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. Various analysis pipelines are also implemented for users’ convenience, including using \textit{limma} for differential expression analysis or differential peak detection\textsuperscript{1}, calculating correlation coefficient for gene correlation study\textsuperscript{2}, subpopulation analysis pipeline with PCA and k-means clustering\textsuperscript{3}, highly variable gene discovery\textsuperscript{4} and hierarchical clustering analysis\textsuperscript{5}. Linnorm can work with raw count, CPM, RPKM, FPKM and TPM and is compatible with data generated from simple count algorithms\textsuperscript{6} and supervised learning algorithms\textsuperscript{7}.

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 distribution.

\textsuperscript{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.
\textsuperscript{2}Implemented as the Linnorm.Cor function.
\textsuperscript{3}Implemented as the Linnorm.PCA function.
\textsuperscript{4}Implemented as the Linnorm.HVar function.
\textsuperscript{5}Implemented as the Linnorm.HClust function.
\textsuperscript{6}Such as HTSeq, Rsubread and etc
\textsuperscript{7}Such as Cufflinks, eXpress, Kallisto, RSEM, Sailfish, and etc
# Contents

1 Introduction 3  
1.1 Linnorm 3  
1.1.1 Datatypes and Input Format 3  
1.2 RnaXSim 4  
1.2.1 Inputs 4  

2 Examples with Source Codes 5  
2.1 Linnorm 5  
2.1.1 Linnorm-limma Differential Expression Analysis 5  
2.1.1.1 Obtain example data 5  
2.1.1.2 Analysis procedure 5  
2.1.1.3 Print out the most significant genes 5  
2.1.1.4 Volcano Plot 6  
2.1.2 Single cell RNA-seq DEG Analysis 7  
2.1.3 PCA K-means Clustering. Cell subpopulation analysis. 9  
2.1.3.1 Simple subpopulation analysis 9  
2.1.3.2 Analysis with known subpopulations 10  
2.1.4 Gene Correlation Network Analysis 12  
2.1.4.1 Plot a co-expression network. 12  
2.1.4.2 Identify genes that belong to a cluster. 14  
2.1.4.3 Draw a correlation heatmap. 14  
2.1.5 Highly variable gene analysis 15  
2.1.5.1 Mean vs SD plot highlighting significant genes. 16  
2.1.6 Hierarchical Clustering 17  
2.1.6.1 Hierarchical Clustering plot 18  
2.1.7 Basic Linnorm Transformation 18  
2.1.7.1 Calculate Fold Change 19  
2.2 RnaXSim 21  
2.2.1 RNA-seq Raw Count Simulation 21  
2.2.1.1 Default 21  
2.2.1.2 Advanced 21  

3 Frequently Asked Questions 23  
3.1 How do I convert Linnorm Transformed dataset back to CPM/TPM? 23  
3.2 Can I use Linnorm Transformed dataset to calculate Fold Change? 23  
3.3 I only have 1 replicate for each sample set. Can I perform Linnorm Transformation? 23  
3.4 I only have 1 replicate for each sample set. Can I perform Differential Expression Analysis with Linnorm and limma? 23  
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? 23  
3.6 During installation, an error says that Linnorm is not available. What is the problem? 23  

4 Bug Reports, Questions and Suggestions 24
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 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 PCA and 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
- PCA k-means clustering (Linnorm.PCA)
  - Single cell RNA-seq subpopulation analysis
- Highly variable gene discovery (Linnorm.HVar)
- Gene correlation analysis (Linnorm.Cor)
- Hierarchical clustering analysis (Linnorm.HClust)
- 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)

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:

<table>
<thead>
<tr>
<th>Gene1</th>
<th>1</th>
<th>2</th>
<th>1</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gene2</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>...</td>
</tr>
<tr>
<td>Gene3</td>
<td>10.87</td>
<td>11.56</td>
<td>12.98</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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

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

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.

Each sample in the data matrix is assumed to be replicates of each other. The seqc package is a good source of such datasets.
2 Examples with Source Codes

2.1 Linnorm

2.1.1 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.

2.1.1.1 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.

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

2.1.1.2 Analysis procedure The Linnorm-limma pipeline only consists of two steps.

1. Create limma design matrix

   ```r
   #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))
#Just copy these lines to R
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.)

   ```r
   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

   ```r
   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.
   ```

2.1.1.3 Print out the most significant genes
1. Write out the results to a tab delimited file.

```r
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.

```r
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.

<table>
<thead>
<tr>
<th>Gene</th>
<th>logFC</th>
<th>XPM</th>
<th>t</th>
<th>P.Value</th>
<th>adj.P.Val</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESP1</td>
<td>3.4767</td>
<td>1983297.3</td>
<td>11.5860</td>
<td>0</td>
<td>0</td>
<td>13.4824</td>
</tr>
<tr>
<td>PKMYT1</td>
<td>4.3928</td>
<td>1458559.7</td>
<td>10.5951</td>
<td>0</td>
<td>0</td>
<td>12.0917</td>
</tr>
<tr>
<td>LY6H</td>
<td>4.8593</td>
<td>332377.1</td>
<td>10.3371</td>
<td>0</td>
<td>0</td>
<td>11.7096</td>
</tr>
<tr>
<td>MND1</td>
<td>5.2747</td>
<td>153593.2</td>
<td>9.9475</td>
<td>0</td>
<td>0</td>
<td>11.1164</td>
</tr>
<tr>
<td>CDKN3</td>
<td>4.8787</td>
<td>5437751.9</td>
<td>9.8429</td>
<td>0</td>
<td>0</td>
<td>10.9537</td>
</tr>
<tr>
<td>PLVAP</td>
<td>3.5451</td>
<td>31123379.5</td>
<td>9.1870</td>
<td>0</td>
<td>0</td>
<td>9.8993</td>
</tr>
<tr>
<td>GABRD</td>
<td>5.2132</td>
<td>1251463.6</td>
<td>9.1806</td>
<td>0</td>
<td>0</td>
<td>9.8886</td>
</tr>
<tr>
<td>IGF2AS</td>
<td>-3.9964</td>
<td>126894.8</td>
<td>-9.0374</td>
<td>0</td>
<td>0</td>
<td>9.6700</td>
</tr>
<tr>
<td>MESP2</td>
<td>4.3560</td>
<td>596508.0</td>
<td>8.9820</td>
<td>0</td>
<td>0</td>
<td>9.5574</td>
</tr>
<tr>
<td>NDUFA4L2</td>
<td>2.7021</td>
<td>8830098.7</td>
<td>8.8068</td>
<td>0</td>
<td>0</td>
<td>9.2604</td>
</tr>
</tbody>
</table>

2.1.1.4 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.

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

2. Record significant genes for coloring

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

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

```r
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)
```
2.1.2 Single cell RNA-seq DEG Analysis

By default, Linnorm filters gene with excess zero counts for data modeling, which is optimized for RNA-seq data. Since single cell RNA-seq data often contain a larger amount of zeroes, it is suggested to adjust the minZeroPortion argument during 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:

```r
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))
#Just copy these lines to R
design <- model.matrix(~ 0+factor(design))
colnames(design) <- c("group1", "group2")
rownames(design) <- colnames(IslamData)

## 2. DEG Analysis

### Example 1: Filter genes with more than 50% of the expressions being zero
scRNAseqResults <- Linnorm.limma(IslamData,design,minZeroPortion=0.5)

### Example 2: Do not filter gene list.
scRNAseqResults <- Linnorm.limma(IslamData,design,minZeroPortion=0)
2.1.3 PCA K-means Clustering. Cell subpopulation analysis.

In this section, we use Islam et al. (2011)’s single cell RNA-seq dataset to perform subpopulation 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.

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

2.1.3.1 Simple subpopulation analysis

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

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

2. Draw PCA k-means clustering plot.

```r
# Here, we can see three clusters.
print(PCArезультат$plot$plot)
```
2.1.3.2 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:
minZeroPortion:
it tells Linnorm how much zero in each gene is acceptable.
Suggested: 2/3 for RNA-seq, 0 for scRNA-seq.
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],
minZeroPortion=0, Group=Groups, num_center=3, num_PC=2)

3. Draw PCA k-means clustering plot.

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

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

2. Perform analysis.

```r
# Setting plotNetwork to TRUE will create a figure file in your current directory. Setting it to FALSE will still create an igraph object, so that users can plot it manually later.
# For this vignette, we will plot it manually instead 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.

```r
Genes10 <- results$Results[order(results$Results[,7]),[1:10,]
# Users can print the gene list by the following command:
# print(Genes10)
```

<table>
<thead>
<tr>
<th>Gene1</th>
<th>Gene2</th>
<th>XPM1</th>
<th>XPM2</th>
<th>Cor</th>
<th>p.value</th>
<th>q.value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gm15772</td>
<td>Rpl26</td>
<td>2605.3468</td>
<td>2696.5858</td>
<td>0.9993</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Ftl2</td>
<td>Ftl1</td>
<td>21464.6783</td>
<td>21647.4451</td>
<td>0.9993</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Gm9786</td>
<td>Oaz1</td>
<td>240.8487</td>
<td>246.4908</td>
<td>0.9962</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Gm5643</td>
<td>Hnrnpa1</td>
<td>238.1502</td>
<td>249.1101</td>
<td>0.9957</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Gm6654</td>
<td>Rps26</td>
<td>2753.4278</td>
<td>3102.3858</td>
<td>0.9918</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rpl18a</td>
<td>Gm15427</td>
<td>1432.3802</td>
<td>1453.9128</td>
<td>0.9916</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Gm10333</td>
<td>Nutf2</td>
<td>93.5194</td>
<td>96.4387</td>
<td>0.9873</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Bex4</td>
<td>Bex1</td>
<td>141.7229</td>
<td>158.0351</td>
<td>0.9859</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>r_B2_Mm1t</td>
<td>r_B2_Mm1a</td>
<td>30234.6271</td>
<td>46666.4581</td>
<td>0.9846</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>r_(T)n</td>
<td>r_(A)n</td>
<td>12515.2129</td>
<td>12656.7597</td>
<td>0.9840</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

2.1.4.1 Plot a co-expression network. We will demonstrate how to plot the figure "networkplot.png" here.

```r
library(igraph)

# Attaching package: 'igraph'
# The following objects are masked from 'package:stats':
# decorate, spectrum
# The following object is masked from 'package:base':
# union
Thislayout <- layout_with_kk(results$igraph)
plot(results$igraph,
    # Edge color when correlation is positive
    plot.Pos.cor.col="red",
    # Edge color when correlation is negative
    plot.Neg.cor.col="green")
```
#Vertex size
vertex.size=1.5,
#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
dge.width=0.05,
#Use the layout created previously
layout=Thislayout
)
2.1.4.2 Identify genes that belong to a cluster. For example, what are the genes that belong to the same cluster as the Mmp2 gene? a. Identify the cluster that Mmp2 belongs to.

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

b. 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.
```

2.1.4.3 Draw a correlation heatmap.

1. Randomly choose 100 genes from clustering results

```
top100 <- results$Cluster[sample(1:nrow(results$Cluster), 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",
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),
mat=rbind(c(4, 3), c(2, 1)),
Gene name font size
cexRow=0.3,
cexCol=0.3,
Margins
margins = c(8, 8)
)
```
2.1.5 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.

```r
data(Islam2011)
#Identify spike in genes:
SPIKEIN <- rownames(Islam2011)[grep(“Ppia”, rownames(Islam2011)),
grep(“H2afz”, rownames(Islam2011)), grep(“Hprt1”, rownames(Islam2011))]
```

2. Analysis

```r
#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"]),][1:10,5:7]

# Users can print the gene list by the following command:
# print(Genes10)

<table>
<thead>
<tr>
<th>RNA_SPIKE_5</th>
<th>0.7043</th>
<th>0e+00</th>
<th>0.0000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snrpa</td>
<td>0.5270</td>
<td>0e+00</td>
<td>0.0001</td>
</tr>
<tr>
<td>r_L1M5</td>
<td>0.5134</td>
<td>0e+00</td>
<td>0.0002</td>
</tr>
<tr>
<td>Hmgb2</td>
<td>0.4772</td>
<td>0e+00</td>
<td>0.0007</td>
</tr>
<tr>
<td>r_RLTR13D6</td>
<td>0.4541</td>
<td>0e+00</td>
<td>0.0018</td>
</tr>
<tr>
<td>Rsl1d1</td>
<td>0.4379</td>
<td>0e+00</td>
<td>0.0030</td>
</tr>
<tr>
<td>r_L1M4</td>
<td>0.4012</td>
<td>1e-04</td>
<td>0.0118</td>
</tr>
<tr>
<td>Ankfy1</td>
<td>0.3906</td>
<td>2e-04</td>
<td>0.0125</td>
</tr>
<tr>
<td>Hnrnpd</td>
<td>0.3905</td>
<td>2e-04</td>
<td>0.0125</td>
</tr>
<tr>
<td>Hnrnpf</td>
<td>0.3946</td>
<td>1e-04</td>
<td>0.0125</td>
</tr>
</tbody>
</table>

2.1.5.1 Mean vs SD plot highlighting significant genes.

1. Simply print the plot from results.

print(results$plot$plot)
2.1.6 Hierarchical Clustering

In this section, we will perform hierarchical clustering on Islam2011 data. 1. Obtain data.

```r
data(Islam2011)
```

2. Assign group to samples.

```r
#48 ESC, 44 EF, and 4 NegCtrl
Group <- c(rep("ESC",48),rep("EF",44),rep("NegCtrl",4))
colnames(Islam2011) <- paste(colnames(Islam2011),Group,sep="_")
```

3. Perform Analysis.

```r
#Note that there are 3 known clusters.
HClust.Results <- Linnorm.HClust(Islam2011,Group=Group, num_Clust=3, fontsize=1.8)
```
2.1.6.1 Hierarchical Clustering plot  We can simply print the plot out.

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

2.1.7 Basic Linnorm Transformation

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

1. Linnorm Transformation

```R
library(Linnorm)
Transformed <- Linnorm(Islam2011, minZeroPortion=0)
#minZeroPortion is suggested to be 0 for scRNA-seq.
```

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

```R
#You can use this file with Excel.
write.table(Transformed, "Transformed_Matrix.txt", quote=FALSE, sep="\t", col.names=TRUE, row.names=TRUE)
```
2.1.7.1 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.

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

2. Linnorm Transformation.

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

3. Undo Logarithm from LNormData.

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

4. Calculate Fold Change.

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

#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.
```

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

```r
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\nTCGA Partial LIHC data Paired Tumor vs Adjacent Normal")
legend("topright",legend=paste("mean = ", round(mean(foldchanges),2), "\nStdev = ", round(sd(foldchanges),2)))
```
Probability Density of Fold Change.
TCGA Partial LIHC data
Paired Tumor vs Adjacent Normal

mean = 0.65
Stdev = 1.03
2.2 RnaXSim

2.2.1 RNA-seq Raw Count 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 Sample A.

```r
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[sample(1:nrow(SampleA), 10000), 1:10]
```

2. Simulate an RNA-seq dataset.

```r
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.

```r
# 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 Sample A.

```r
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[sample(1:nrow(SampleA), 10000), 1:10]
```

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

```r
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

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

Answer: Here is an example:

```r
#Obtain a transformed dataset for an example.
data(LIHC)
library(Linnorm)
#LNormData is a matrix of Linnorm Transformed dataset.
LNormData <- Linnorm(LIHC)

#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!