# Package 'Tnseq'

April 13, 2017

**Version** 0.1.2

**Date** 2017-4-13

**Title** Identification of Conditionally Essential Genes in Transposon Sequencing Studies

**Author** Lili Zhao <zhaolili@umich.edu>

**Maintainer** Lili Zhao <zhaolili@umich.edu>

**Imports** methods, limma, edgeR, Ckmeans.1d.dp, Biobase

**Depends** DESeq

**Description** Identification of conditionally essential genes using high-throughput sequencing data from transposon mutant libraries.

**License** GPL (>= 3)

**LazyData** TRUE

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2017-04-13 20:53:45 UTC

## R topics documented:



---

`BiasFactor` *Calculate the bias factor induced by the genomic replication process for each insertion*

---

## Description

The whole genome is first divided into N consecutive regions (windows). The number of read counts is summed within each window and is modelled as a function of the window using the locally weighed scatterplot smoothing (LOESS) function. Finally, we calculate a fitted value for each insertion, and then normalize these fitted values to have the product of these values equal to one. The normalized values are the bias factors(one for each insertion).

## Usage

```
BiasFactor(Location, Count, window=10000)
```

## Arguments

| | |
|---|---|
| Location | a numeric vector specifying insertion locations. |
| Count | a numeric vector of read counts over locations. |
| window | the number of regions to divide the whole genome. |

## Value

| | |
|---|---|
| bias.factor | bias factor for each insertion. |
| yfit | the fitted count value for each insertion. |
| w | window id |
| yw | total read counts over windows. |
| ywfit | the fitted count value for each window. |

## Author(s)

Lili Zhao <zhaolili@umich.edu>

## Examples

```
## Not run:
data(serratia)
countData=serratia[,-c(1,2,3)]
location=serratia$Loc


# obtain bias factor for the first sample
foo=BiasFactor(location, countData[,1],  window=10000)

# obtain the adjusted counts after the bias correction
countData.adj=countData[,1]/foo$bias.factor



## End(Not run)
```

`serratia` *serratia data example*

## Description

a small subset of the real serratia transposon sequencing data. serratia is a data frame containing insertion count data from two inoculum pools, and each pool has 6 samples (2 input and 4 output samples).

## Usage

```
data("serratia")
```

## Format

A data frame with 2442 observations on the following 15 variables.

**GeneID** gene name

**Length** gene length

**Loc** insertion locations

**SM_control_lib1_rep1** first input sample from pool 1

**SM_control_lib1_rep2** second input sample from pool 1

**SM_control_lib2_rep1** first input sample from pool 2

**SM_control_lib2_rep2** second input sample from pool 2

**SM_target_lib1_rep1** first output sample from pool 1

**SM_target_lib1_rep2** second output sample from pool 1

**SM_target_lib1_rep3** third output sample from pool 1

**SM_target_lib1_rep4** forth output sample from pool 1

**SM_target_lib2_rep1** first output sample from pool 2

**SM_target_lib2_rep2** second output sample from pool 2

**SM_target_lib2_rep3** third output sample from pool 2

**SM_target_lib2_rep4** forth output sample from pool 2

## Examples

```
data(serratia)
```

`TnseqDiff` *Identify conditionally essential genes using high-throughput sequencing data from transposon mutant libraries*

## Description

It utilizes two steps to estimate the essentiality for each gene in the genome. First, it construcsts a confidence distribution (CD) function containing evidence of conditional essentiality for each insertion by comparing read counts of that insertion between conditions. Second, it combines insertion-level CD functions to infer the essentiality for the corresponding gene. (Zhao et al., 2017). If no replicate in both conditions in any pool, the insertion-level p-values are calculated from the DESeq (using method="blind" and sharingMode="fit-only"), and then combined using the stouffer method.

## Usage

```
TnseqDiff(countData, geneID, location, pool, condition, weights="equal", bayes=FALSE,
        p.nb=FALSE, norm=TRUE, cut=0)
```

## Arguments

| | |
|---|---|
| `countData` | a read count matrix. It has n rows (each row corresponding to a unique insertion site) and m columns (each column corresponding to a sample). |
| `geneID` | a character string of gene names for the insertion sites in countData. |
| `location` | a numeric vector specifying insertion locations. |
| `pool` | a numeric vector specifying the pool id that each sample in countData belongs to. |
| `condition` | a character string specifying the condition that each sample in countData belongs to. |
| `weights` | a character string specifying weights for the insertion sites. These weights are used to weight the insertion-level evidence for the combination. It must be `"equal"` (default) or `"hc"`; `"equal"` assumes equal weights for all insertions, and `"hc"` generates small weights for low insertion counts in the input condition. These weights are determined using the algorithm described in (Wang and Song, 2001). |
| `bayes` | a logical indicating whether moderated estimates are used to construct the CD function. |
| `p.nb` | a logical requesting a p-value based on a negative binomial model. If `TRUE`, the insertion-level p-values are derived from the DESeq (using the default setting), and then combined using the stouffer method. |
| `norm` | a logical indicating whether normalization is performed. If `TRUE` (default), TMM (trimmed mean of M values) normalization is used. |
| `cut` | an insertion is excluded from the analysis if the total read counts over all samples is less than cut. `cut=0` (Default) include all data. |

## Value

There are two output files, one is named as resTable, and the other is named as est.insertion. The resTable file is a data frame, which contains the following columns:

| | |
|---|---|
| `ID` | gene names |
| `NumInser` | number of insertions for each gene in input condition. |
| `Unique.NumInser` | number of unique insertions for each gene in input condition. If there is only one pool, Unique.NumInser should be equal to NumInser. |
| `Mean.x` | averaged counts for samples in condition x. The average is calculated after counts are normalized in DESeq. |
| `FC` | the fold change of condition B over A derived from the combined CD function (if replicates=FALSE, the FC is the ratio of the total counts within the gene). |
| `logFC` | log2 fold change of group B over group A. |
| `pvalue` | p-values indicating significances of the differential tests. |

The est.insertion file is a data frame when there are replicates in all pools (for advanced users), and it contains estimates from the linear model for each insertion:

| | |
|---|---|
| `ID` | gene names |
| `Mean1,Mean2` | means of the two conditions. |
| `df` | degrees of freedom for the slope estimate. |
| `bhat` | slope estimate (logFC). |
| `se` | standard error of the slope estimate. |

## Author(s)

Lili Zhao <zhaolili@umich.edu>

## References

Wang, H. and Song, M. (2001). Ckmeans.1d.dp: optimal k-means clustering in one dimension by dyanamic programing. *The R Journal*, 3(2), 29-33.

Zhao, L., Wu, W., Anderson, M. T., Li, Y. Mobley, H. L. T., and Bachman, M. A.(2017). Inseq: Identification of Conditionally Essential Genes in Transposon Sequencing Studies. Submitted to BMC Bioinformatics.

## Examples

```
data(serratia)

# Test the first 100 insertions
serr=serratia[1:100,]
# obtain the count matrix
countData=serr[,-c(1,2,3)]
```

```
condition=c(rep("Input",4),rep("Output",8))
pool=c(1,1,2,2,1,1,1,1,2,2,2,2)
geneID=as.character(serr$GeneID)
location=serr$Loc

foo<-TnseqDiff(countData, geneID, location, pool, condition)

res=foo$resTable

# adjust pvalues using Benjamini & Hochberg
res$padj=p.adjust(res$pvalue, method = "BH")


## when no replicate in both conditions in each pool
countData=serr[,c(4,6,8,12)]
condition=c("Input","Input","Output","Output")
pool=c(1,2,1,2)
geneID=as.character(serr$GeneID)
location=serr$Loc

foo<-TnseqDiff(countData, geneID, location, pool, condition)

res=foo$resTable


# when there is only one pool
countData=serr[,c(4,5,8,9,10,11)]
condition=c(rep("Input",2),rep("Output",4))
pool=c(1,1,1,1,1,1)
geneID=as.character(serr$GeneID)
location=serr$Loc

foo<-TnseqDiff(countData, geneID, location, pool, condition)

res=foo$resTable
```

# Index