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 scRNA-seq, RNA-seq, ChIP-seq count data or any large scale count data. Its main function is to normalize and transform such datasets for parametric tests. Various analysis pipelines are also implemented for users' convenience, including using *limma* for differential expression analysis or differential peak detection[1], co-expression network analysis[2], subpopulation analysis pipeline with t-SNE K-means clustering[3], PCA K-means clustering[4] and hierarchical clustering analysis[5], highly variable gene discovery[6] and data imputation[7]. Linnorm can work with raw count, CPM, RPKM, FPKM and TPM and is compatible with data generated from simple count algorithms[8] and supervised learning algorithms[9]. Additionally, the RnaXSim function is included for simulating RNA-seq data for the evaluation of DEG analysis methods.

[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]Implemented as the Linnorm.Cor function.
[3]Implemented as the Linnorm.tSNE function.
[4]Implemented as the Linnorm.PCA function.
[5]Implemented as the Linnorm.HClust function.
[6]Implemented as the Linnorm.HVar function.
[7]Implemented as the Linnorm.DataImput function.
[8]Such as HTSeq, *Rsubread* and etc
[9]Such as Cufflinks, eXpress, Kallisto, RSEM, Sailfish, and etc

Contents

1 Introduction

Linnorm is a normalization and transformation method.

The Linnorm R package contains a variety of functions for single cell RNA-seq data analysis by utilizing Linnorm.

- scRNA-seq Expression Normalization/Transformation
  - Output transformed data matrix for analysis (Linnorm)
  - or output normalized data matrix for analysis (Linnorm.Norm)
- The Linnorm-limma pipeline (Linnorm.limma)
  - Differential expression analysis
  - Differential peak detection
- t-SNE k-means clustering (Linnorm.tSNE)
  - Single cell RNA-seq subpopulation analysis
- PCA k-means clustering (Linnorm.PCA)
  - Single cell RNA-seq subpopulation analysis
- Hierarchical clustering analysis (Linnorm.HClust)
- Highly variable gene discovery (Linnorm.HVar)
- Coexpression network analysis (Linnorm.Cor)
- Data Imputation (Linnorm.DataImput)
- RNA-seq data simulation for negative binomial, Poisson, log normal or gamma distribution. (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 (scRNA-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)

We suggest RPKM, FPKM or TPM for most purposes.

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	...
...	...	...	...	...
...	...	...	...	...

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

By using the argument, "input ="Linnorm"", several functions provided by the Linnorm package can also accept Linnorm transformed datasets as input.

2 Examples with Source Codes

2.1 Linnorm

2.1.1 Basic Linnorm Transformation

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

1. Linnorm's normalizing transformation

```
library(Linnorm)
data(Islam2011)
#Do not filter gene list:
Transformed <- Linnorm(Islam2011)

#Filter low count genes
FTransformed <- Linnorm(Islam2011, keepAll=FALSE)
```

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

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

#This file filtered low count genes.
write.table(FTransformed, "Transformed.txt",
quote=FALSE, sep="\t", col.names=TRUE, row.names=TRUE)
```

2.1.2 Linnorm-limma Differential Expression Analysis

- 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.

2.1.2.1 Obtain example data

1. Get 20 samples of RNA-seq data. These 20 samples are paired tumor and adjacent normal tissue samples from 10 individuals from TCGA LIHC dataset.

```
library(Linnorm)
data(LIHC)
datamatrix <- LIHC
```

2.1.2.2 Analysis procedure

The Linnorm-limma pipeline only consists of two steps.

1. Create limma design matrix

```
#10 samples for condition 1 and 10 samples for condition 2.
#You might need to edit this line
design <- c(rep(1,10),rep(2,10))
#There lines can be copied directly.
design <- model.matrix(~ 0+factor(design))
colnames(design) <- c("group1", "group2")
rownames(design) <- colnames(datamatrix)
```

2. Linnorm-limma Differential Expression Analysis

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

```
library(Linnorm)
#The Linnorm-limma pipeline only consists of one line.
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 for further analysis

```
library(Linnorm)
#Just add output="Both" into the argument list
BothResults <- Linnorm.limma(datamatrix,design,output="Both")

#To separate results into two matrices:
DEG_Results <- BothResults$DEResults
TransformedMatrix <- BothResults$Linnorm
#The DEG_Results matrix now contains DEG analysis results.
#The TransformedMatrix matrix now contains a Linnorm
#Transformed dataset.
```

2.1.2.3 Print out the most significant genes

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 10 genes.

```
Genes10 <- DEG_Results[order(DEG_Results[,"adj.P.Val"]),][1:10,]
#Users can print the gene list by the following command:
#print(Genes10)

#logFC: log 2 fold change of the gene.
#XPM: If input is raw count or CPM, XPM is CPM.
#   If input is RPKM, FPKM or TPM, XPM is TPM.
#t: moderated t statistic.
#P.Value: p value.
#adj.P.Val: Adjusted p value. This is also called False Discovery Rate or q value.}
#B: log odds that the feature is differential.
```

	logFC	XPM	t	P.Value	adj.P.Val	B
SFTA1P|207107	4.4481	0.2135	15.5774	0	0e+00	18.050
MESP1|55897	3.3216	2.1546	10.2545	0	0e+00	11.689
PKMYT1|9088	4.1229	1.5978	9.9596	0	0e+00	11.221
CDKN3|1033	4.7182	5.9590	9.5544	0	0e+00	10.525
HOXD9|3235	5.4774	0.8435	9.0194	0	1e-04	9.629
HOXD1|3231	2.5451	0.0657	8.9166	0	1e-04	9.476
CLEC1B|51266	-3.3694	13.9090	-8.8467	0	1e-04	9.341
KIF4A|24137	4.2165	0.9223	8.6877	0	1e-04	9.105
CDC45|8318	4.5057	1.4721	8.5908	0	1e-04	8.908
IGF2AS|51214	-3.0068	0.1443	-8.5298	0	1e-04	8.799

2.1.2.4 Volcano Plot

1. Remove Genes which fold changes are INF. You can skip this if there are no INF values in the fold change column.

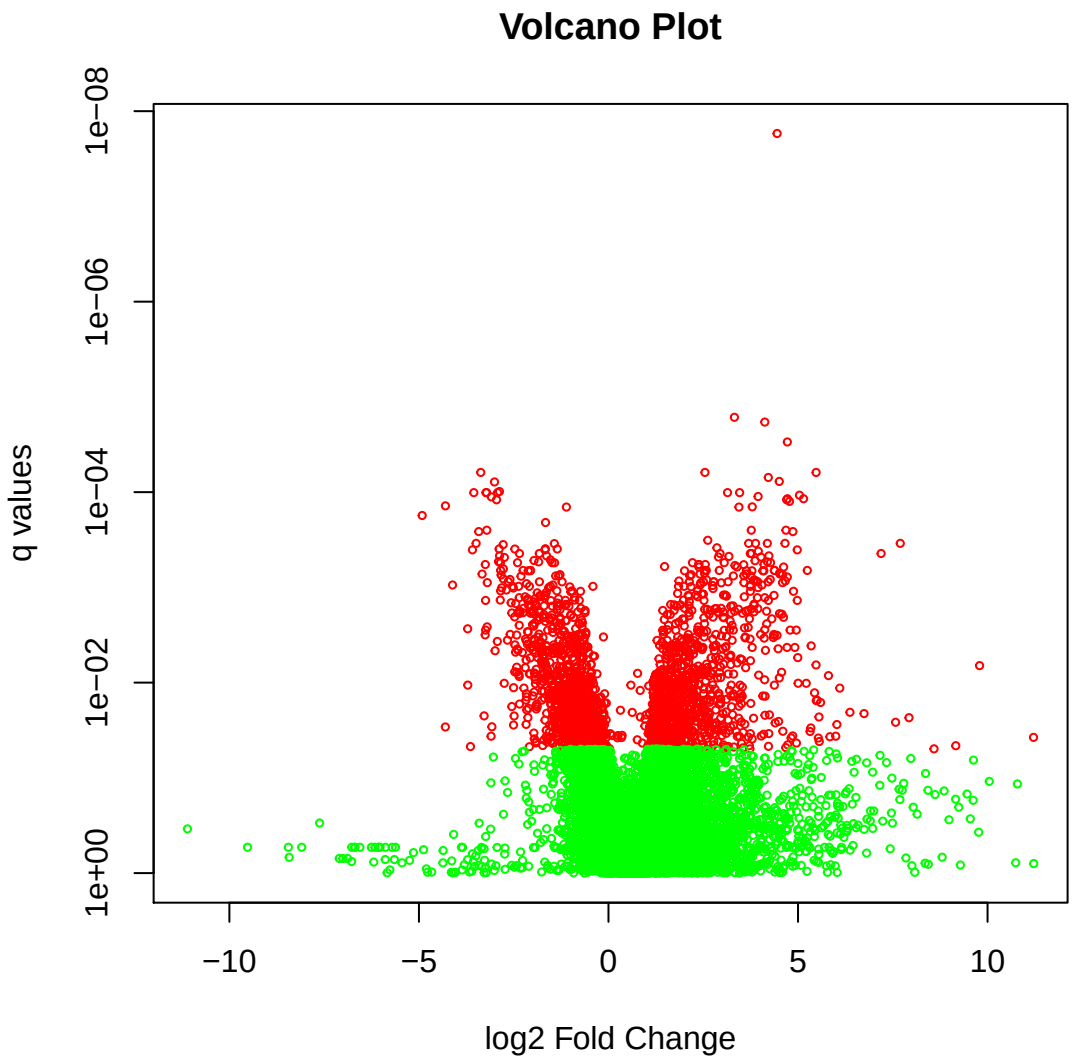
```
NoINF <- DEG_Results[which(!is.infinite(DEG_Results[,1])),]
```

2. Record significant genes for coloring

```
SignificantGenes <- NoINF[NoINF[,5] <= 0.05,1]
```

3. 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)
```



2.1.3 Single cell RNA-seq DEG Analysis

In this section, we use Islam et al. (2011)'s single cell RNA-seq dataset to perform DEG analysis. In this analysis, we are using 48 mouse embryonic stem cells and 44 mouse embryonic fibroblasts.

Read data:

```
library(Linnorm)
data(Islam2011)
IslamData <- Islam2011[,1:92]
```

1. Create limma design matrix

```
#48 samples for condition 1 and 44 samples for condition 2.
#You might need to edit this line
design <- c(rep(1,48),rep(2,44))
#There lines can be copied directly.
design <- model.matrix(~ 0+factor(design))
colnames(design) <- c("group1", "group2")
rownames(design) <- colnames(IslamData)
```

2. DEG Analysis

```
#Example 1: Filter low count genes.
#and genes with high technical noise.
scRNAseqResults <- Linnorm.limma(IslamData,design,keepAll=FALSE)


#Example 2: Do not filter gene list.
scRNAseqResults <- Linnorm.limma(IslamData,design)
```

2.1.4 PCA K-means Clustering. Cell subpopulation analysis

In this section, we use Islam et al. (2011)'s single cell RNA-seq dataset to perform subpopluation analysis. The 96 samples are consisted of 48 mouse embryonic stem cells, 44 mouse embryonic fibroblasts and 4 negative controls. We do not use the negative controls here. This section is basically identical to the t-SNE K-means Clustering section above.

1. Read data.

```
library(Linnorm)
data(Islam2011)
```

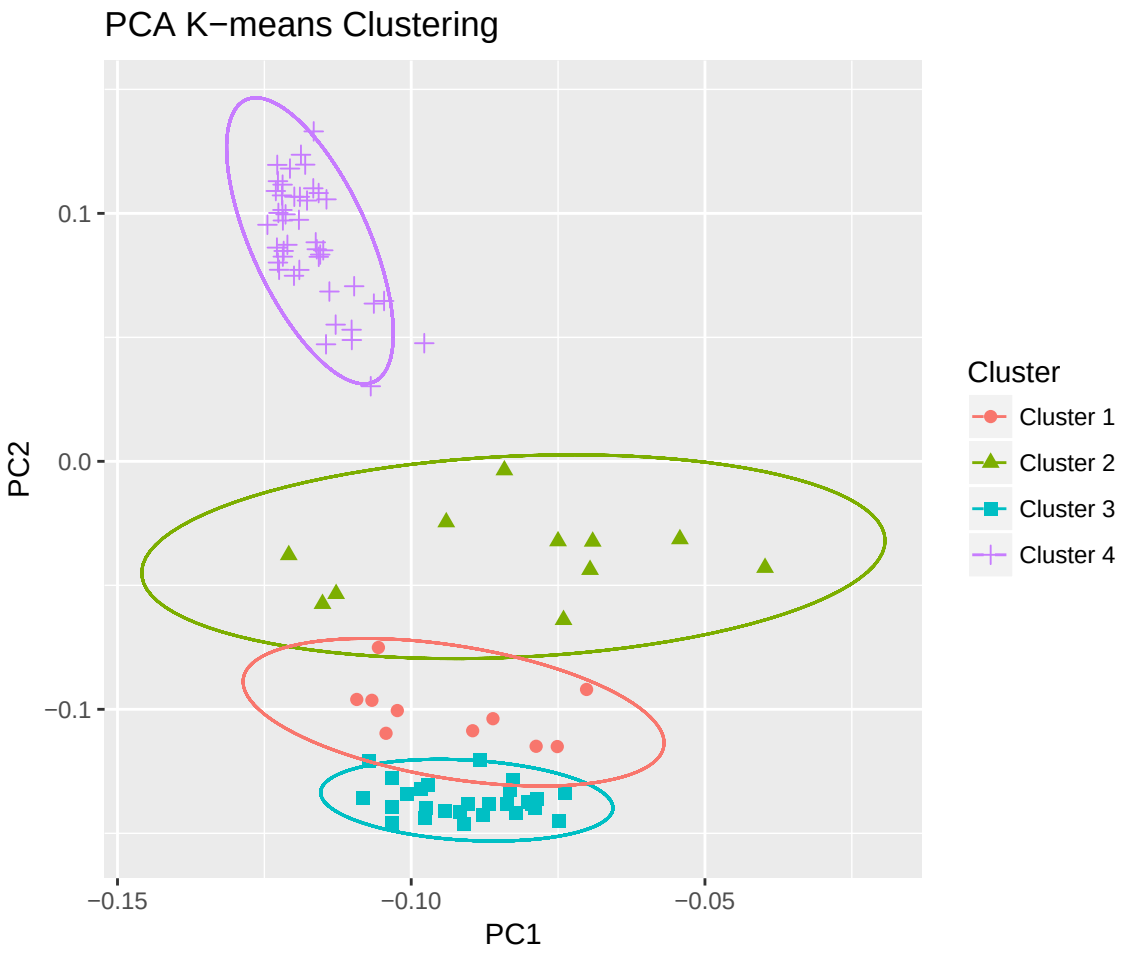
2.1.4.1 Simple subpopulation analysis

1. Perform PCA analysis using default configurations. Only keep genes with less than half of the samples being zero.

```
PCA.results <- Linnorm.PCA(Islam2011[,1:92])
```

2. Draw PCA k-means clustering plot.

```
#Here, we can see multiple clusters.
print(PCA.results$plot$plot)
```



2.1.4.2 Analysis with known subpopulations

1. Assign known groups to samples.

```
#The first 48 samples belong to mouse embryonic stem cells.
Groups <- rep("ES_Cells",48)
#The next 44 samples are mouse embryonic fibroblasts.
Groups <- c(Groups, rep("EF_Cells",44))
```

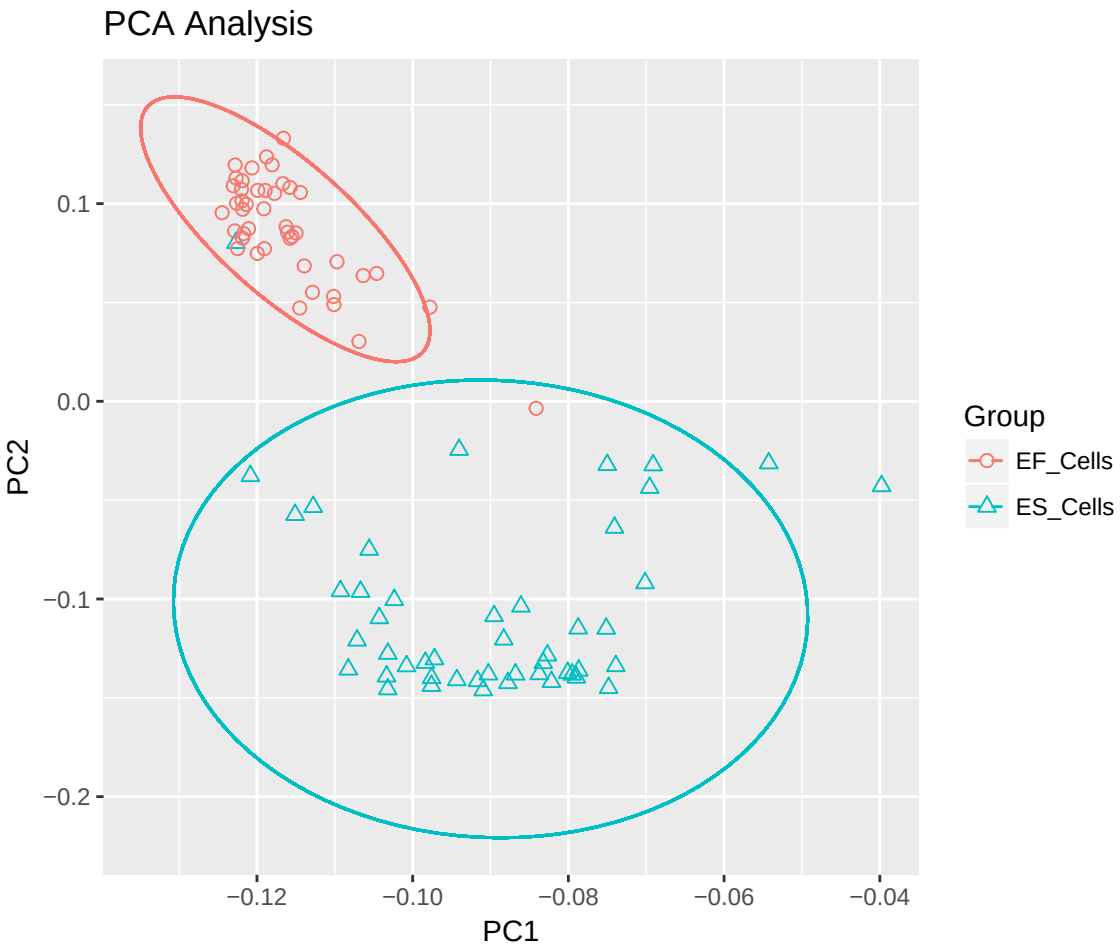
2. Perform PCA analysis.

```
#Useful arguments:
#Group:
#allows user to provide each sample's information.
#num_center:
#how many clusters are supposed to be there.
#num_PC
#how many principal components should be used in k-means
```

```
#clustering.

PCA.results <- Linnorm.PCA(Islam2011[,1:92],
Group=Groups, num_center=2, num_PC=3)
```

3. Draw PCA k-means clustering plot.

```
#Here, we can see two clusters.
print(PCA.results$plot$plot)
```



2.1.5 Gene Co-expression Network Analysis

In this section, we are going to use Islam2011 single cell RNA-seq embryonic stem cell data and perform Gene Correlation Analysis.

1. Obtain data.

```
data(Islam2011)

#Obtain embryonic stem cell data
datamatrix <- Islam2011[,1:48]
```

2. Perform analysis.

```
#Setting plotNetwork to TRUE will create a figure file in your current directory.
#Setting it to FALSE will stop it from creating a figure file, but users can plot it
#manually later using the igraph object in the output.
#For this vignette, we will plot it manually in step 4.
results <- Linnorm.Cor(datamatrix, plotNetwork=FALSE,
#Edge color when correlation is positive
plot.Pos.cor.col="red",
#Edge color when correlation is negative
plot.Neg.cor.col="green")
```

3. Print out the most significant 10 pairs of genes.

```
Genes10 <- results$Results[order(results$Results[,5]),][1:10,]
#Users can print the gene list by the following command:
#print(Genes10)
```

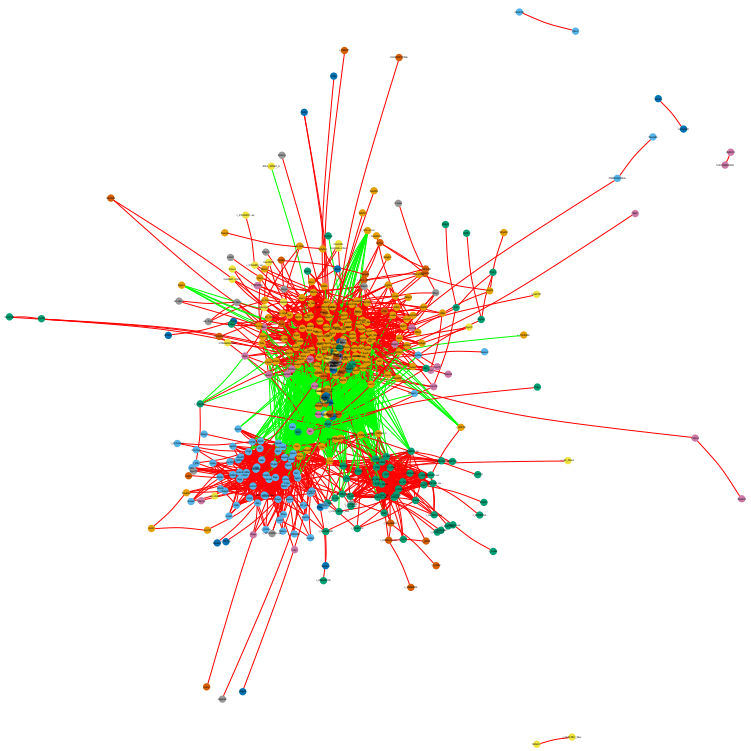
Gene1	Gene2	Cor	p.value	q.value
Rpl26	Gm15772	0.9994	0	0
Ftl2	Ftl1	0.9990	0	0
Oaz1	Gm9786	0.9970	0	0
Rps26	Gm6654	0.9887	0	0
r_(T)n	r_(A)n	0.9876	0	0
Hnrnpa1	Gm5643	0.9869	0	0
Gm15427	Rpl18a	0.9841	0	0
Bex4	Bex1	0.9830	0	0
r_B2_Mm1t	r_B2_Mm1a	0.9810	0	0
RNA_SPIKE_1	RNA_SPIKE_2	0.9789	0	0

2.1.5.1 Plot a co-expression network

We will demonstrate how to plot the figure "networkplot.png" here.

```
library(igraph)
##
## Attaching package: 'igraph'
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
## The following object is masked from 'package:base':
##
##     union
Thislayout <- layout_with_kk(results$igraph)
plot(results$igraph,
```

```
#Vertex size
vertex.size=2,
#Vertex color, based on clustering results
vertex.color=results$Cluster$Cluster,
#Vertex frame color
vertex.frame.color="transparent",
#Vertex label color (the gene names)
vertex.label.color="black",
#Vertex label size
vertex.label.cex=0.05,
#This is how much the edges should be curved.
edge.curved=0.1,
#Edge width
edge.width=0.05,
#Use the layout created previously
layout=Thislayout
)
```



2.1.5.2 Identify genes that belong to a cluster

For example, what are the genes that belong to the same cluster as the Mmp2 gene?

1. Identify the cluster that Mmp2 belongs to.

```
TheCluster <- which(results$Cluster[,1] == "Mmp2")
```

2. Obtain the list of genes that belong to this cluster.

```
#Index of the genes
ListOfGenes <- which(results$Cluster[,2] == TheCluster)

#Names of the genes
GeneNames <- results$Cluster[ListOfGenes,1]

#Users can write these genes into a file for further analysis.
```

2.1.5.3 Draw a correlation heatmap

1. Choose 100 most significant genes from clustering results

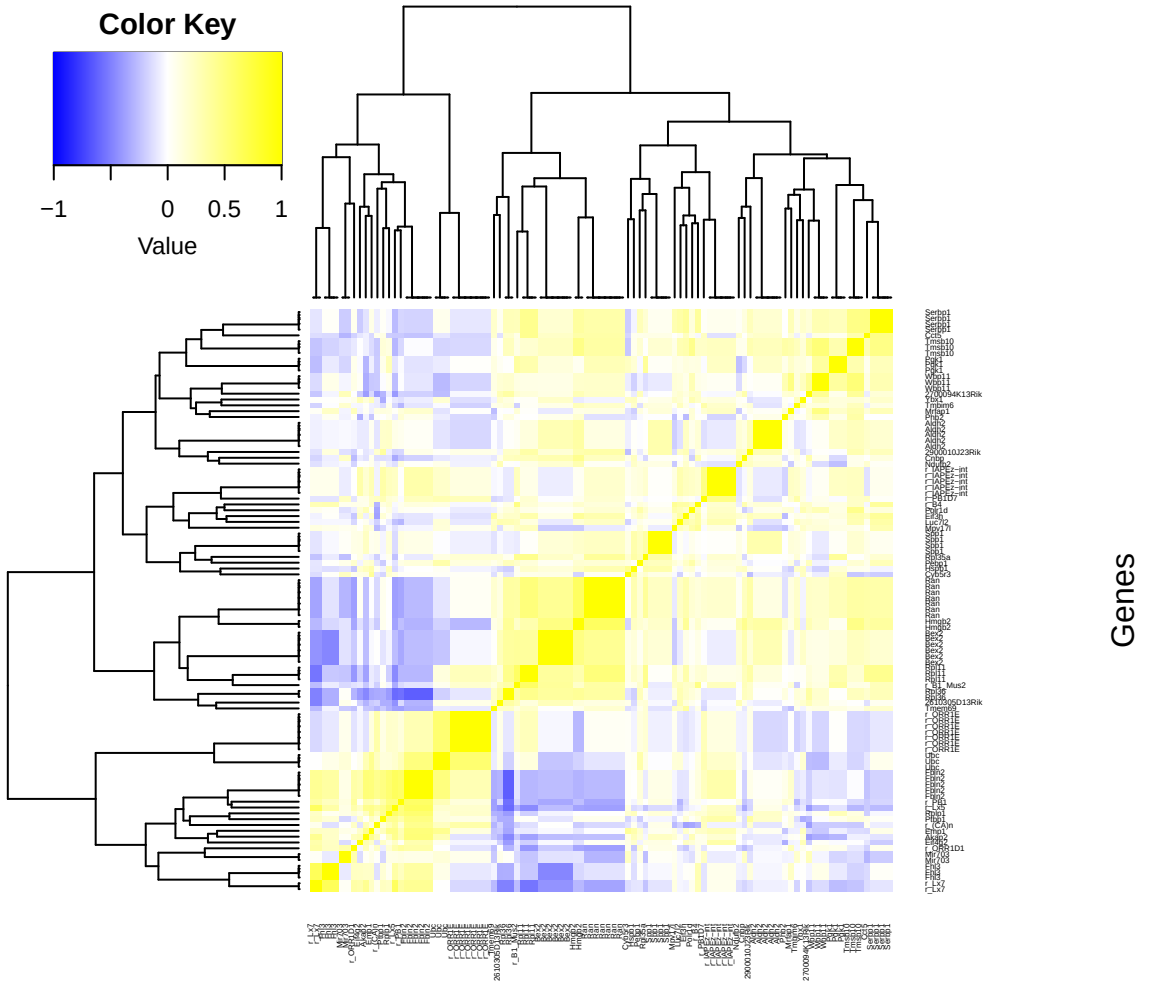
```
top100 <- results$Results[order(results$Results[,4],decreasing=FALSE)[1:100],1]
```

2. Extract these 100 genes from the correlation matrix.

```
Top100.Cor.Matrix <- results$Cor.Matrix[top100,top100]
```

3. Draw a correlation heatmap.

```
library(RColorBrewer)
library(gplots)
##
## Attaching package: 'gplots'
## The following object is masked from 'package:stats':
##
##     lowess
heatmap.2(as.matrix(Top100.Cor.Matrix),
#Hierarchical clustering on both row and column
Rowv=TRUE, Colv=TRUE,
#Center white color at correlation 0
symbreaks=TRUE,
#Turn off level trace
trace="none",
#x and y axis labels
xlab = 'Genes', ylab = "Genes",
#Turn off density info
density.info='none',
#Control color
key.ylab=NA,
col=colorRampPalette(c("blue", "white", "yellow"))(n = 1000),
lmat=rbind(c(4, 3), c(2, 1)),
#Gene name font size
cexRow=0.3,
cexCol=0.3,
#Margins
margins = c(8, 8)
)
```



2.1.6 Highly variable gene analysis

In this section, we will perform highly variable gene discovery on the embryonic stem cells from Islam(2011).

1. Obtain data.

```
data(Islam2011)

#Identify spike in genes:
SPIKEIN <- rownames(Islam2011)[c(grep("Ppia",rownames(Islam2011)),
grep("H2afz",rownames(Islam2011)),grep("Hprt1",rownames(Islam2011)))]
```

2. Analysis

```
#The first 48 columns are the embryonic stem cells.
results <- Linnorm.HVar(Islam2011[,1:48], spikein=SPIKEIN)
```

3. Print out the most significant 10 genes.

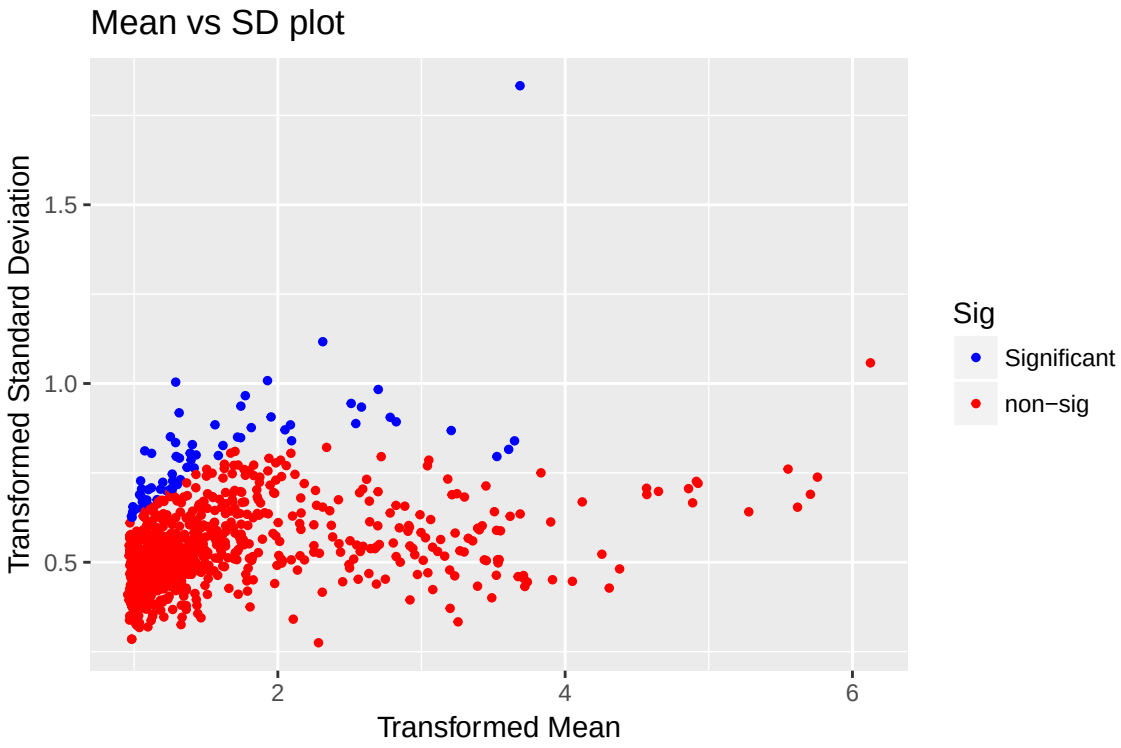
```
resultsdata <- results$Results
Genes10 <- resultsdata[order(resultsdata[,"q.value"]),][1:10,3:5]
#Users can print the gene list by the following command:
#print(Genes10)
```

	Normalized.Log2.SD.Fold.Change	p.value	q.value
RNA_SPIKE_5	1.4344	0.0000	0.0000
Snrpa	0.9013	0.0002	0.0689
Hmgb2	0.8124	0.0006	0.1695
Rsl1d1	0.7608	0.0012	0.2569
r_RMER16	0.7371	0.0016	0.2806
Bend4	0.6937	0.0028	0.3569
Hnrnpd	0.6913	0.0029	0.3569
Hnrnpf	0.6775	0.0034	0.3614
Gpx4	0.6694	0.0038	0.3614
Supt16h	0.6423	0.0052	0.4474

2.1.6.1 Mean vs SD plot highlighting significant genes

1. Simply print the plot from results.

```
print(results$plot$plot)
```



2.1.7 Hierarchical Clustering

In this section, we will perform hierarchical clustering on Islam2011 data.

1. Obtain data.

```
data(Islam2011)
Islam <- Islam2011[,1:92]
```

2. Assign group to samples.

```
#48 ESC, 44 EF, and 4 NegCtrl
Group <- c(rep("ESC",48),rep("EF",44))
colnames(Islam) <- paste(colnames(Islam),Group,sep="_")
```

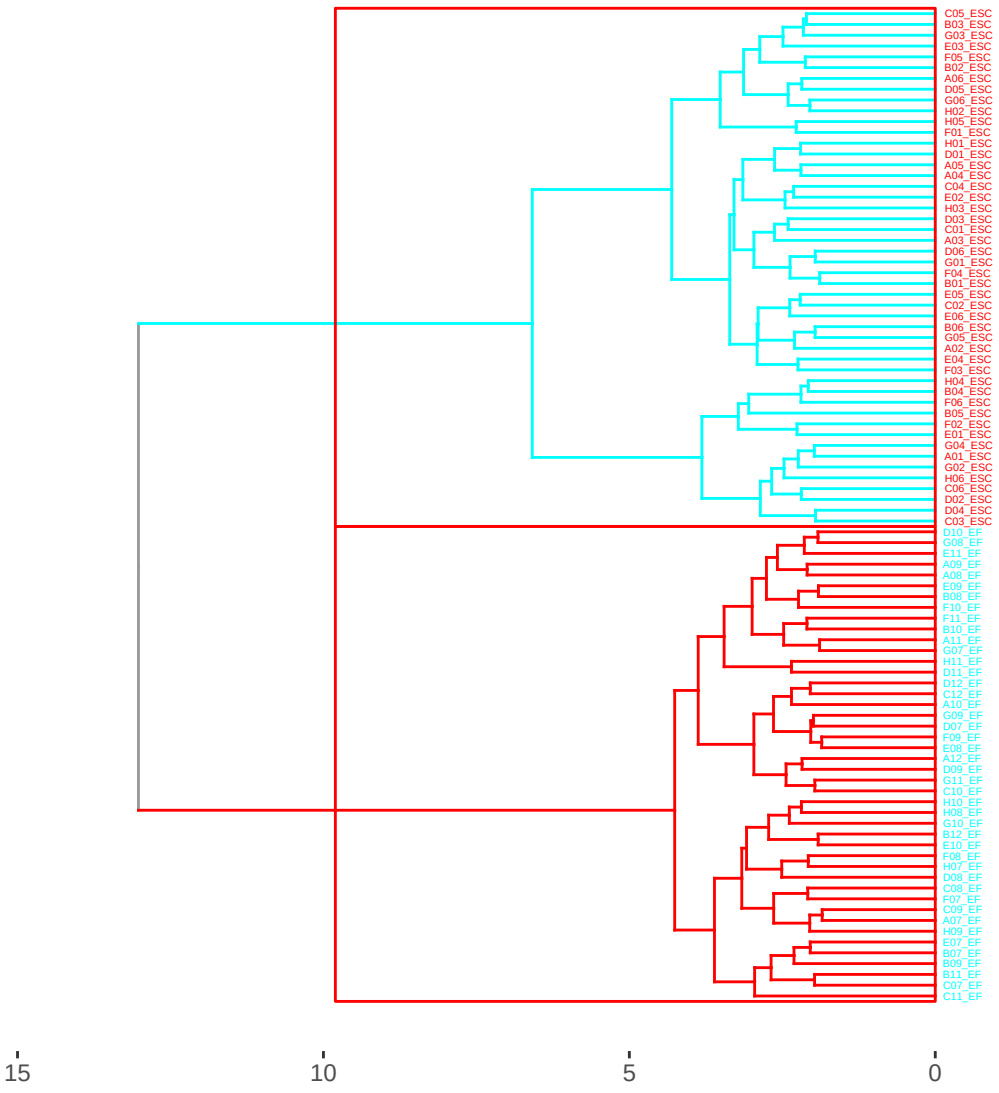
3. Perform Analysis.

```
#Note that there are 3 known clusters.
HClust.Results <- Linnorm.HClust(Islam,Group=Group, num_Clust=2, fontsize=1.5)
```

2.1.7.1 Hierarchical Clustering plot

We can simply print the plot out.

```
print(HClust.Results$plot$plot)
```



2.1.8 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 20 samples of TCGA RNA-seq data.

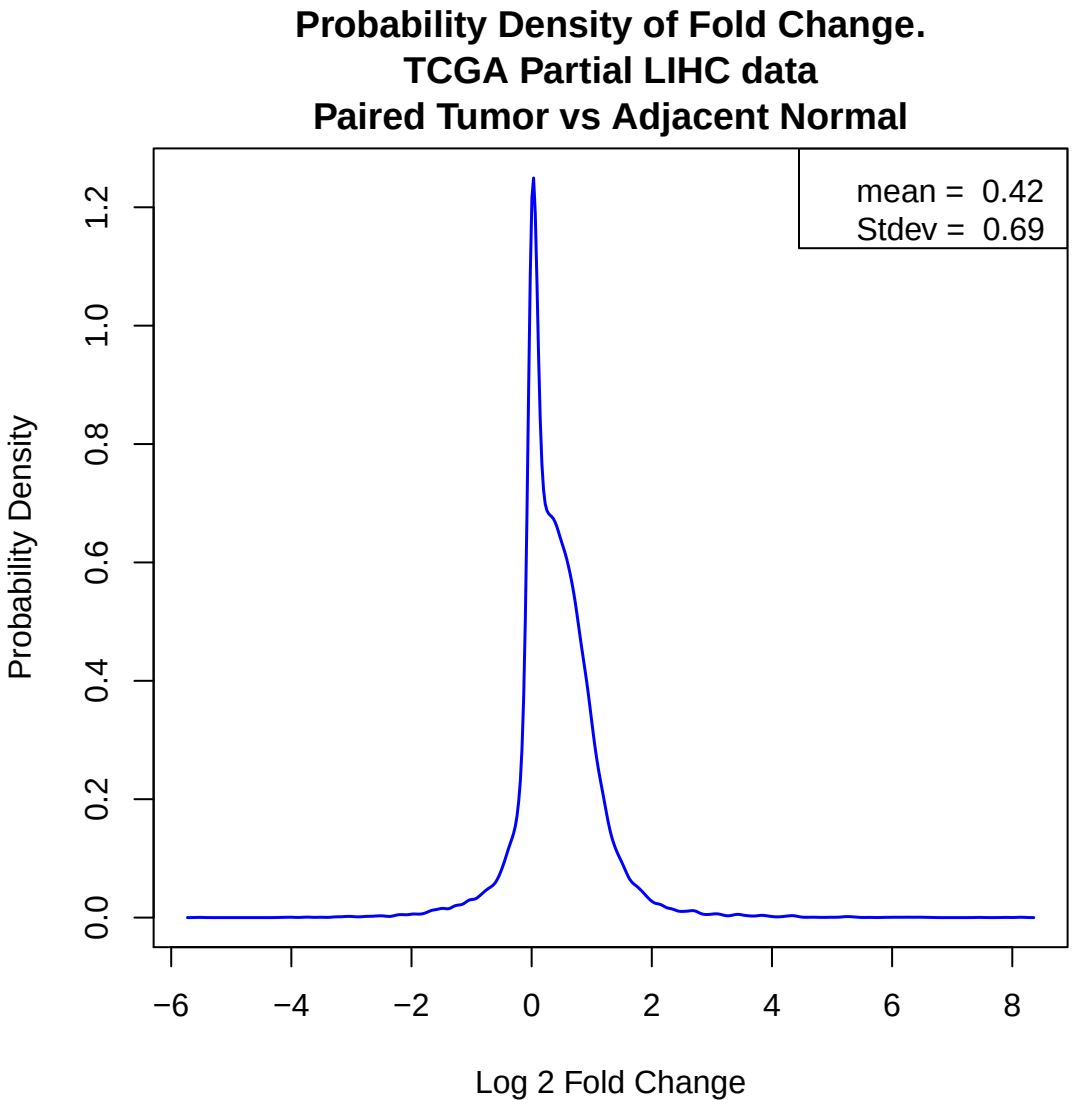
```
library(Linnorm)
data(LIHC)
```

2. Calculate Fold Change.

```
#Now, we can calculate fold changes between
#sample set 1 and sample set 2.
#Index of sample set 1
set1 <- 1:10
#Index of sample set 2
set2 <- 11:20

#Define a function that calculates log 2 fold change from TPM + 1:
log2fc <- function(x) {
return(log(mean(x[set1] + 1)/mean(x[set2] + 1),2))
}

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


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

#Remove Infinite values.
FCMatrix <- FCMatrix[!is.infinite(foldchanges),,drop=FALSE]

#Now FCMatrix contains fold change results.
```

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

```
Density <- density(FCMatrix[,1])
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.\nTCGA Partial LIHC data
Paired Tumor vs Adjacent Normal")
legend("topright",legend=paste("mean = ",
round(mean(FCMatrix[,1]),2),
"\nStdev = ", round(sd(FCMatrix[,1]),2)))
```



2.2 RnaXSim

2.2.1 RNA-seq Expression Data Simulation

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

1. Get RNA-seq data from SEQC. RnaXSim assume that all samples are replicate of each other.

```
data(SEQC)
SampleA <- SEQC
```

2. Simulate an RNA-seq dataset.

```
library(Linnorm)
#This will generate two sets of RNA-seq data with 5 replicates each.
#It will have 20000 genes totally with 5000 genes being differentially
#expressed. It has the Negative Binomial 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]
```

2.2.1.2 Advanced

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

1. Get RNA-seq data from SEQC.

```
data(SEQC)
SampleA <- SEQC
```

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

```
library(Linnorm)
SimulatedData <- RnaXSim(SampleA,
distribution="Gamma", #Distribution in the simulated dataset.
#Put "NB" for Negative Binomial, "Gamma" for Gamma,
#"Poisson" for Poisson and "LogNorm" for Log Normal distribution.
NumRep=5, #Number of replicates in each sample set.
#5 will generate 10 samples in total.
NumDiff = 1000, #Number of differentially expressed genes.
NumFea = 5000 #Total number of genes in the dataset
)
```

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

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

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

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

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

3 Frequently Asked Questions.

1. 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.

2. I only have two samples in total. Can I perform Linnorm Transformation?
   Answer: No, you cannot. Linnorm requires a minimum of 3 samples.
3. 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.
4. 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, which is the default.
5. Do I need to run Linnorm.Norm() in addition to transforming the datset with Linnorm()?
   Answer: Linnorm()'s transformation also performs Linnorm.Norm()'s normalization step. Therefore, please **DO NOT** rerun Linnorm.Norm() before or after Linnorm().

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!