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]**

4 January 2019

[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 single cell RNA-seq (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[1]. Various functions and analysis pipelines are also implemented for users' convenience, including data imputation[2], stable gene selection for scRNA-seq[3], using *limma* for differential expression analysis or differential peak detection[4], co-expression network analysis[5], highly variable gene discovery[6] and cell subpopulation analysis with t-SNE K-means clustering[7], PCA K-means clustering[8] and hierarchical clustering analysis[9]. Linnorm can work with raw count, CPM, RPKM, FPKM and TPM and is compatible with data generated from simple count algorithms[10] and supervised learning algorithms[11]. Additionally, the RnaXSim function is included for simulating RNA-seq data for the evaluation of DEG analysis methods.

[1] Implemented as Linnorm and Linnorm.Norm

[2] Implemented as the Linnorm.DataImput function.

[3] for scRNA-seq data without spike-in genes or users who do not want to rely on spike-in genes. Implemented as the Linnorm.SGenes function.

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

[5] Implemented as the Linnorm.Cor function.

[6] Implemented as the Linnorm.HVar function.

[7] Implemented as the Linnorm.tSNE function.

[8] Implemented as the Linnorm.PCA function.

[9] Implemented as the Linnorm.HClust function.

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

[11] Such as Cufflinks, eXpress, Kallisto, RSEM, Sailfish, and etc

Contents

#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 Transformation/Normalization/Imputation
  - Normalizing transformation (Linnorm)
  - Normalization (Linnorm.Norm)
  - Data Imputation (Linnorm.DataImput)
- Stable gene selection for scRNA-seq datasets without spike-in genes (Linnorm.SGenes)
- The Linnorm-limma pipeline (Linnorm.limma)
  - Differential expression analysis
  - Differential peak detection
- Coexpression network analysis (Linnorm.Cor)
- Highly variable gene discovery (Linnorm.HVar)
- Single cell RNA-seq subpopulation analysis (Clustering)
  - t-SNE k-means clustering (Recommended) (Linnorm.tSNE)
  - PCA k-means clustering (Linnorm.PCA)
  - Hierarchical clustering analysis (Linnorm.HClust)
- RNA-seq data simulation for negative binomial, Poisson, log normal or gamma distribution. (RnaXSim)

Compared to RNA-seq or other large scale count data, scRNA-seq data have more noises and more zero counts. Linnorm is developed to work with scRNA-seq data. Since RNA-seq data are similar to scRNA-seq data, but with less noise, Linnorm is also compatible with RNA-seq data analysis.

##Datatypes and Input Format

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

- Raw Count (scRNA-seq, RNA-seq, ChIP-seq or any large scale count data)
- 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. Please do **NOT** input Log transformed datasets, as Linnorm will perform log transformation.

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 setting the **RowSamples** argument as **TRUE**, Linnorm can accept a matrix where each column is a sample or replicate and each row should be a Gene/Exon/Isoform/etc. Example:

	Gene 1	Gene 2	Gene 3	...
Sample1	1	3	10.87	...
Sample2	2	0	11.56	...
Sample3	1	4	12.98	...
.....	...	...	...	...
.....	...	...	...	...

Note that Linnorm should work slightly faster with **RowSamples** set to **TRUE.**

By using the argument, "input ="Linnorm"", several functions provided by the Linnorm package can also accept Linnorm transformed datasets as input. Therefore, the dataset doesn't need to be re-transformed multiple times for multiple functions.

##Installation Instructions can be found on Linnorm's bioconductor page. Please note that some functions in this vignette are only available in Linnorm version 1.99.x+.

#Examples with Source Codes

##Data Normalization/Transformation/Imputation

###Normalizing 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)

#Obtain data
data(Islam2011)

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

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

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)
```

###Normalization

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

1. Linnorm Normalization

```
library(Linnorm)
Normalized <- Linnorm.Norm(Islam2011)
#Important: depending on downstream analysis, please
#set output to be "XPM" or "Raw".
#Set to "XPM" if downstream tools will convert the
#dataset into CPM or TPM.
#Set to "Raw" if input is raw counts and downstream
#tools will work with raw counts.
```

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

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

####Calculating Fold Change and the effects of normalization strength

#####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 10 samples of TCGA RNA-seq data and convert it into TPM.

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

2. Data filteirng and Normalization

```
#Now, we can calculate fold changes between
#sample set 1 and sample set 2.
#Index of sample set 1
set1 <- 1:5
#Index of sample set 2
set2 <- 6:10

#Normalization
LIHC2 <- Linnorm.Norm(LIHC,output="XPM")

#Optional: Only use genes with at least 50%
#of the values being non-zero
LIHC2 <- LIHC2[rowSums(LIHC2 != 0) >= ncol(LIHC2)/2,]
```

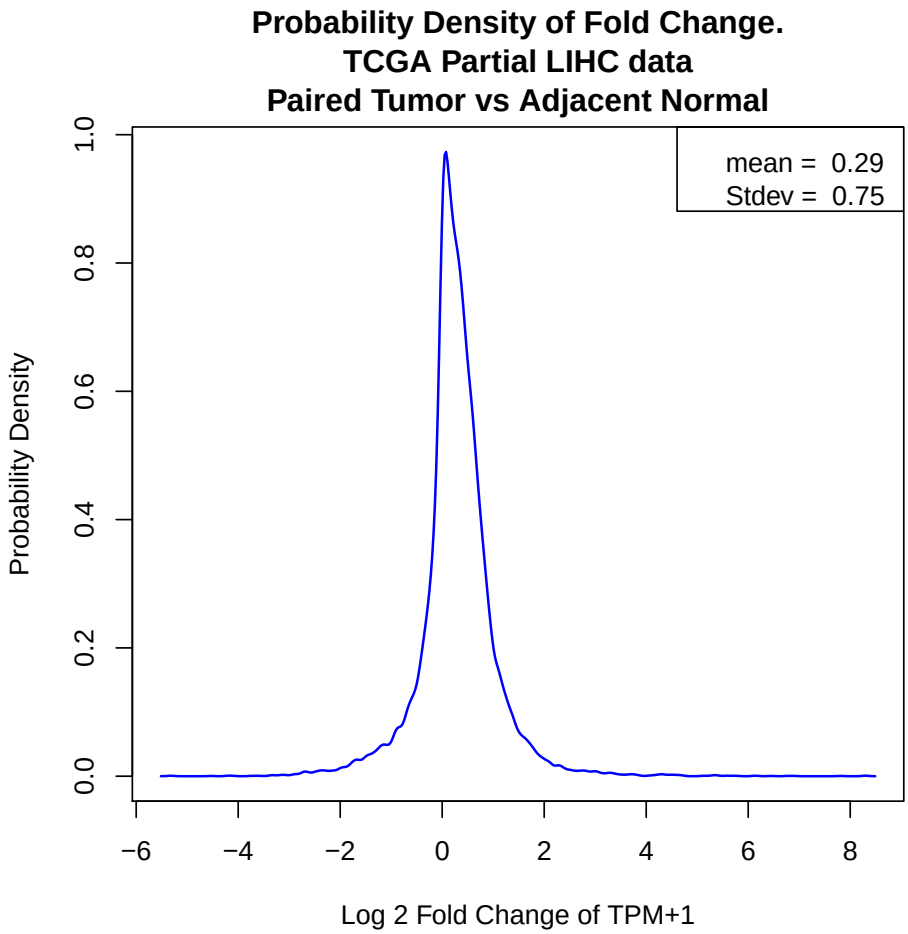
2. Calculate Fold Change.

```
#Put resulting data into a matrix
FCMatrix <- matrix(0, ncol=1, nrow=nrow(LIHC2))
rownames(FCMatrix) <- rownames(LIHC2)
colnames(FCMatrix) <- c("Log 2 Fold Change")
FCMatrix[,1] <- log((rowMeans(LIHC2[,set1])+1)/(rowMeans(LIHC2[,set2])+1),2)
#Now FCMatrix contains fold change results.

#If the optional filtering step is not done,
#users might need to remove infinite and zero values:
#Remove Infinite values.
FCMatrix <- FCMatrix[!is.infinite(FCMatrix[,1]),,drop=FALSE]
#Remove Zero values
FCMatrix <- FCMatrix[FCMatrix[,1] != 0,,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 of TPM+1",
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)))
```

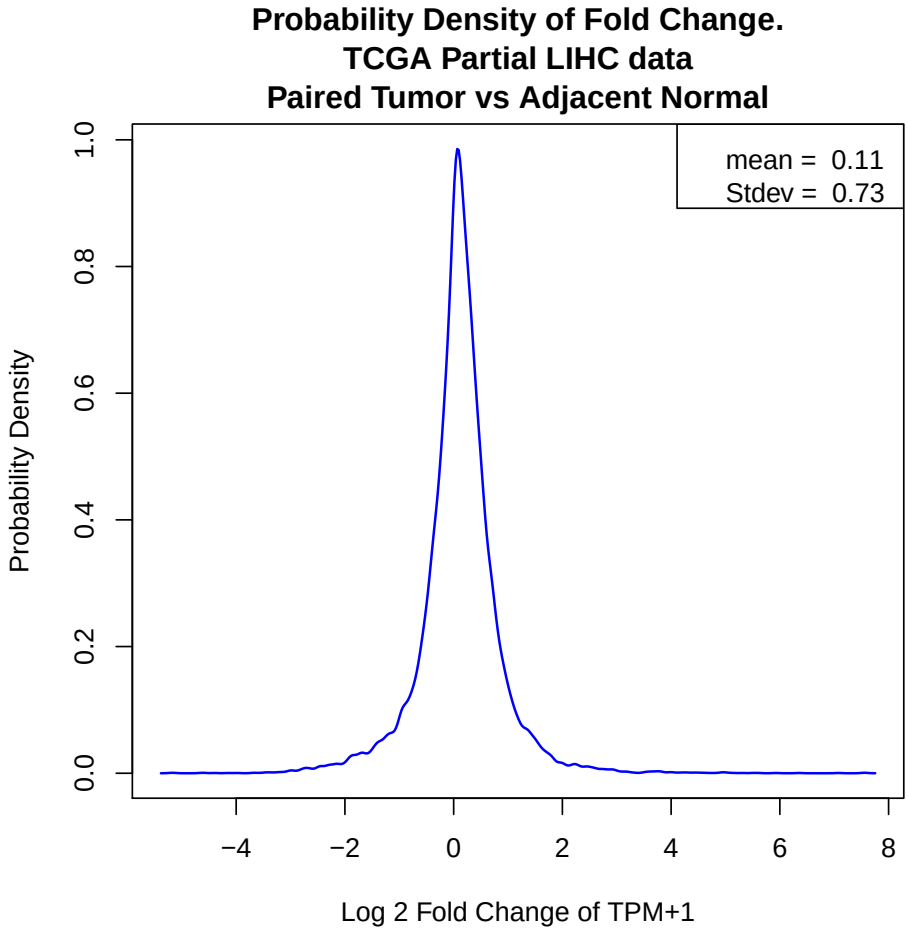


0.0.0.0.1 Effects of normalization strength

4. Using a higher normalization strength in Linnorm.Norm

```
#Normalization with BE_strength set to 1.
#This increases normalization strength.
LIHC2 <- Linnorm.Norm(LIHC,output="XPM",BE_strength=1)

#Optional: Only use genes with at least 50%
#of the values being non-zero
LIHC2 <- LIHC2[rowSums(LIHC2 != 0) >= ncol(LIHC2)/2,]

FCMatrix <- matrix(0, ncol=1, nrow=nrow(LIHC2))
rownames(FCMatrix) <- rownames(LIHC2)
colnames(FCMatrix) <- c("Log 2 Fold Change")
FCMatrix[,1] <- log((rowMeans(LIHC2[,set1])+1)/(rowMeans(LIHC2[,set2])+1),2)
#Now FCMatrix contains fold change results.
```

```
Density <- density(FCMatrix[,1])
plot(Density$x,Density$y,type="n",xlab="Log 2 Fold Change of TPM+1",
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)))
```



We can see that using a stronger normalization strength can decrease the amount of differences between the two conditions; this causes the mean of fold change to decrease. By default, BE_strength is set to 0.5 to prevent overfitting. However, user can adjust it based on the dataset. Please note that the Linnorm() function for normalizing transformation also allows users to control normalization strength.

###Data Imputation

By default, Linnorm's data imputation function replaces zerores in the dataset.

However, we do not recommend performing data imputation unless it is shown to improve results.

1. Linnorm Data Imputation

```
library(Linnorm)
data(Islam2011)
#Obtain transformed data
Transformed <- Linnorm(Islam2011)

#Data Imputation
DataImputed <- Linnorm.DataImput(Transformed)

#Or, users can directly perform data imputation during transformation.
DataImputed <- Linnorm(Islam2011,DataImputation=TRUE)
```

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

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

##Stable Gene Selection

This function selects stable gene/features from the dataset. It is implemented for users who need spike-in genes or do not want to rely on spike-in genes in scRNA-seq data analysis.

1. Stable Genes Selection

```
library(Linnorm)
data(Islam2011)
#Obtain stable genes
StableGenes <- Linnorm.SGenes(Islam2011)
```

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

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

##Differential Expression Analysis using Linnorm-limma pipeline

- 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.

###RNA-seq data ####Analysis procedure

We use 10 samples of RNA-seq data. These 10 samples are paired tumor and adjacent normal tissue samples from 5 individuals from the TCGA LIHC dataset.

1. Obtain data

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

The Linnorm-limma pipeline only consists of two steps.

2. Create limma design matrix

```
#5 samples for condition 1 and 5 samples for condition 2.
#You might need to edit this line
design <- c(rep(1,5),rep(2,5))
#These lines can be copied directly.
design <- model.matrix(~ 0+factor(design))
colnames(design) <- c("group1", "group2")
rownames(design) <- colnames(datamatrix)
```

3. 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.
```

####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.

#Note all columns are printed here
```

	logFC	XPM	t	P.Value	adj.P.Val
CACNA1S|779	0.6931	0.0065	43.9444	0	0
HOXD4|3233	2.9023	0.0684	40.2719	0	0
PYY|5697	2.0851	0.0342	37.4934	0	0
VGF|7425	1.4371	0.0180	35.1250	0	0
SFTA1P|207107	4.2981	0.1972	32.6785	0	0
LOC221122|221122	2.5027	0.0493	29.4309	0	0
LOC127841|127841	1.7792	0.0257	26.8891	0	0
IRX5|10265	2.7649	0.0612	24.7712	0	0
CCNA1|8900	1.4929	0.0192	24.6046	0	0
TCP10L2|401285	-1.3745	0.0168	-22.0944	0	0

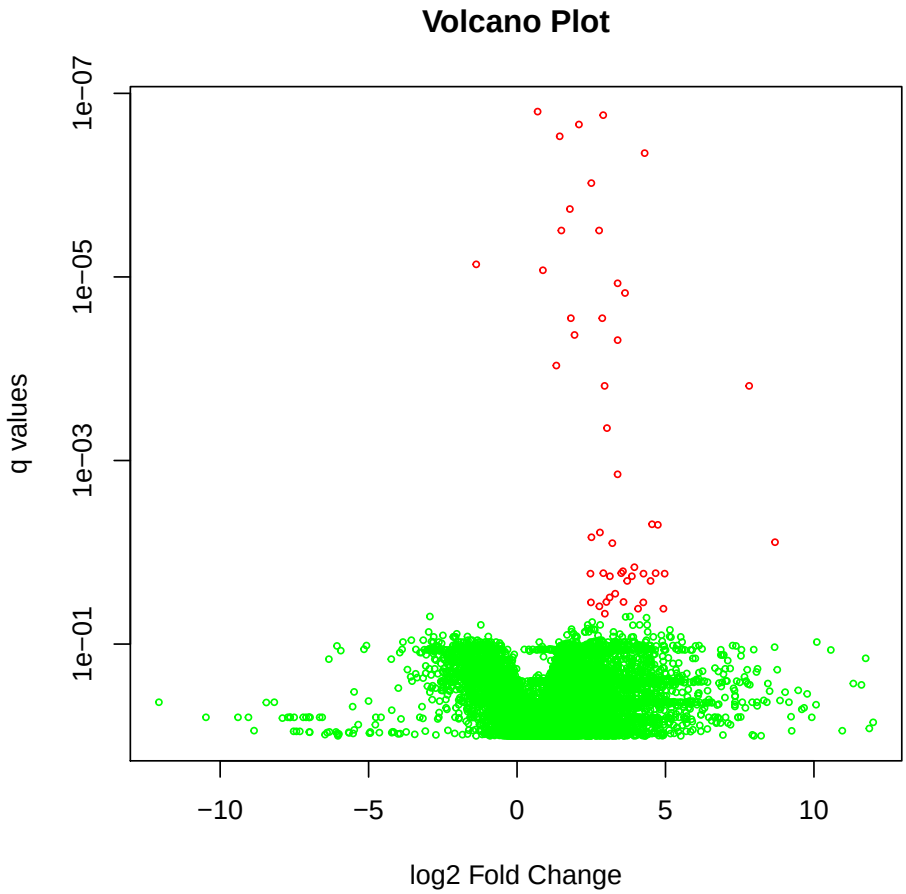
0.0.0.1 Volcano Plot

#####Procedure 1. Record significant genes for coloring

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

2. Draw volcano plot with Log q values. Green dots are non-significant, red dots are significant.

```
plot(x=DEG_Results[,1], y=DEG_Results[,5], col=ifelse(DEG_Results[,1] %in%
SignificantGenes, "red", "green"),log="y", ylim =
rev(range(DEG_Results[,5])), main="Volcano Plot",
xlab="log2 Fold Change", ylab="q values", cex = 0.5)
```



###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.

1. Obtain data

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

2. 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)
```

3. DEG Analysis

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


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

##Gene Co-expression Network Analysis
###Analysis Procedure 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.

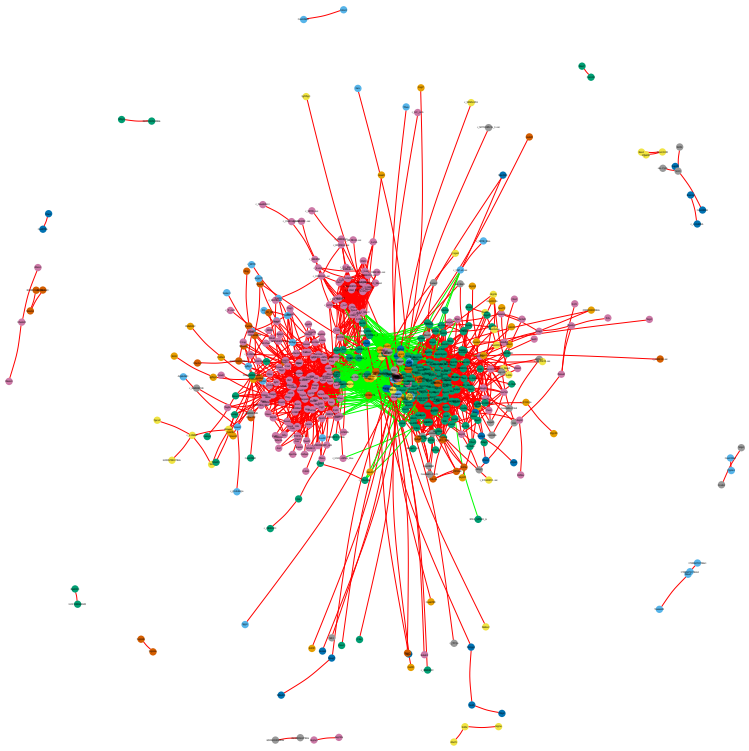
```
Genes10 <- results$Results[order(results$Results[,5]),][1:10,]
#Users can print the gene list by the following command:
#print(Genes10)
```

Gene1	Gene2	Cor	p.value	q.value
Amd2	Amd1	1.0000	0	0
1700047I17Rik2	1700047I17Rik1	1.0000	0	0
Rpl26	Gm15772	0.9995	0	0
Ftl2	Ftl1	0.9990	0	0
Gm13138	Rex2	0.9982	0	0
Oaz1	Gm9786	0.9973	0	0
BC002163-Chr4	Ndufs5	0.9964	0	0
Rps26	Gm6654	0.9888	0	0
r_(T)n	r_(A)n	0.9885	0	0
Hnrnpa1	Gm5643	0.9852	0	0

###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
)
```



###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.
```

###Draw a correlation heatmap

1. Choose 100 most significant genes from clustering results

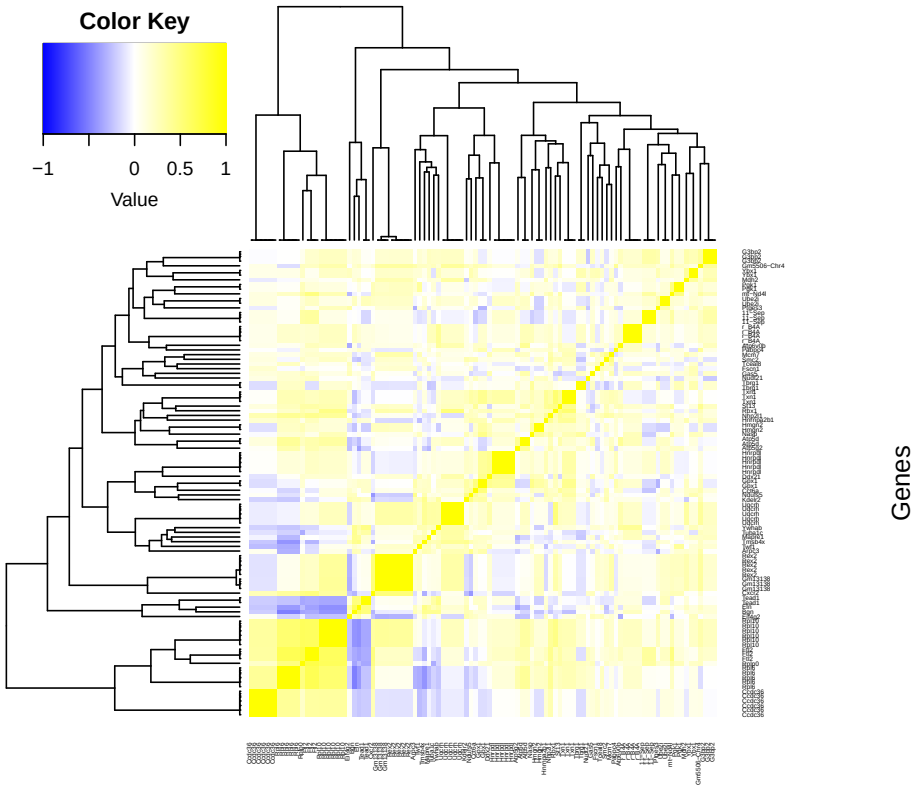
```
top100 <- results$Results[order(results$Results[,4],decreasing=FALSE)[1:100],1]
```

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)
)
```



##Highly variable gene analysis
###Analysis Procedure

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

1. Obtain data.

```
data(Islam2011)
```

2. Analysis

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

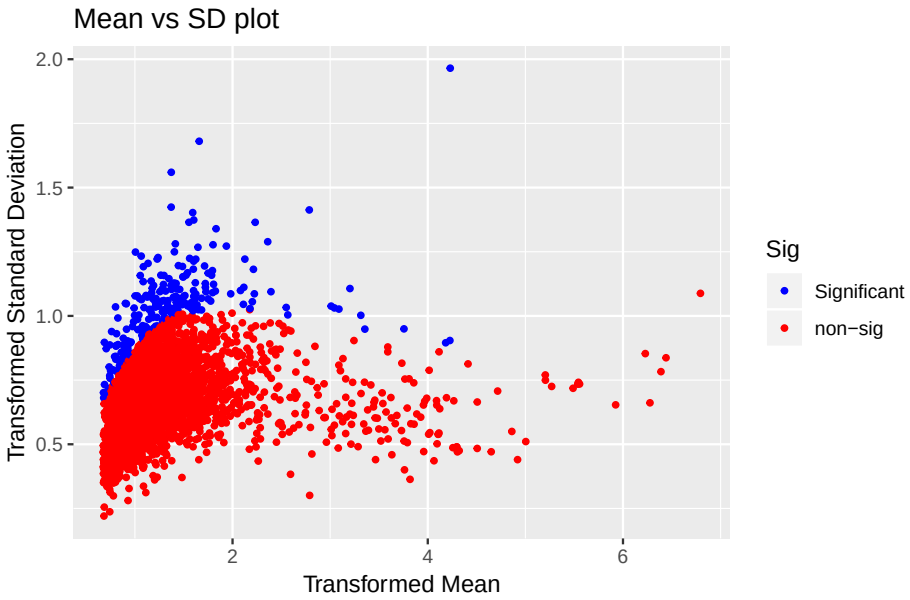
3. Print out the most significant 10 genes.

```
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	1.3220	0e+00	0.0004
Ptcra	1.0955	0e+00	0.0166
Sec61a2	1.0399	0e+00	0.0289
r_(CAA)n	0.9373	1e-04	0.1126
Kctd5	0.9151	2e-04	0.1262
r_(TTG)n	0.8830	3e-04	0.1446
Hmgb2	0.8929	2e-04	0.1446
Efna4	0.8713	3e-04	0.1499
Mrpl45	0.8586	4e-04	0.1597
Skil	0.8157	7e-04	0.1943

###Mean vs SD plot highlighting significant genes
1. Simply print the plot from results.

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




```
#By default, this plot highlights genes/features with p value less than 0.05.
```

##Cell subpopulation analysis

###t-SNE K-means Clustering

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.

1. Read data.

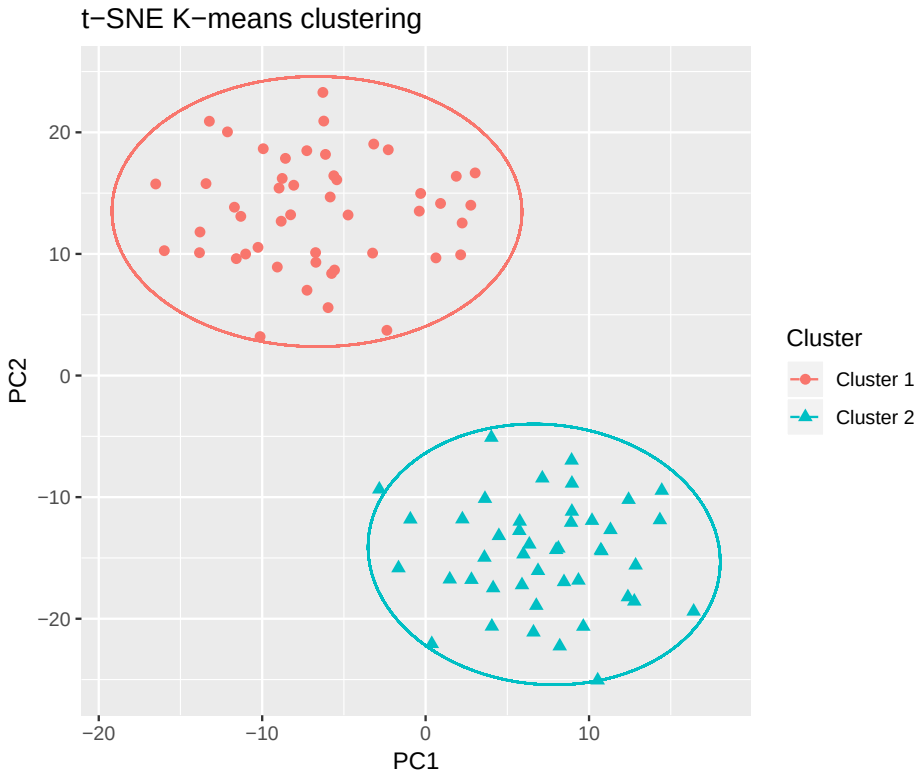
```
library(Linnorm)
data(Islam2011)
```

####Simple subpopulation analysis
1. Perform t-SNE analysis using default configurations. This also automiatically determines the number of cell subpopulations.

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

2. Draw t-SNE k-means clustering plot.

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



####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))
```

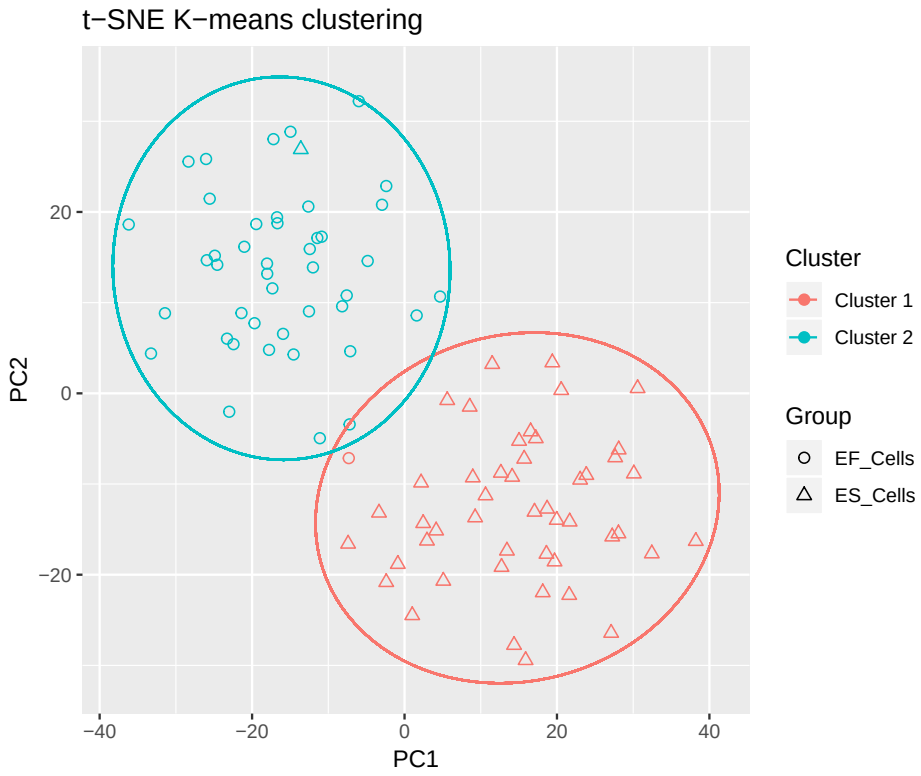
2. Perform tSNE analysis.

```
#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.

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

```
        Group=Groups, num_center=2)
```

3. Draw t-SNE k-means clustering plot.

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



###PCA K-means Clustering.

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)
```

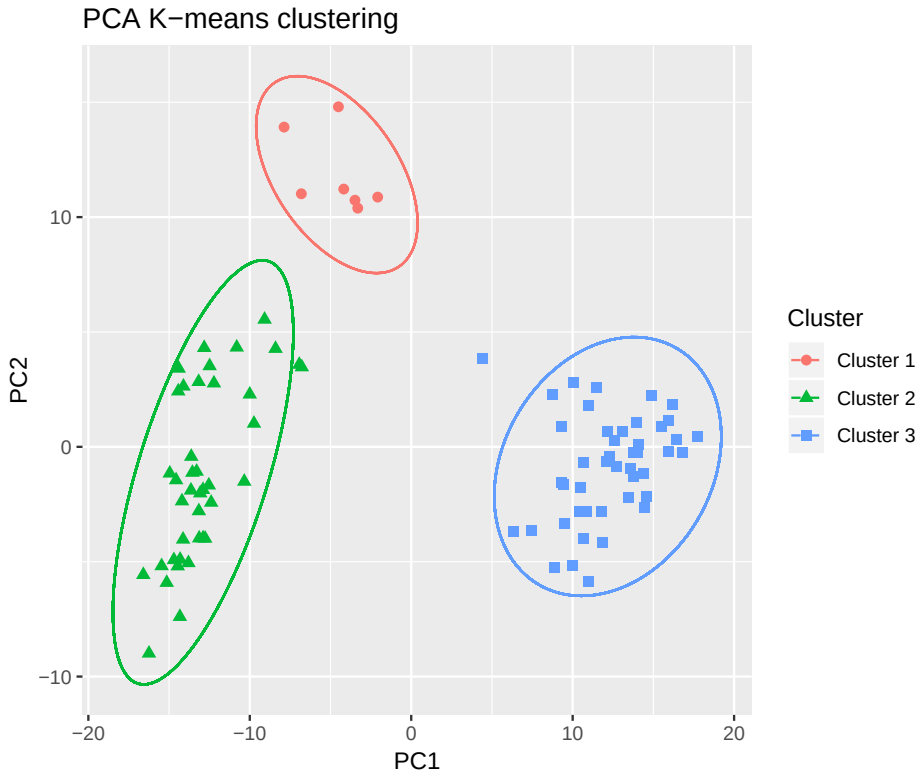
####Simple subpopulation analysis

1. Perform PCA analysis using default configurations.

```
PCA.results <- Linnorm.PCA(Islam2011[,1:92])
## To perform cell clustering, Linnorm.tSNE is strongly recommended. Linnorm.PCA is only provided as a refere
```

2. Draw PCA k-means clustering plot.

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



####Analysis with known subpopulations

1. Assign known groups to samples.

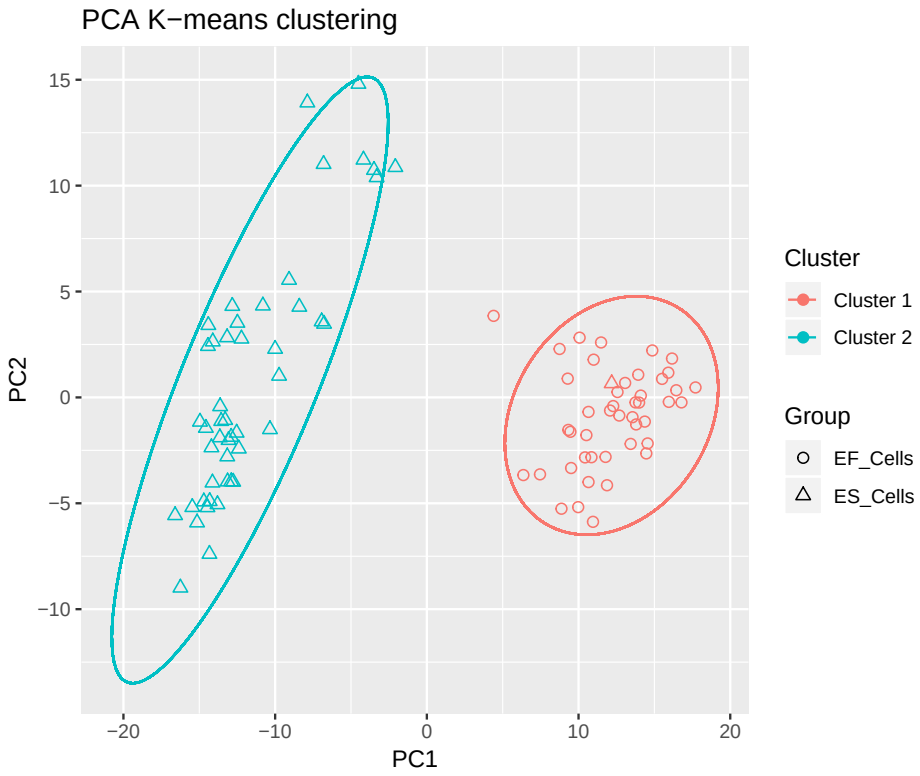
```
#The first 48 samples belong to mouse embryonic stem cells.
Groups <- rep("ES_Cells",48)
#The next 44 samples are mouse embryonic fibroblasts.
Groups <- c(Groups, rep("EF_Cells",44))
```

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)
## To perform cell clustering, Linnorm.tSNE is strongly recommended. Linnorm.PCA is only provided as a refere
```

3. Draw PCA k-means clustering plot.

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



###Hierarchical Clustering
####Analysis Procedure 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.

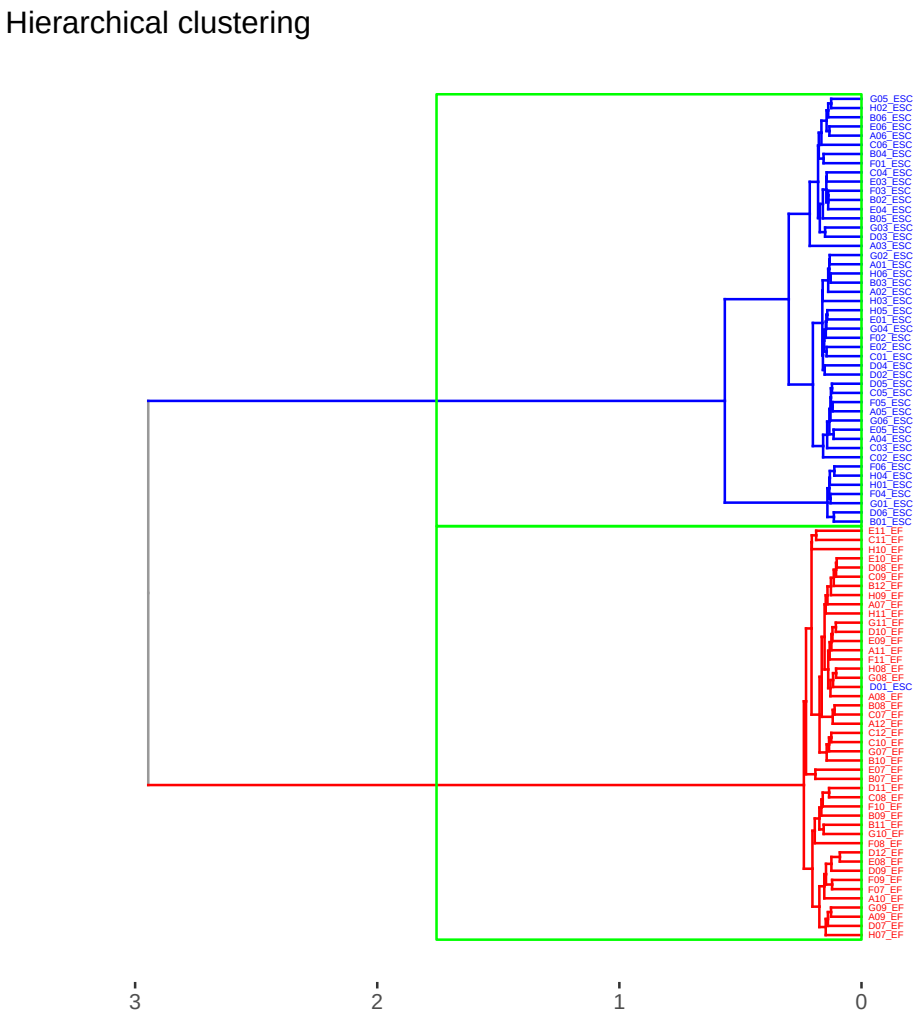
```
#48 ESC, 44 EF, and 4 NegCtrl
Group <- c(rep("ESC",48),rep("EF",44))
colnames(Islam) <- paste(colnames(Islam),Group,sep="_")
```

3. Perform Analysis.

```
#Note that there are 3 known clusters.
HClust.Results <- Linnorm.HClust(Islam,Group=Group,
num_Clust=2, fontsize=1.5, Color = c("Red","Blue"), RectColor="Green")
```

####Hierarchical Clustering plot
We can simply print the plot out.

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



##RnaXSim

###RNA-seq Expression Data Simulation ####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.

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

2. Simulate an RNA-seq dataset.

```
#This will generate two sets of RNA-seq data with 5 replicates each.
#It will have 20000 genes totally with 2000 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]
```

####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]
```

#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().

#Bug Reports, Questions and Suggestions

We appreciate and welcome any Bug Reports, Questions and Suggestions. They can be posted on bioconductor's support site, https://support.bioconductor.org. Please remember to add the Linnorm tag in your post so that Ken will be notified. Bioconductor's posting guide can be found at http://www.bioconductor.org/help/support/.