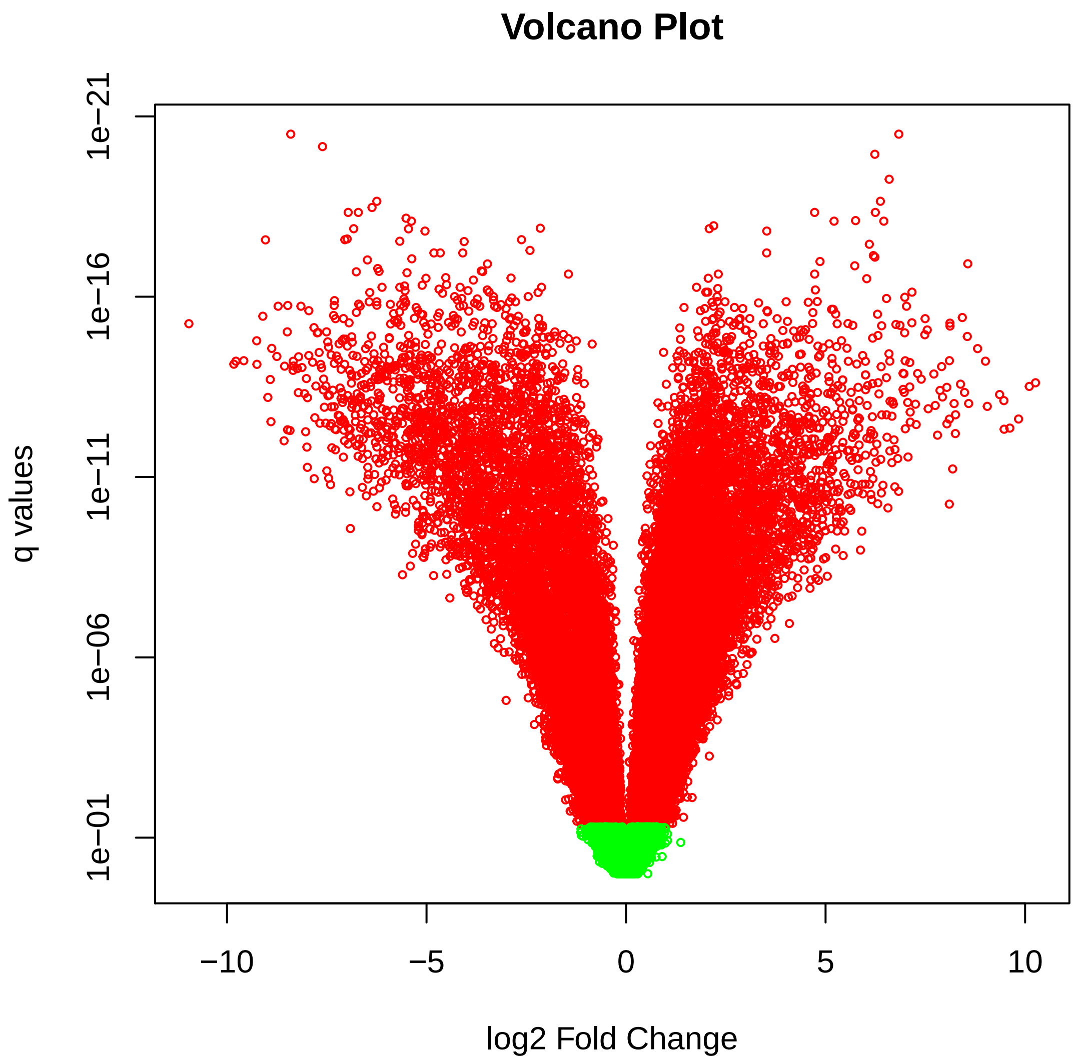
2. Print out the most significant 15 genes.

```
Genes15 <- DEG_Results[order(DEG_Results[,"adj.P.Val"]),][1:15,]
print(Genes15)
```

	logFC	XPM	t	P.Value	adj.P.Val	B
TNR	-8.4022	96.3887	-333.7924	0	0	45.9693
MELK	6.8351	24.2548	329.6666	0	0	45.8984
SPHKAP	-7.6060	55.3862	-299.0662	0	0	45.3081
RHOXF2B	6.2347	15.9251	280.7410	0	0	44.8913
MAGEA6	6.5943	20.4941	242.2300	0	0	43.8156
FAM135B	-6.2456	21.3969	-210.3517	0	0	42.6555
IGL_	6.3735	1201.2170	210.2697	0	0	42.6522
SPYVAW	-6.3662	23.2873	-201.4029	0	0	42.2734
SWOKER	6.2466	16.0593	194.2460	0	0	41.9469
CRTAM	-6.7085	29.5992	-192.0522	0	0	41.8429
LOC150622	-6.9627	35.3580	-190.4060	0	0	41.7635
FN1	4.7254	1585.6061	189.9818	0	0	41.7429
MIYARU	-5.5129	12.7623	-183.1696	0	0	41.4012
UCA1	5.7521	11.3366	179.8159	0	0	41.2255
DNTT	5.2128	7.7338	177.3103	0	0	41.0909

3. Volcano Plot

```
#Remove Genes which fold changes are INF. You can skip this if there is no INF
#values in the fold change column.
NoINF <- DEG_Results[which(!is.infinite(DEG_Results[,1])),]

#Record significant genes for coloring
SignificantGenes <- NoINF[NoINF[,5] <= 0.05,1]

#Draw volcano plot with Log q values.
#Green dots are non-significant, red dots are significant.
plot(x=NoINF[,1], y=NoINF[,5], col=ifelse(NoINF[,1] %in% SignificantGenes,
"red", "green"),log="y", ylim = rev(range(NoINF[,5])), main="Volcano Plot",
xlab="log2 Fold Change", ylab="q values", cex = 0.5)
```



2.2.2 Linnorm Normlization for Parametric Tests

In this section, we are going to perform a normalization on SEQC's Sample A RNA-seq data and perform Gene Correlation Analysis as an example.

2.2.2.1 Linnorm Normalization Here, we will demonstrate how to generate and ouput Linnorm Normalized dataset into a tab delimited file.

1. Get RNA-seq data from Sample A.

```
library(seqc)
SampleA <- ILM_aceview_gene_BGI[,grepl("A_",colnames(ILM_aceview_gene_BGI))]
rownames(SampleA) <- ILM_aceview_gene_BGI[,2]
```

2. Linnorm Normalization

```
library(Linnorm)
#We are only normalizing part of the matrix for an example.
SampleA <- SampleA[,1:20]
Normalized <- Linnorm(SampleA)
```

3. Write out the results to a tab delimited file.

```
#You can use this file with Excel.
write.table(Normalized, "Normalized_Matrix.txt", quote=F, sep="\t",
col.names=TRUE, row.names=TRUE)
```

2.2.2.2 Gene Correlation Analysis In this section, we will use the normalized Gene Expression Matrix from the last section for gene correlation analysis.

1. Filter Genes with more than 25% of the values being zero.

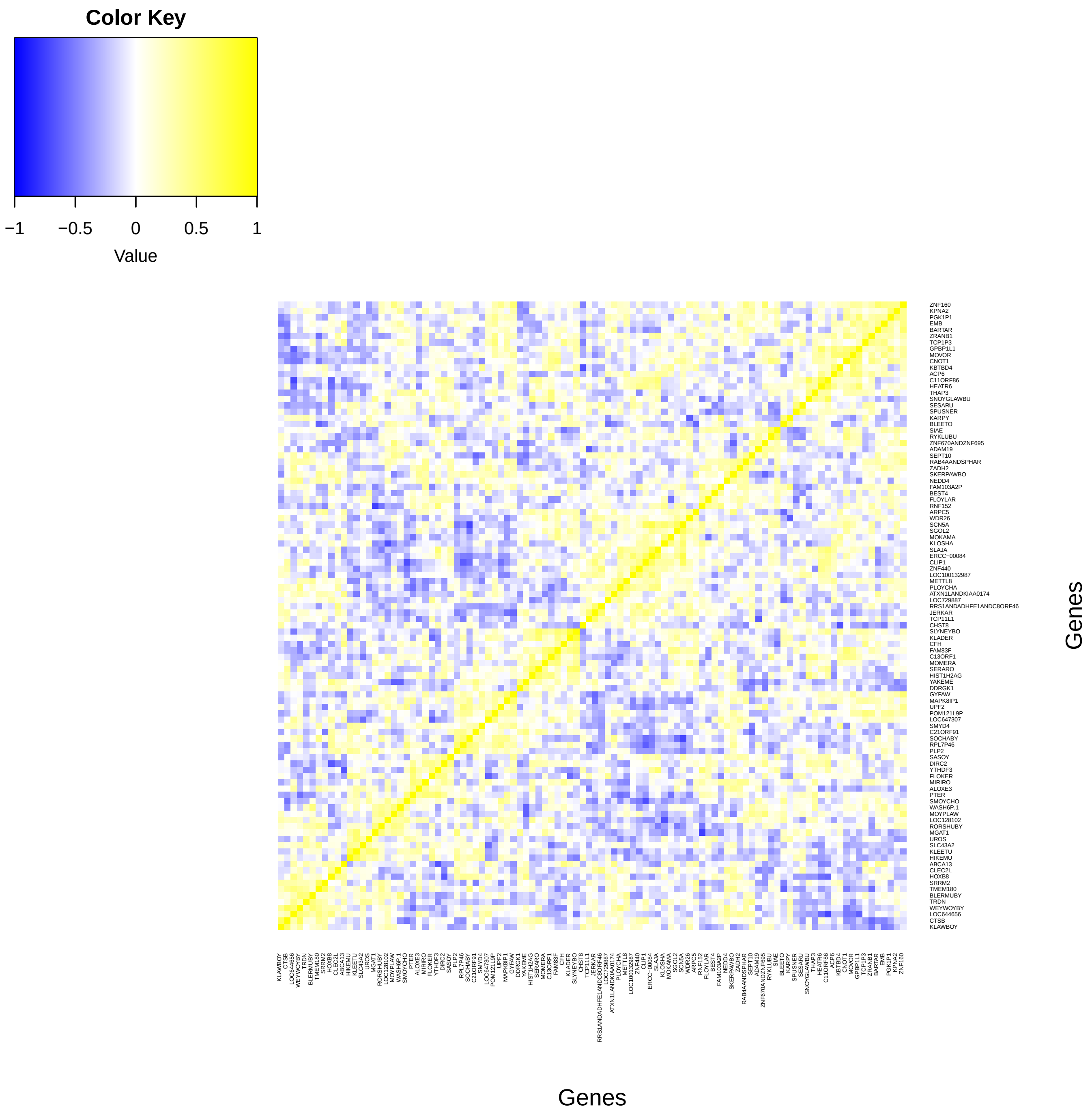
```
Normalized <- Normalized[rowSums(Normalized == 0) <= length(Normalized[1,])/4,]
```

2. Calculate Pearson correlation coefficients between combinations of genes.

```
#Here, we randomly choose 100 genes for an example.
#However, users can provide a custom gene list here.
set.seed(12345)
Normalized <- Normalized[sample(1:length(Normalized[,1]),100),]

#Calculate correlation.
PCC <- cor(t(Normalized))
```

4. Check their correlation with each other using a heatmap.
   - Draw Heatmap

```
#Please install these libraries if you need to.
library(RColorBrewer)
library(gplots)
heatmap.2(as.matrix(PCC), Rowv=TRUE, Colv=TRUE, dendrogram='none',
symbreaks=TRUE , trace="none", xlab = 'Genes', ylab = "Genes",
density.info='none' ,key.ylab=NA, col = colorRampPalette(c("blue",  "white",
"yellow"))(n = 1000), lmat=rbind(c(4, 3), c(2,
1)),cexRow=0.3,cexCol=0.3,margins = c(8, 8))
```



2.2.2.3 Calculate Fold Change Fold change can be calculated by the Linnorm.limma function. It is included in Differential Expression Analysis results. However, for users who would like to calculate fold change from Linnorm normalized dataset and analyze it. Here is an example.

1. Get RNA-seq data from Sample A and B.

```
library(seqc)

SampleA <- ILM_aceview_gene_BGI[,grepl("A_",colnames(ILM_aceview_gene_BGI))]
rownames(SampleA) <- ILM_aceview_gene_BGI[,2]
```

```
SampleB <- ILM_aceview_gene_BGI[,grepl("B_",colnames(ILM_aceview_gene_BGI))]
rownames(SampleB) <- ILM_aceview_gene_BGI[,2]
```

2. Combine them into one matrix.

```
datamatrix <- cbind(SampleA,SampleB)
```

4. Linnorm Normalization.

```
library(Linnorm)
LNormData <- Linnorm(datamatrix)
```

5. Undo Logarithm from LNormData.

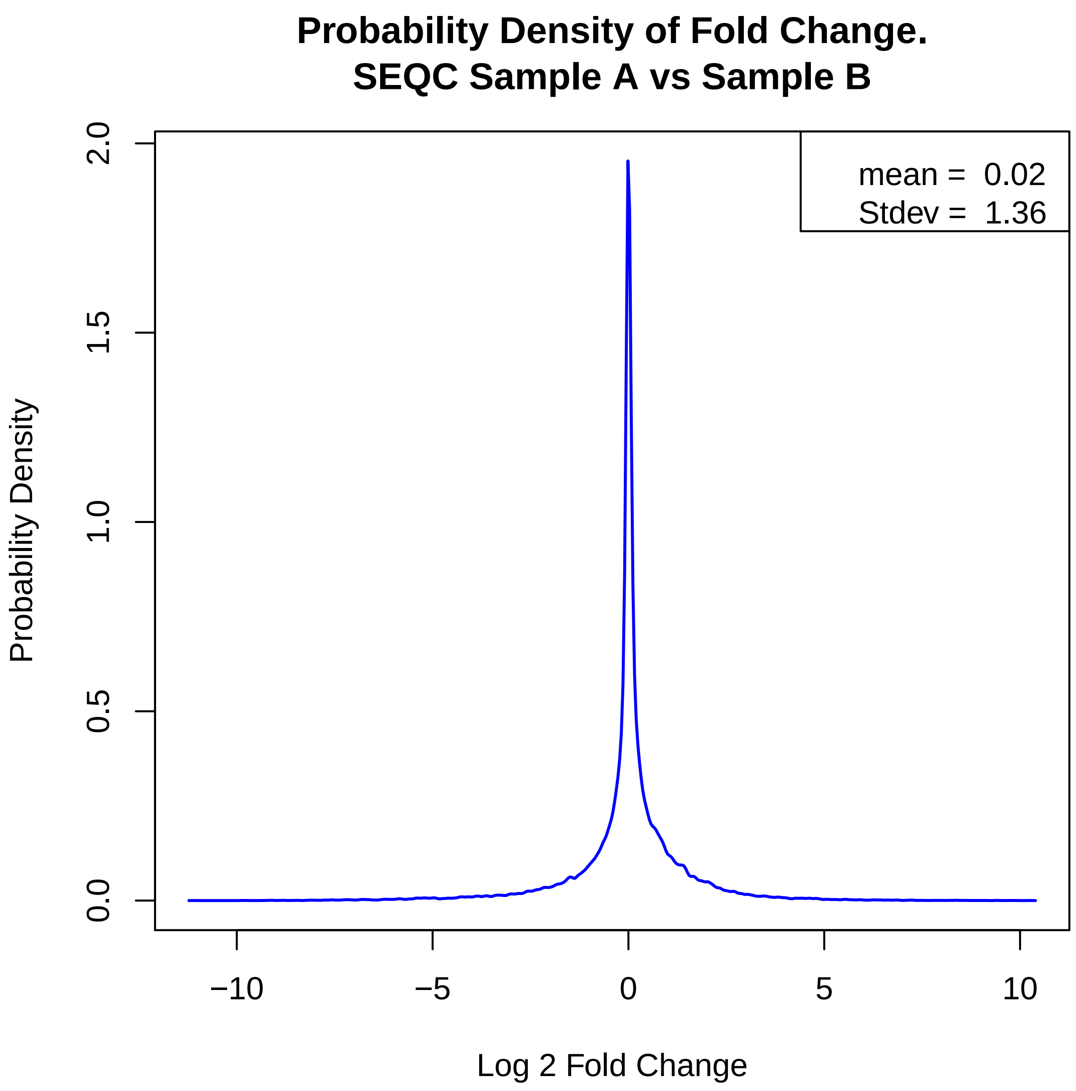
```
#Let LNormData be a matrix of Linnorm Normalized dataset.
Newdata <- exp(LNormData)
```

6. Calculate Fold Change.

```
#Now, we can calculate fold changes between sample set 1 and sample set 2.
#Index of sample set 1 from LNormData and Newdata:
set1 <- 1:80
#Index of sample set 2 from LNormData and Newdata:
set2 <- 81:160

#Define a function that calculates log 2 fold change:
log2fc <- function(x) {
return(log(mean(x[set1])/mean(x[set2]),2))
}

#Calculate log 2 fold change of each gene in the dataset:
foldchanges <- unlist(apply(Newdata,1,log2fc))

#Put resulting data into a matrix
FCMatrix <- matrix(nrow=length(foldchanges),ncol=1)
rownames(FCMatrix) <- rownames(LNormData)
colnames(FCMatrix) <- c("Log 2 Fold Change")
FCMatrix[,1] <- foldchanges

#Now FCMatrix contains fold change results.
```

7. Draw a probability density plot of the fold changes in the dataset.

```
Density <- density(foldchanges)
plot(Density$x,Density$y,type="n",xlab="Log 2 Fold Change", ylab="Probability Density",)
lines(Density$x,Density$y, lwd=1.5, col="blue")
title("Probability Density of Fold Change.\nSEQC Sample A vs Sample B")
legend("topright",legend=paste("mean = ", round(mean(foldchanges),2),
"\nStdev = ", round(sd(foldchanges),2)))
```


Probability Density of Fold Change.
SEQC Sample A vs Sample B
mean = 0.02
Stdev = 1.36
Probability Density
0.0
0.5
1.0
1.5
2.0
−10
−5
0
5
10
Log 2 Fold Change

2.3 RnaXSim

2.3.1 RNA-seq Raw Count Simulation

2.3.1.1 Default In this section, we will run RnaXSim with default settings as a demonstration.

1. Get RNA-seq data from Sample A.

```
library(seqc)

SampleA <- ILM_aceview_gene_BGI[,grepl("A_",colnames(ILM_aceview_gene_BGI))]
rownames(SampleA) <- ILM_aceview_gene_BGI[,2]
```

2. Simulate an RNA-seq dataset.

```
library(Linnorm)
#This will generate two sets of RNA-seq data with 3 replicates each.
#It will have 20000 genes totally with 5000 genes being differentially
#expressed. It has the Poisson distribution.
SimulatedData <- RnaXSim(SampleA)
```

3. Separate data into matrices and vectors as an example.

```
#Index of differentially expressed genes.
DE_Index <- SimulatedData[[2]]

#Expression Matrix
ExpMatrix <- SimulatedData[[1]]

#Sample Set 1
Sample1 <- ExpMatrix[,1:3]

#Sample Set 2
Sample2 <- ExpMatrix[,4:6]
```

2.3.1.2 Advanced In this section, we will show an example where RnaXSim is run with customized settings.

1. Get RNA-seq data from Sample A.

```
library(seqc)

SampleA <- ILM_aceview_gene_BGI[,grepl("A_",colnames(ILM_aceview_gene_BGI))]
rownames(SampleA) <- ILM_aceview_gene_BGI[,2]
```

2. Prepare custom parameters for the simulation.

```
#Number of replicates for each sample set.
NumReplicates <- 5
#Number of Genes in the Samples.
NumGenes <- 5000
#Number of Differentially Expressed Genes.
NumDiffGenes <- 1000
#Distribution. Put "NB" for Negative Binomial, "Gamma" for Gamma,
#"Poisson" for Poisson and "LogNorm" for Log Normal distribution.
Distribution <- "Gamma"
```

3. Simulate an RNA-seq dataset using the above parameters.

```
library(Linnorm)
SimulatedData <- RnaXSim(SampleA, distribution=Distribution,
NumRep=NumReplicates, NumDiff = NumDiffGenes, NumFea = NumGenes)
```

4. Separate data into matrices and vectors for further usage.

```
#Index of differentially expressed genes.
DE_Index <- SimulatedData[[2]]

#Expression Matrix
ExpMatrix <- SimulatedData[[1]]

#Sample Set 1
Sample1 <- ExpMatrix[,1:3]

#Sample Set 2
Sample2 <- ExpMatrix[,4:6]
```

3 Frequently Asked Questions

3.1 How do I convert Linnorm Normalized dataset back to CPM/TPM?

Answer: Here is an example:

```
#Obtain a normalized dataset for an example.
library(seqc)
SampleA <- ILM_aceview_gene_BGI[,grepl("A_",colnames(ILM_aceview_gene_BGI))]
rownames(SampleA) <- ILM_aceview_gene_BGI[,2]
SampleA <- SampleA[,1:4]
library(Linnorm)
LNormData <- Linnorm(SampleA)

#LNormData is a matrix of Linnorm Normalized dataset.
#XPMdata is a CPM or TPM matrix.
XPMdata <- ((exp(LNormData) - 1)/mean(colSums(LNormData))) * 1000000
#Now, XPMdata contains a CPM dataset if the original data is raw count or CPM.
#It contains a TPM dataset if the original data is RPKM, FPKM or TPM.


#On the other hand, if LNormData was filtered or changed in some unknown ways;
# and that there is no way to obtain the original dataset for the calculation
#of CPM or TPM. Then, you can convert it this way:
XPMdata <- exp(LNormData) - 1
for (i in seq_along(XPMdata[1,])) {
    XPMdata[,i] <- (XPMdata[,i]/sum(XPMdata[,i])) * 1000000
}
#Now, XPMdata contains a CPM dataset if the original data is raw count or CPM.
#It contains a TPM dataset if the original data is RPKM, FPKM or TPM.
```

3.2 Can I use Linnorm Normalized dataset to calculate Fold Change?

Answer: Linnorm Normalized dataset is a log transformed dataset. You should not use it to calculate fold change directly. To do it correctly, please refer to the calculate fold change section.

3.3 I only have 1 replicate for each sample set. Can I perform Linnorm Normalization?

Answer: Yes, you can.

3.4 I only have 1 replicate for each sample set. Can I perform Differential Expression Analysis with Linnorm and limma?

Answer: No, linear model based methods must have replicates. So, limma wouldn't work.

3.5 There are a lot of fold changes with INF values in Linnorm.limma output. Can I convert them into numerical units like those in the voom-limma pipeline?

Answer: Since the expression in one set of sample can be zero, while the other can be otherwise, it is mathematically correct to generate INFs. However, it is possible for Linnorm.limma function to prevent generating INFs by setting the noINF argument as TRUE.

4 Bug Reports, Questions and Suggestions

We welcome any Bug Reports, Questions and Suggestions. They can be sent to Ken Yip at `shunyip@bu.edu`. Please add the keyword Linnorm in the email's subject line or title. We appreciate your help in making Linnorm better. Thanks!