

Package ‘CountClust’

May 23, 2019

Type Package

Title Clustering and Visualizing RNA-Seq Expression Data using Grade of Membership Models

Version 1.12.0

Date 2016-03-13

Maintainer Kushal Dey <kkdey@uchicago.edu>

Description Fits grade of membership models (GoM, also known as admixture models) to cluster RNA-seq gene expression count data, identifies characteristic genes driving cluster memberships, and provides a visual summary of the cluster memberships.

Depends R (>= 3.4), ggplot2 (>= 2.1.0)

URL <https://github.com/kkdey/CountClust>

License GPL (>= 2)

LazyData true

Encoding UTF-8

Imports SQUAREM, slam, maptpx, plyr(>= 1.7.1), cowplot, gtools, flexmix, picante, limma, parallel, reshape2, stats, utils, graphics, grDevices

Suggests knitr, kableExtra, BiocStyle, Biobase, roxygen2, RColorBrewer, devtools, xtable

VignetteBuilder knitr

biocViews ImmunoOncology, RNASeq, GeneExpression, Clustering, Sequencing, StatisticalMethod, Software, Visualization

RoxygenNote 6.0.1

git_url <https://git.bioconductor.org/packages/CountClust>

git_branch RELEASE_3_9

git_last_commit 68a642a

git_last_commit_date 2019-05-02

Date/Publication 2019-05-22

Author Kushal Dey [aut, cre],
Joyce Hsiao [aut],
Matthew Stephens [aut]

R topics documented:

AbundanceGoM	2
BatchCorrectedCounts	2
compare_omega	3
compGoM	5
ex.counts	6
ExtractHighCorFeatures	6
ExtractTopFeatures	7
FitGoM	8
GTExV6Brain.FitGoM	9
handleNA	9
MouseDeng2014.FitGoM	10
MouseJaitinSpleen.FitGoM	10
nullmodel_GoM	11
RemoveSparseFeatures	12
StructureGGplot	12

Index	15
--------------	-----------

AbundanceGoM	<i>GoM model fit for abundance data</i>
--------------	---

Description

GoM model fit for abundance data

Usage

AbundanceGoM

Format

A list of GoM model output

Value

A list of GoM model output

BatchCorrectedCounts	<i>Obtain Batch effect Corrected counts</i>
----------------------	---

Description

This function first converts counts data to log CPM data , then apply a linear model with the batch effect as a factor. We take the sum of intercept, residuals and mean batch effect across all the batches and then inverse transform it back to counts to get rid of batch effects.

Usage

BatchCorrectedCounts(data, batch_lab, use_parallel = TRUE)

Arguments

data count matrix, with samples along the rows and features along the columns.
 batch_lab batch label vector.
 use_parallel if TRUE, we do a parallel analysis over features, else serial application.

Value

Returns a counts data. with same dimension as the input data, but which is corrected for batch_lab.

Examples

```
# Simulation example
N=500;
K=4;
G=100;
Label.Batch=c(rep(1,N/4),rep(2,N/4),rep(3,N/4),rep(4,N/4));
alpha_true=matrix(rnorm((K)*G,0.5,1),nrow=(K));
library(gtools)
tt <- 10;
omega_true = matrix(rbind(rdirichlet(tt*10,c(3,4,2,6)),
                          rdirichlet(tt*10,c(1,4,6,3)),
                          rdirichlet(tt*10,c(4,1,2,2)),
                          rdirichlet(tt*10,c(2,6,3,2)),
                          rdirichlet(tt*10,c(3,3,5,4))), nrow=N);

B=max(Label.Batch);
sigmab_true=2;
beta_true=matrix(0,B,G);
for(g in 1:G)
{
  beta_true[,g]=rnorm(B,mean=0,sd=sigmab_true);
}
read_counts=matrix(0,N,G);
for(n in 1:N){
  for(g in 1:G)
  {
    read_counts[n,g]=rpois(1, omega_true[n,]*%*%exp(alpha_true[,g]
                                                    + beta_true[Label.Batch[n],g]));
  }
}

batchcorrect_counts <- BatchCorrectedCounts(read_counts, Label.Batch,
                                             use_parallel=FALSE)
```

Description

This function computes a re-ordering of the clusters from GoM model fit in one model to make it comparable with that from another. The two models are applied on the same set of samples with same number of clusters, but the features may change from one model to another. The two models may not be of same type as well. One could be a DAPC model, the other a standard topic model. Aids in checking for consistency in topic proportion patterns across multiple GoM methods or across different types of feature sets.

Usage

```
compare_omega(omega1, omega2)
```

Arguments

omega1 cluster membership proportion matrix (N x K) from model 1
 omega2 cluster membership proportion matrix (N x K) from model 2

Value

Returns a list containing

kl.dist A symmetric KL divergence matrix across the re-ordered clusters of two omega matrices
 kl.order_model2_topics re-ordering of the clusters for omega2 to match the clusters for omega1 based on KL divergence
 kl.information_content A measure based on KL information to record how much information in omega2 is explained by omega1. Varies from 0 to 1
 cor.dist A correlation matrix across the re-ordered clusters of two omega matrices
 cor.order_model2_topics re-ordering of the clusters for omega2 to match the clusters for omega1 based on correlation information
 cor.information_content A measure based on correlation information to record how much information in omega2 is explained by omega1. Varies from 0 to 1

Examples

```
tt=10;
omega1=matrix(rbind(gtools::rdirichlet(tt*10,c(3,4,2,6)),
                   gtools::rdirichlet(tt*10,c(1,4,6,3)),
                   gtools::rdirichlet(tt*10,c(4,1,2,2))), nrow=3*10*tt);
omega2=matrix(rbind(gtools::rdirichlet(tt*10,c(1,2,4,6)),
                   gtools::rdirichlet(tt*10,c(1,4,6,3)),
                   gtools::rdirichlet(tt*10,c(3,1,5,2))), nrow=3*10*tt);
out <- compare_omega(omega1, omega2)
```

compGoM	<i>compGoM: compare GoM model fits across K or across different runs through log-likelihood, BIC and null loglikelihood</i>
---------	---

Description

This function takes the `FitGoM/maptpx` fitted model and computes log likelihood, BIC and null model loglikelihood for the fitted GoM models.

Usage

```
compGoM(data, model)
```

Arguments

<code>data</code>	matrix on which GoM model is fitted (samples along rows, genes along columns)
<code>model</code>	<code>FitGoM</code> or <code>maptpx::topics</code> function output (either a class topics or a list of class topics).

Value

`compGoM_models` a vector list that returns the BIC and loglikelihood values for each of the fitted models in `model`.

Examples

```
read.data <- function() {
  x <- tempfile()
  download.file(paste0("https://cdn.rawgit.com/kkdey/",
    "singleCellRNASeqMouseDeng2014",
    "/master/data/Deng2014MouseEsc.rda"),
    destfile = x, quiet = TRUE)
  z <- get(load((x)))
  return(z)
}
Deng2014MouseESC <-read.data()

# Extract observed counts
deng.counts <- Biobase::exprs(Deng2014MouseESC)

# Import GoM fitting results
data("MouseDeng2014.FitGoM")
names(MouseDeng2014.FitGoM)

compGoM(data = t(deng.counts),
  model = MouseDeng2014.FitGoM)
compGoM(data = t(deng.counts),
  model = MouseDeng2014.FitGoM$clust_3)
```

```
ex.counts
```

```
counts data for GTEx V6 Brain data for 200 genes
```

Description

counts data for GTEx V6 Brain data for 200 genes

Usage

```
ex.counts
```

Format

A data frame 1259 by 200 in dimensions

Value

A data frame 1259 by 200 in dimensions

```
ExtractHighCorFeatures
```

```
Extracting most highly correlated genes with GoM topics/clusters
```

Description

This function compares grades of membership profile for each cluster in GoM model fit with the data expression profile to identify genes that are mostly strongly associated with each topic.

Usage

```
ExtractHighCorFeatures(omega, data, num_genes = 100)
```

Arguments

omega	<i>omega</i> matrix, the relative grades of memberships from the GoM model fitting (a $N \times K$ matrix where N is number of samples, K number of topics).
data	$G \times N$ matrix of the expression profile of genes across samples, where G is the number of features and N number of samples
num_genes	The number of top associated genes with each cluster. Defaults to 100

Value

A list containing two items - a $K \times num_genes$ matrix of the top strongly associated/correlated indices/features for K clusters, and another $K \times num_genes$ matrix of the absolute values of the correlations.

ExtractTopFeatures *Extracting top driving genes of GoM clusters*

Description

This function uses relative gene expression profile of the GoM clusters and applies a KL-divergence based method to obtain a list of top features that drive each of the clusters.

Usage

```
ExtractTopFeatures(theta, top_features = 10, method = c("poisson",  
"bernoulli"), options = c("min", "max"), shared = FALSE)
```

Arguments

theta	<i>theta</i> matrix, the relative gene expression profile of the GoM clusters (cluster probability distributions) from the GoM model fitting (a $G \times K$ matrix where G is number of features, K number of topics).
top_features	The top features in each cluster k that are selected based on the feature's ability to distinguish cluster k from cluster $1, \dots, K$ for all cluster $k \neq l$. Default: 10.
method	The underlying model assumed for KL divergence measurement. Two choices considered are "bernoulli" and "poisson". Default: poisson.
options	if "min", for each cluster k , we select features that maximize the minimum KL divergence of cluster k against all other clusters for each feature. If "max", we select features that maximize the maximum KL divergence of cluster k against all other clusters for each feature.
shared	if TRUE, then we report genes that can be highly expressed in more than one cluster. Else, we stick to only those genes that are highest expressed only in a specific cluster.

Value

A matrix ($K \times \text{top_features}$) which tabulates in k -th row the top feature indices driving the cluster k .

Examples

```
data("MouseDeng2014.FitGoM")  
theta_mat <- MouseDeng2014.FitGoM$clust_6$theta;  
top_features <- ExtractTopFeatures(theta_mat, top_features=100, method="poisson", options="min");  
top_features$indices  
top_features$scores
```

FitGoM	<i>Run Grade of Membership (GoM) model with multiple starting points</i> !
--------	---

Description

Fits grade of membership model `FitGoM()` to count data with multiple starting points and choose the best fit using BIC (Bayesian Information Criterion). the multiple starting points ensure that the output is more reliable.

Usage

```
FitGoM(data, K, tol = 0.1, num_trials = 1, options, path_rda = NULL,
        control = list())
```

Arguments

<code>data</code>	counts data $N \times G$, with N , the number of samples along the rows and G , number of genes along columns.
<code>K</code>	the vector of clusters or topics to be fitted. Must be an integer, unlike in <code>JFitGom()</code> . So you need to apply this function separately for each <code>K</code> .
<code>tol</code>	Tolerance value for GoM model absolute log posterior increase at successive iterations (set to 0.1 as default).
<code>num_trials</code>	The number of trials with different starting points used.
<code>options</code>	the measure used to choose best fit, either "BF" or "BIC" measures can be used. BF is more trustworthy, but BIC can be used for better model comparison.
<code>path_rda</code>	The directory path for saving the GoM model output. If NULL, it will return the output to console.
<code>control</code>	Control parameters for the GoM model fits. Same as <code>topics()</code> function of <code>maptpx</code> package.

Value

Outputs the best GoM model fit output for cluster `K` and saves it at the directory path in `path_rda` if the latter is provided.

References

Matt Taddy. On Estimation and Selection for Topic Models. AISTATS 2012, JMLR W&CP 22.

Pritchard, Jonathan K., Matthew Stephens, and Peter Donnelly. Inference of population structure using multilocus genotype data. *Genetics* 155.2 (2000): 945-959.

Examples

```
data("ex.counts")
out <- FitGoM(ex.counts, K=2, tol=100, num_trials=5,
              control=list(tmax=100))
```

GTEXV6Brain.FitGoM	<i>GoM model fit for GTEX V6 Brain bulk-RNA data</i>
--------------------	--

Description

GoM model fit for GTEX V6 Brain bulk-RNA data

Usage

```
GTEXV6Brain.FitGoM
```

Format

A list of GoM model output for k=7

Value

A list of GoM model output for k=7

handleNA	<i>Deal with NAs in the dataset!</i>
----------	--------------------------------------

Description

This function handles the NA values in the count data. If for a feature, the proportion of NAs is greater than threshold proportion, then we remove the feature, otherwise we use MAR substitution scheme using the distribution of the non NA values for the feature. If threshold proportion is 0, it implies removal of all features with NA values. Default value of threshold proportion is 0.

Usage

```
handleNA(data, thresh_prop = 0)
```

Arguments

data	count data in a sample by feature matrix.
thresh_prop	threshold proportion of NAs for removal of feature or replacing the NA values.

Details

This function removes NAs from the counts data

Value

Returns a list with

data	The modified data with NA substitution and removal
na_removed_cols	The columns in the data with NAs that were removed
na_sub_cols	The columns in the data with NAs that were substituted

Examples

```
mat <- rbind(c(2,4,NA),c(4,7,8),c(3,NA,NA));  
handleNA(mat, thresh_prop=0.5)  
handleNA(mat)
```

MouseDeng2014.FitGoM *GoM model fit for Deng et al 2014 single cell RNA-seq data on mouse*

Description

GoM model fit for Deng et al 2014 single cell RNA-seq data on mouse

Usage

```
MouseDeng2014.FitGoM
```

Format

A list of GoM model output for 6 clusters (k=2:7)

Value

A list of GoM model output for 6 clusters (k=2:7)

MouseJaitinSpleen.FitGoM
GoM model fit for Jaitin et al 2014 single cell RNA-seq data on mouse

Description

GoM model fit for Jaitin et al 2014 single cell RNA-seq data on mouse

Usage

```
MouseJaitinSpleen.FitGoM
```

Format

A list of GoM model output for k=7

Value

A list of GoM model output for k=7

nullmodel_GoM	<i>Null models for Grade of Membership (GoM) cluster validation</i>
---------------	---

Description

Use null models (popular in ecology) to generate randomized matrix of counts given the observed data matrix, fit the GoM model to these null matrices and compare the fit on null model data with that on the observed data. Used for validating the GoM clusters

Usage

```
nullmodel_GoM(counts, K, tol = 0.1, null.model = c("frequency", "richness",
  "independentswap", "trialswap"), iter_fill = 1000, iter_randomized = 100,
  plot = TRUE)
```

Arguments

counts	The counts matrix (N x G): N- the number of samples, G- number of features
K	The number of clusters to fit
tol	The tolerance of the GoM model fitted
null.model	The type of nullmodel used (similar to the randomizeMatrix() function argument in picante package)
iter_fill	The number of swaps/fills in each randomized matrix build
iter_randomized	The number of randomization matrices generated
plot	If TRUE, plots density of log Bayes factor

Value

Returns a list with

GoMBF.obs	log BF for the observed counts with K=2 against the null with no clusters
GoMBF.rand	a vector of log BF for each randomized count matrix with K=2 against the null with no clusters
pval	the p-value of the observed log Bayes factor against the ones from randomized matrices

Examples

```
data("ex.counts")
nullmodel_GoM(ex.counts,
  K=2,
  tol=500,
  null.model="frequency",
  iter_randomized=3,
  plot=FALSE)
```

RemoveSparseFeatures *Removes features with a lot of 0 counts*

Description

This function deals with zero counts in the counts dataset. If for a feature, the proportion of zeros across the samples is greater than `filter_prop`, then we remove the feature.

Usage

```
RemoveSparseFeatures(data, filter_prop = 0.9)
```

Arguments

`data` count data in a sample by feature matrix.
`filter_prop` threshold proportion. If the proportion of zeros for the feature exceeds this threshold then we remove the feature altogether. Default is 0.9.

Value

Returns a list with

`data` filtered data with sparse features removed
`sparse_features` the feature names of the features found sparse and removed

Examples

```
mat <- rbind(c(2,0,3,0,4),c(4,5,5,0,0),c(30,34,63,25,0),c(0,0,0,0,0));
RemoveSparseFeatures(mat, filter_prop = 0.5)
RemoveSparseFeatures(mat)
```

StructureGGplot *Structure plot using ggplot2*

Description

Make the traditional Structure plot of GoM model with ggplot2

Usage

```
StructureGGplot(omega, annotation = NULL,
  palette = RColorBrewer::brewer.pal(8, "Accent"), figure_title = "",
  yaxis_label = "Tissue type", order_sample = TRUE,
  sample_order_decreasing = TRUE, sample_order_opts = 1,
  split_line = list(split_lwd = 1, split_col = "white"), plot_labels = TRUE,
  axis_tick = list(axis_ticks_length = 0.1, axis_ticks_lwd_y = 0.1,
  axis_ticks_lwd_x = 0.1, axis_label_size = 3, axis_label_face = "bold"),
  legend_title_size = 8, legend_key_size = 0.4, legend_text_size = 5)
```

Arguments

<code>omega</code>	Cluster membership probabilities of each sample. Usually a sample by cluster matrix in the Topic model output. The cluster weights sum to 1 for each sample.
<code>annotation</code>	data.frame of two columns: <code>sample_id</code> and <code>tissue_label</code> . <code>sample_id</code> is a vector consisting of character type of variable, which indicates the unique identifying number of each sample. <code>tissue_label</code> is a vector consisting of factor type of variable, which indicates the sample phenotype that is to be used in sorting and grouping the samples in the Structure plot; for example, tissue of origin in making Structure plot of the GTEx samples. Default is set to "none" for when no phenotype information is used to order the sample vectors.
<code>palette</code>	Colors assigned to label the clusters. The first color in the palette is assigned to the cluster that is labeled 1 (usually arbitrarily assigned during the clustering process). Note: The number of colors must be the same or greater than the number of clusters. When the number of clusters is greater than the number of colors, the clusters that are not assigned a color are filled with white in the figure. The recommended choice of color palette is <code>RColorBrewer</code> , for instance <code>RColorBrewer::brewer.pal(8, "Accent")</code> or <code>RColorBrewer::brewer.pal(9, "Set1")</code> .
<code>figure_title</code>	Title of the plot.
<code>yaxis_label</code>	Axis label for the phenotype used to order the samples, for example, tissue type or cell type.
<code>order_sample</code>	Whether to order the samples that are of the same tissue label or phenotype label, that is, having the same label in the <code>tissue_label</code> variable. If <code>TRUE</code> , we order samples that are of the same phenotype label and sort the samples by membership of most representative cluster. If <code>FALSE</code> , we keep the order in the data.
<code>sample_order_decreasing</code>	If <code>order_sample=TRUE</code> , then order the sample in descending (<code>TRUE</code>) or ascending order.
<code>sample_order_opts</code>	Orders by different choices of clusters in a batch. Can take the values 1, 2, 3 or 4 corresponding to 4 ordering options. Default equal to 1.
<code>split_line</code>	Control parameters for the line that separates phenotype subgroups in the plot.
<code>plot_labels</code>	If <code>TRUE</code> , the plot the axis labels.
<code>axis_tick</code>	Control parameters for x-axis and y-axis tick sizes.
<code>legend_title_size</code>	The size of the title of the Structure Plot representation.
<code>legend_key_size</code>	The size of the legend key in Structure plot.
<code>legend_text_size</code>	the size specification of the legend text.

Value

Plots the Structure plot visualization of the GoM model

Examples

```
data("MouseDeng2014.FitGoM")
```

```

# extract the omega matrix: membership weights of each cell
names(MouseDeng2014.FitGoM$clust_6)
omega <- MouseDeng2014.FitGoM$clust_6$omega
tissue_label <- rownames(omega)

# make annotation matrix
annotation <- data.frame(
  sample_id = paste0("X", c(1:NROW(omega))),
  tissue_label = factor(rownames(omega),
    levels = rev( c("zy", "early2cell",
      "mid2cell", "late2cell",
      "4cell", "8cell", "16cell",
      "earlyblast", "midblast",
      "lateblast") ) ) )

head(annotation)

# setw rownames of omega to be sample ID
rownames(omega) <- annotation$sample_id

StructureGGplot(omega = omega,
  annotation = annotation,
  palette = RColorBrewer::brewer.pal(8, "Accent"),
  yaxis_label = "development phase",
  order_sample = TRUE,
  axis_tick = list(axis_ticks_length = .1,
    axis_ticks_lwd_y = .1,
    axis_ticks_lwd_x = .1,
    axis_label_size = 7,
    axis_label_face = "bold"))

```

Index

- *Topic **GoM**,
 - compGoM, 5
 - *Topic **Structure**
 - FitGoM, 8
 - *Topic **batch**
 - BatchCorrectedCounts, 2
 - *Topic **clustering**,
 - FitGoM, 8
 - *Topic **clustering**
 - handleNA, 9
 - *Topic **counts**
 - BatchCorrectedCounts, 2
 - FitGoM, 8
 - handleNA, 9
 - RemoveSparseFeatures, 12
 - *Topic **data**,
 - BatchCorrectedCounts, 2
 - FitGoM, 8
 - handleNA, 9
 - RemoveSparseFeatures, 12
 - *Topic **datasets**
 - AbundanceGoM, 2
 - ex.counts, 6
 - GTEXV6Brain.FitGoM, 9
 - MouseDeng2014.FitGoM, 10
 - MouseJaitinSpleen.FitGoM, 10
 - *Topic **effect**
 - BatchCorrectedCounts, 2
 - *Topic **extraction**
 - RemoveSparseFeatures, 12
 - *Topic **feature**
 - RemoveSparseFeatures, 12
 - *Topic **fit**
 - compGoM, 5
 - *Topic **model**
 - compGoM, 5
 - *Topic **plot**
 - FitGoM, 8
- AbundanceGoM, 2
- BatchCorrectedCounts, 2
- compare_omega, 3
- compGoM, 5
- ex.counts, 6
- ExtractHighCorFeatures, 6
- ExtractTopFeatures, 7
- FitGoM, 8
- GTEXV6Brain.FitGoM, 9
- handleNA, 9
- MouseDeng2014.FitGoM, 10
- MouseJaitinSpleen.FitGoM, 10
- nullmodel_GoM, 11
- RemoveSparseFeatures, 12
- StructureGGplot, 12