Package ‘CountClust’

March 22, 2017

Type Package
Title Clustering and Visualizing RNA-Seq Expression Data using Grade of Membership Models
Version 1.1.4
Date 2016-11-05
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 (>= 3.3.0), ggplot2 (>= 2.1.0)
URL https://github.com/kkdey/CountClust
License GPL (>= 2)
LazyData true
Encoding UTF-8
Imports maptpx, slam, plyr(>= 1.7.1), cowplot, gtools, flexmix, picante, limma, parallel, reshape2, stats, utils, graphics, grDevices
Suggests knitr, BiocStyle, Biobase, roxygen2, RColorBrewer, devtools, xtable
VignetteBuilder knitr
biocViews RNASeq, GeneExpression, Clustering, Sequencing, StatisticalMethod, Software, Visualization
RoxygenNote 5.0.1
NeedsCompilation no
Author Kushal Dey [aut, cre], Joyce Hsiao [aut], Matthew Stephens [aut]

R topics documented:

AbundanceGoM ................................................................. 2
BatchCorrectedCounts ....................................................... 2
cg_topics ........................................................................... 3
AbundanceGoM  

GoM model fit for abundance data

**Description**

GoM model fit for abundance data

**Usage**

`AbundanceGoM`

**Format**

A list of GoM model output

**Value**

A list of GoM model output

---

BatchCorrectedCounts  

Obtain Batch effect Corrected counts

**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)`
### cg_topics

**Compute the center of gravity of the clusters**

#### Description

This function computes the center of gravity for each cluster by taking weighted mean of each component of features where the weights are determined from the theta matrix of the topic model fit.

#### 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

```r
# 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)

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)
```
**Usage**

```r
cg_topics(theta, feature.comp)
```

**Arguments**

- `theta`: The cluster probability distribution/theta matrix obtained from the GoM model fitting (it is a \( G \times K \) matrix where \( G \) is number of features, \( K \) number of topics)
- `feature.comp`: the numeric matrix \((G \times L)\) comprising of values for each feature \( g \) and feature metadata \( l \).

**Value**

Returns a matrix of cluster centers of gravity for the \( L \) feature metadata.

**Examples**

```r
N=360;
M=560;
lat <- rep(1:N, M);
long <- rep(1:M, each=N)
comp <- cbind(lat, long);
data(AbundanceGoM)
center_gravity <- cg_topics(AbundanceGoM$theta, comp);
```

---

**compare_omega**  

*Re-ordering cluster membership proportion matrices and Information calculation*

**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**

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

**Arguments**

- `omega1`: cluster membership proportion matrix \((N \times K)\) from model 1
- `omega2`: cluster membership proportion matrix \((N \times K)\) from model 2
**compGoM**

**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**

```r
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)
```

**Description**

This function takes the **FitGoM** output model fits and compute log likelihood, BIC and null model loglikelihood for the GoM models.

**Usage**

```r
compGoM(data, model_output)
```

**Arguments**

- **data**: matrix on which GoM model is fitted (samples along rows, genes along columns)
- **model_output**: **FitGoM** output (a list).
Value

compGoM_models a vector of GoM model fit BIC, loglikelihood and null model loglikelihood for each model in FiGoM model input.

Examples

```r
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_output = MouseDeng2014.FitGoM)
```

---

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**

\[
\text{ExtractHighCorFeatures}(\omega, \text{data}, \text{num\_genes} = 100)
\]

**Arguments**

- \(\omega\): \(\text{omega}\) 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).
- \(\text{data}\): \(G \times N\) matrix of the expression profile of genes across samples, where \(G\) is the number of features and \(N\) number of samples.
- \(\text{num\_genes}\): The number of top associated genes with each cluster. Defaults to 100

**Value**

A list containing two items - a \(K \times \text{num\_genes}\) matrix of the top strongly associated/correlated indices/features for \(K\) clusters, and another \(K \times \text{num\_genes}\) matrix of the absolute values of the correlations.

**Examples**

```r
data("MouseDeng2014.FitGoM")
omega_mat <- MouseDeng2014.FitGoM$clust_6$omega;
read.data1 = function() {
  x = tempfile()
z = get(load((x)))
return(z)
}
Deng2014MouseESC <- read.data1()
deng.counts <- Biobase::exprs(Deng2014MouseESC)
out <- ExtractHighCorFeatures(omega_mat, deng.counts, num_genes=10)
```

### ExtractTopFeatures

**Extracting top driving genes driving 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"))

Arguments

theta       theta 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 $1, \ldots, K$ 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.

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");

FitGoM

Grade of Membership (GoM) model fit!

Description

Fits a grade of membership model to count data. Default input includes a sample-by-feature matrix, the number of clusters (topics) to fit ($K$). The function is a wrapper of the topics() function implemented in Matt Taddy’s maptpx package.

Usage

FitGoM(data, K, tol = 0.1, path_rda = NULL, control = list())

Arguments

data counts data $N \times G$, with $N$, the number of samples along the rows and $G$, number of genes along columns.

K the vector of clusters or topics to be fitted.
tol

Tolerance value for GoM model absolute log posterior increase at successive iterations (set to 0.1 as default).

path_rda

The directory path for saving the GoM model output. If NULL, it will return the output to console.

control

Control parameters. Same as topics() function of maptpx package.

Value

Saves the GoM model fit output for each cluster in vector K at the directory path in path_rda.

References


Examples

data("ex.counts")
out <- FitGoM(ex.counts, K=4, tol=100, control=list(tmax=100))
handleNA  

*Deal with NAs in the dataset!*

**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**

```r
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**

```r
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**

```r
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)

---

**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**

*Null models for Grade of Membership (GoM) cluster validation*

**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)`
**RemoveSparseFeatures**

**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.
StructureGGplot

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

```r
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

```r
StructureGGplot(omega, annotation = NULL, palette = RColorBrewer::brewer.pal(8, "Accent"), figure_title = "", yaxis_label = "Tissue type", order_sample = TRUE, sample_order_decreasing = TRUE, 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"))
```

Arguments

- `omega`: 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.
- `annotation`: data.frame of two columns: sample_id and tissue_label. sample_id is a vetor consisting of character type of variable, which indicates the unique identifying number of each sample. tissue_label 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 Structre plot; for example, tissue of origin in making Structure plot of the GTEx samples. Default is set to NULL for when no phenotype information is used to order the sample vectors.
- `palette`: 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 RColorBrewer, for instance RColorBrewer::brewer.pal(8, "Accent") or RColorBrewer::brewer.pal(9, "Set1").
- `figure_title`: Title of the plot.
StructureGGplot

yaxis_label  Axis label for the phenotype used to order the samples, for example, tissue type or cell type.

order_sample  Whether to order the samples that are of the same tissue label or phenotype label, that is, having the same label in the tissue_label variable. If TRUE, we order samples that are of the same phenotype label and sort the samples by membership of most representative cluster. If FALSE, we keep the order in the data.

sample_order_decreasing  If order_sample=TRUE, then order the sample in descending (TRUE) or ascending order.

split_line  Control parameters for the line that separates phenotype subgroups in the plot.

plot_labels  If TRUE, the plot the axis labels.

axis_tick  Control parameters for x-axis and y-axis tick sizes.

Value
Plots the Structure plot visualization of the GoM model

Examples

# Example 1
data("MouseDeng2014.FitGoM")

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

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")

# Example 2
# Import Deng et al data

# function to read Deng data from GitHub
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()
deng.counts <- Biobase::exprs(Deng2014MouseESC)
deng.meta_data <- Biobase::pData(Deng2014MouseESC)
deng.gene_names <- rownames(deng.counts)
samples_subvector <- which(!duplicated(deng.meta_data$cell_type))[1:3]
  # Fit GoM on 3 samples with K = 3
  fit_k3 <- FitGoM( t(deng.counts[, samples_subvector]),
    K = 3, tol=0.1)
  names(fit_k3$clust_3)
  omega <- fit_k3$clust_3$omega
  # make annotation matrix
  annotation <- data.frame(
    sample_id = paste0("X", c(1:NROW(omega))),
    tissue_label = factor( as.character(deng.meta_data$cell_type[samples_subvector]),
      levels = rev( as.character(deng.meta_data$cell_type[samples_subvector]) ) )
  )
  rownames(omega) <- annotation$sample_id
  StructureGGplot(omega = omega,
    annotation = annotation,
    palette = RColorBrewer::brewer.pal(3, "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, 10
*Topic **counts**
  BatchCorrectedCounts, 2
  FitGoM, 8
  handleNA, 10
  RemoveSparseFeatures, 12
*Topic **data,**
  BatchCorrectedCounts, 2
  FitGoM, 8
  handleNA, 10
  RemoveSparseFeatures, 12
*Topic **datasets**
  AbundanceGoM, 2
  ex.counts, 6
  GTExV6Brain.FitGoM, 9
  MouseDeng2014.FitGoM, 10
  MouseJaitinSpleen.FitGoM, 11
*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
cg_topics, 3

compare_omega, 4
compGoM, 5
ex.counts, 6
ExtractHighCorFeatures, 7
ExtractTopFeatures, 7
FitGoM, 8
GTExV6Brain.FitGoM, 9
handleNA, 10
MouseDeng2014.FitGoM, 10
MouseJaitinSpleen.FitGoM, 11
nullmodel_GoM, 11
RemoveSparseFeatures, 12
StructureGGplot, 13