Linnorm User Manual

Shun H. Yip[1,2,3], Panwen Wang[3], Jean-Pierre Kocher[3], Pak Chung Sham[1,4,5], Junwen Wang[3,6]

[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 Health Sciences Research and Center for Individualized Medicine, Mayo Clinic, Scottsdale, AZ, 85259, USA;
[4] Department of Psychiatry, LKS Faculty of Medicine, The University of Hong Kong, Hong Kong SAR, China;
[5] State Key Laboratory in Cognitive and Brain Sciences, The University of Hong Kong, Hong Kong SAR, China;
[6] Department of Biomedical Informatics, Arizona State University, Scottsdale, AZ, 85259, USA.

Abstract

Linnorm is an R package for the analysis of RNA-seq, scRNA-seq, ChIP-seq count data or any large scale count data. Its main function is to normalize and transform such datasets for parametric tests. Examples of parametric tests include using *limma* for differential expression analysis or differential peak detection[1] or calculating Pearson correlation coefficient for gene correlation study. Linnorm can work with raw count, CPM, RPKM, FPKM and TPM and is compatible with data generated from simple count algorithms[2] and supervised learning algorithms[3].

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.

[1] The Linnorm-limma pipeline is implemented as the "Linnorm.limma" function. Please cite both Linnorm and limma if you use this function for publication.

[2] Such as HTSeq, *Rsubread* and etc

[3] Such as Cufflinks, eXpress, RSEM, Sailfish, and etc

Contents

1 Introduction

Linnorm is a count data transformation method. Linnorm transforms the dataset toward both homoscedasticity and normality; and it only has one transformation parameter for the whole dataset. This approach ensures that the resulting dataset can better satisfy the assumptions made by the linear models, which is used by *limma*. Furthermore, since Linnorm is actively transforming the dataset toward homoscedasticity and normality, it performs well with datasets which may be deviated from the negative binomial distribution, such as those generated by supervised learning based gene quantification software. Additionally, since homoscedasticity and normality are also assumed by many other parametric tests, such as the Pearson correlation coefficient, Linnorm transformed datasets are not restricted to differential expression analysis.

The Linnorm R package contains a variety of functions for RNA-seq data analysis. It has three functions.

- RNA-seq Expression Normalization/Transformation (Linnorm)
  - Output transformed data matrix for analysis
  - Gene Correlation Analysis
- The Linnorm-limma pipeline (Linnorm.limma)
  - Differential expression analysis
  - Differential peak detection
- RNA-seq Raw Count Simulation (RnaXSim)

1.1 Linnorm

1.1.1 Datatypes and Input Format

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

- Raw Count (RNA-seq or ChIP-seq)
- 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)

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	...

	Sample 1	Sample 2	Sample 3	...
...	...	...	...	...
...	...	...	...	...

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. The *seqc* package is a good source of such datasets.

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 both Linnorm and limma if you use the Linnorm.limma function for differential expression analysis for publication.

2.2 Linnorm

2.2.1 Linnorm-limma Differential Expression Analysis

2.2.1.1 Obtain example data

Please skip this step if a data matrix is already available.

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)
```

2.2.1.2 Analysis procedure

The Linnorm-limma pipeline only consists of two steps.

1. Create limma design matrix

```
#6 samples for condition 1 and 8 samples for condition 2.
design <- c(rep(1,6),rep(2,8))
design <- model.matrix(~ 0+factor(design))
colnames(design) <- c("group1", "group2")
rownames(design) <- colnames(datamatrix)
```

2. Linnorm-limma Differential Expression Analysis

- Please cite both Linnorm and *limma* if you are using this function for publication.

a. Basic Differential Expression Analysis. (Follow this if you are not sure what to do.)

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

b. Advanced: 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$DEResults
#The DEG_Results matrix now contains DEG analysis results.

TransformedMatrix <- BothResults$TMatrix
#The TransformedMatrix matrix now contains a Linnorm Transformed dataset.
```

2.2.1.3 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=FALSE, sep="\t", col.names=TRUE,
row.names=TRUE)
```

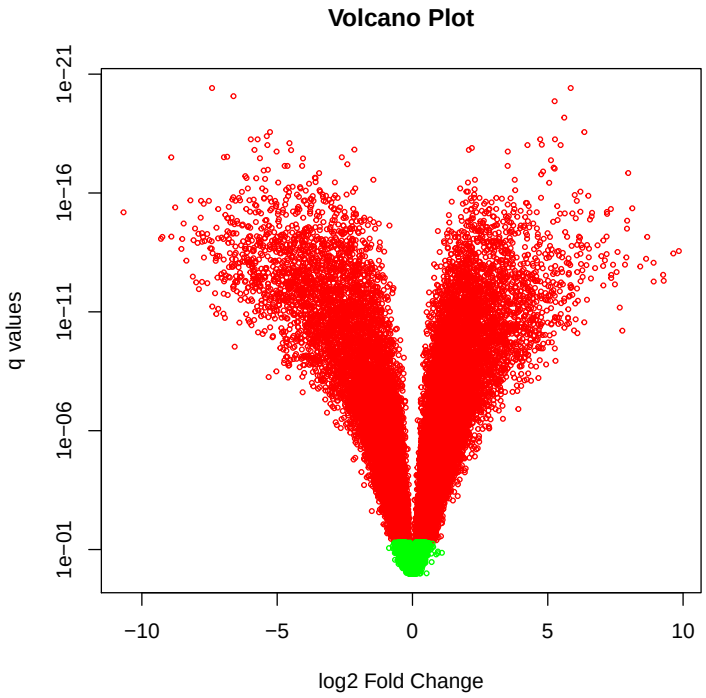
2. Print out the most significant 15 genes.

```
Genes15 <- DEG_Results[order(DEG_Results[,"adj.P.Val"]),][1:15,]
#Users can print the gene list by the following command:
#print(Genes15)
```

	logFC	XPM	t	P.Value	adj.P.Val	B
TNR	-7.4067	96.3887	-333.1798	0	0	45.4809
MELK	5.8479	24.2548	329.0665	0	0	45.4139
SPHKAP	-6.6136	55.3862	-298.5171	0	0	44.8537
RHOXF2B	5.2539	15.9251	280.2283	0	0	44.4569
MAGEA6	5.6094	20.4941	241.7843	0	0	43.4277
FAM135B	-5.2647	21.3969	-209.9637	0	0	42.3110
IGL_	6.3522	1201.2170	209.8817	0	0	42.3077
SPYVAW	-5.3838	23.2873	-201.0308	0	0	41.9417
SWOKER	5.2657	16.0593	193.8870	0	0	41.6257
CRTAM	-5.7224	29.5992	-191.6967	0	0	41.5249
LOC150622	-5.9744	35.3580	-190.0533	0	0	41.4480
FN1	4.7202	1585.6061	189.6332	0	0	41.4282
MIYARU	-4.5444	12.7623	-182.8317	0	0	41.0967
UCA1	4.7788	11.3366	179.4836	0	0	40.9261
DNTT	4.2514	7.7338	176.9834	0	0	40.7953

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)
```


Volcano Plot
q values
1e−21
1e−16
1e−11
1e−06
1e−01
−10
−5
0
5
10
log2 Fold Change

2.2.2 Linnorm Transformation for Parametric Tests

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

2.2.2.1 Linnorm Transformation

Here, we will demonstrate how to generate and ouput Linnorm Transformed 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]
#We are only transforming part of the matrix for an example.
SampleA <- SampleA[,1:10]
```

2. Linnorm Transformation

```
library(Linnorm)
Transformed <- Linnorm(SampleA)
```

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

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

2.2.2.2 Gene Correlation Analysis

In this section, we will use the transformed Gene Expression Matrix from the last section for gene correlation analysis.

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

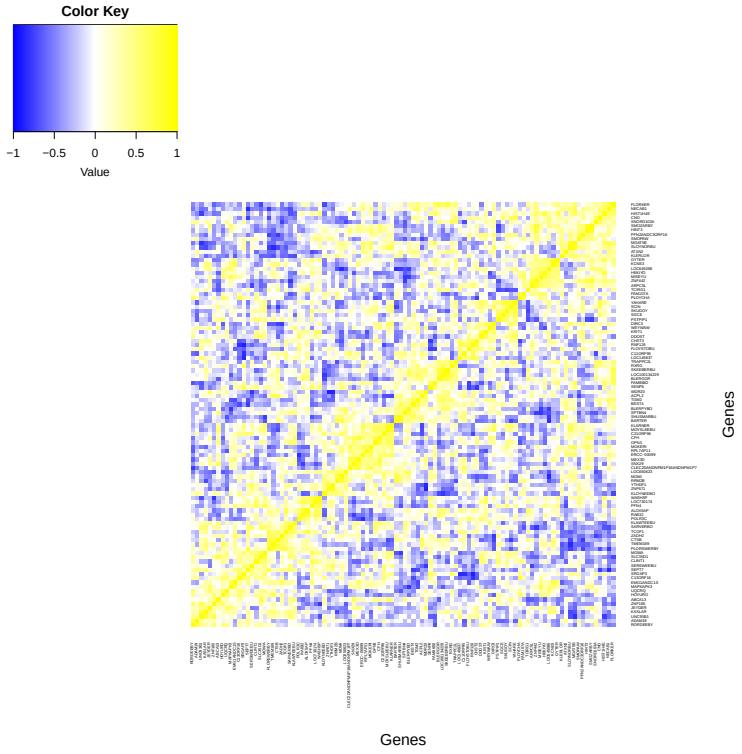
```
Transformed <- Transformed[rowSums(Transformed == 0) <= length(Transformed[1,])/4,]
```

2. Calculate Pearson correlation coefficients between combinations of genes.

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

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

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

```
#Please install these libraries if you need to.
library(RColorBrewer)
library(gplots)
##
## Attaching package: 'gplots'
## The following object is masked from 'package:stats':
##
##     lowess
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 transformed 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.

```
#Again, we are only using part of the matrices for an example.
datamatrix <- cbind(SampleA[,1:5],SampleB[,1:5])
```

4. Linnorm Transformation.

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

5. Undo Logarithm from LNormData.

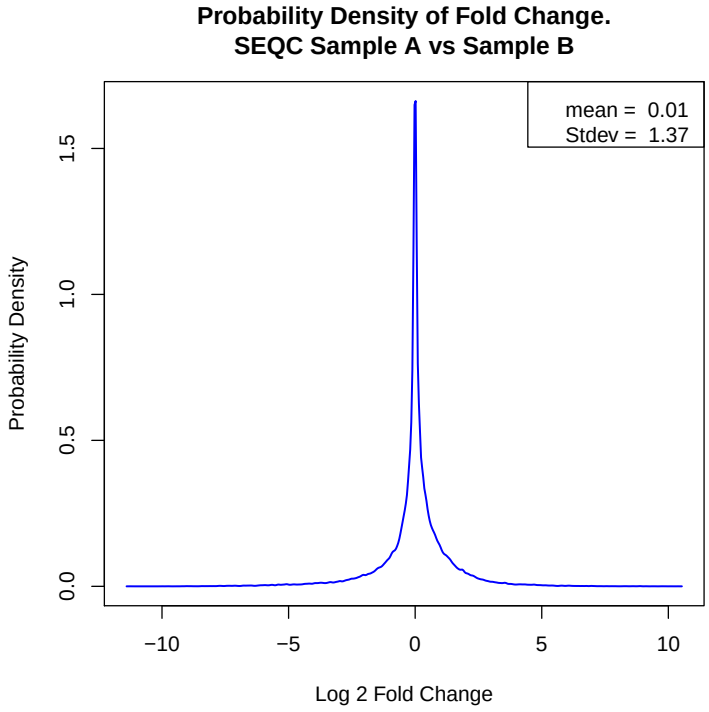
```
#Let LNormData be a matrix of Linnorm Transformed dataset.
Newdata <- exp(LNormData)
```

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 <- 1:5
#Index of sample set 2 from LNormData and Newdata:
set2 <- 6:10

#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)))
```



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]
#We are only using part of the matrix for an example.
#However, users are encouraged to use the whole matrix.
SampleA <- SampleA[,1:10]
```

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]

#We are only using part of the matrix for an example.
#However, users are encouraged to use the whole matrix.
SampleA <- SampleA[,1:10]
```

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 Transformed dataset back to CPM/TPM?

Answer: Here is an example:

```
#Obtain a transformed 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 is a matrix of Linnorm Transformed dataset.
LNormData <- Linnorm(SampleA)

#To convert Linnorm Transformed dataset into CPM or TPM:
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 Transformed dataset to calculate Fold Change?

Answer: Linnorm Transformed 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 Transformation?

Answer: Yes, you can. However, replicates are needed for a more accurate transformation.

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 arithmetically correct to generate INFs. However, it is possible for the Linnorm.limma function to prevent generating INFs by setting the noINF argument as TRUE.

3.6 During installation, an error says that Linnorm is not available. What is the problem?

Answer: Linnorm requires Bioconductor 3.3. You can upgrade Bioconductor by removing it with, "remove.packages("BiocInstaller")", then restart R session and install it again with, "source("http://bioconductor.org/biocLite.R")". Please also update R accordingly for Bioconductor 3.3 if needed.

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!