Package: DCATS (via r-universe)

June 30, 2024

Type Package

Title Differential Composition Analysis Transformed by a Similarity matrix

Version 1.3.0

Description Methods to detect the differential composition abundances between conditions in singel-cell RNA-seq experiments, with or without replicates. It aims to correct bias introduced by missclaisification and enable controlling of confounding covariates. To avoid the influence of proportion change from big cell types, DCATS can use either total cell number or specific reference group as normalization term.

Depends R (>= 4.1.0), stats

License MIT + file LICENSE

Imports MCMCpack, matrixStats, robustbase, aod, e1071

Suggests testthat (>= 3.0.0), knitr, Seurat, SeuratObject, tidyverse, rmarkdown, BiocStyle

VignetteBuilder knitr

RoxygenNote 7.2.3

biocViews SingleCell, Normalization

Encoding UTF-8

Config/testthat/edition 3

Repository https://bioc.r-universe.dev

RemoteUrl https://github.com/bioc/DCATS

RemoteRef HEAD

RemoteSha 1e756e6bde0c590768135a65025dc720da199120

Contents

`create_simMat` *Generate similarity matrix with uniform confusion rate to none-self clusters*

Description

Create a similarity matrix assuming the misclassification error distribute uniformly in all clusters

Usage

```
create_simMat(K, confuse_rate)
```

Arguments

`K`	A integer for number of cluster
`confuse_rate`	A float for confusion rate, uniformly to none-self clusters

Value

a similarity matrix with uniform confusion with other cluster

Examples

```
create_simMat(4, 0.1)
```

dcats_GLM *A Generalised linear model based likelihood ratio testing*

Description

GLM supports both beta-binomial and negative binomial from aod package.

Usage

```
dcats_GLM(
  count_mat,
  design_mat,
  similarity_mat = NULL,
  pseudo_count = NULL,
  base_model = "NULL",
  fix_phi = NULL,
  reference = NULL
)
```

Arguments

count_mat	A matrix of composition sizes (n_sample, n_cluster) for each cluster in each sample
design_mat	A matrix or a data frame of testing candidate factors (n_sample, n_factor) with same sample order as count_mat. All factors should be continous and categorical with only two levels.
similarity_mat	A matrix of floats (n_cluster, n_cluster) for the similarity matrix between cluster group pair. The order of cluster should be consistent with those in 'count_mat'.
pseudo_count	A pseudo count to add for counts in all cell types Default NULL means 0 except if a cell type is empty in one condition, otherwise pseudo_count will be: 0.01 * rowMeans for each condition
base_model	A string value: 'NULL' for 1 factor vs NULL factor testing; 'FULL' for FULL factors vs n-1 factors testing.
fix_phi	A numeric used to provided a fixed phi value for the GLM for all cell types
reference	A vector of characters indicating which cell types are used as reference for normalization. 'NULL' indicates using total count for normalization.

Value

a list of significance p values for each cluster

Examples

```
K <- 3
totals1 = c(100, 800, 1300, 600)
totals2 = c(250, 700, 1100)
diri_s1 = rep(1, K) * 20
diri_s2 = rep(1, K) * 20
simil_mat = DCATS::create_simMat(K, confuse_rate=0.2)
sim_dat <- DCATS::simulator_base(totals1, totals2, diri_s1, diri_s2, simil_mat)
sim_count = rbind(sim_dat$numb_cond1, sim_dat$numb_cond2)
sim_design = data.frame(condition = c("g1", "g1", "g1", "g1", "g2", "g2", "g2"),
gender = sample(c("Female", "Male"), 7, replace = TRUE))
## Using 1 factor vs NULL factor testing
dcats_GLM(sim_count, sim_design, similarity_mat = simil_mat)
## Using full factors vs n-1 factors testing with intercept term
dcats_GLM(sim_count, sim_design, similarity_mat = simil_mat, base_model='FULL')
## Fix phi
dcats_GLM(sim_count, sim_design, similarity_mat = simil_mat, fix_phi = 1/61)
## Specify reference cell type
colnames(sim_count) <- c("celltypeA", "celltypeB", "celltypeC")
```

`detect_reference` *Calculate a global phi for all cell types*

Description

Assuming all cell types share the same phi. This global phi can be calculate by pooling all cell types together to fit a beta binomial distribution.

Usage

```
detect_reference(count_mat, design_mat, similarity_mat = NULL, fix_phi = NULL)
```

Arguments

`count_mat`	A matrix of composition sizes (n_sample, n_cluster) for each cluster in each sample.
`design_mat`	A matrix or a data frame of testing candidate factors (n_sample, n_factor) with same sample order as count_mat. All factors should be continous and categorical with only two levels.
`similarity_mat`	A matrix of floats (n_cluster, n_cluster) for the similarity matrix between cluster group pair. The order of cluster should be consistent with those in 'count_mat'.
`fix_phi`	A numeric used to provided a fixed phi value for the GLM for all cell types.

Value

A data frame with ordered cell types and their p-value. Cell types are ordered by their p-values. The order indicating how they are recommended to be selected as reference cell types.

Examples

```
K <- 3
totals1 = c(100, 800, 1300, 600)
totals2 = c(250, 700, 1100)
diri_s1 = rep(1, K) * 20
diri_s2 = rep(1, K) * 20
simil_mat = DCATS::create_simMat(K, confuse_rate=0.2)
sim_dat <- DCATS::simulator_base(totals1, totals2, diri_s1, diri_s2, simil_mat)
sim_count = rbind(sim_dat$numb_cond1, sim_dat$numb_cond2)
sim_design = data.frame(condition = c("g1", "g1", "g1", "g1", "g2", "g2", "g2"),
gender = sample(c("Female", "Male"), 7, replace = TRUE))
## Using 1 factor vs NULL factor testing
detect_reference(sim_count, sim_design)
```

---

getPhi *Calculate a global phi for all cell types*

Description

Assuming all cell types share the same phi. This global phi can be calculate by pooling all cell types together to fit a beta binomial distribution.

Usage

```
getPhi(count_mat, design_mat)
```

Arguments

count_mat	A matrix of composition sizes (n_sample, n_cluster) for each cluster in each sample
design_mat	A matrix of testing candidate factors (n_sample, n_factor) with same sample order as count_mat

Value

A number indicating a global phi for all cell types

Examples

```
K <- 3
totals1 = c(100, 800, 1300, 600)
totals2 = c(250, 700, 1100)
diri_s1 = rep(1, K) * 20
diri_s2 = rep(1, K) * 20
simil_mat = DCATS::create_simMat(K, confuse_rate=0.2)
sim_dat <- DCATS::simulator_base(totals1, totals2, diri_s1, diri_s2, simil_mat)
sim_count = rbind(sim_dat$numb_cond1, sim_dat$numb_cond2)
sim_design = data.frame(condition = c("g1", "g1", "g1", "g1", "g2", "g2", "g2"))
phi = DCATS::getPhi(sim_count, sim_design)
```

`Haber2017` *Count matrices of intestinal epithelial scRNA-seq data from three conditions*

Description

A data containing the count matrices, the similarity matrix and other variables used to generate the similarity matrix from intestinal epithelial single cell RNA sequencing data with three condition. Count matrices are calculated based on the number of cells in each cell type. The similarity matrix is calculated by support vector machine classifiers using 5-fold cross validation. Top 30 PCs are used as predictors.

Usage

```
Haber2017
```

Format

A list with 7 items:

count_ctrl the count matrix for the control group

count_Hpoly3 the count matrix for three days after H.polygyrus infection

count_Hpoly10 the count matrix for ten days after H.polygyrus infection

count_Salma the count matrix for two days after Salmonella infection

svm_mat the similarity matrix

source the source of this dataset

Source

https://www.nature.com/articles/nature24489

Examples

```
library(DCATS)
data(Haber2017)
```

Kang2017 *Count matrices of 8 pooled lupus patient samples within two conditions*

Description

A data containing the count matrices, the similarity matrix and other variables used to generate the similarity matrix from single cell RNA sequencing data of 8 pooled lupus patient samples within two conditions. Count matrices are calculated based on the number of cells in each cell type. The similarity matrix is calculated by support vector machine classifiers using 5-fold cross validation. Top 30 PCs are used as predictors. The svmDF contains 30 PCs generated by standard Seurat pipeline and the condition, cell type information collected from the original paper.

Usage

```
Kang2017
```

Format

A list with 5 items:

count_ctrl the count matrix for three days after H.polygyrus infection

count_stim the count matrix for ten days after H.polygyrus infection

svm_mat the simularity matrix

svmDF the data frame used to calculate the similarity matrix.

source the source of this dataset

Source

https://www.nature.com/articles/nbt.4042

knn_simMat *Calculate stochastic transition matrix between clusters from a KNN connection matrix*

Description

The transition probability from cluster i to j is the fraction of neighbours of all samples in cluster i that belongs to cluster j. Note, this matrix is asymmetric, so as the input KNN connection matrix.

Usage

```
knn_simMat(KNN_matrix, clusters)
```

Arguments

KNN_matrix	a sparse binary matrix with size (n_sample, n_sample). x_ij=1 means sample j is a neighbour of sample i. As definition, we expect sum(KNN_matrix) = n_sample * K, where K is the number neighbours.
clusters	a (n_sample,) vector of cluster id for each sample.

Value

a similarity matrix calculated based on the knn graph.

Examples

```
data(simulation)
knn_mat = knn_simMat(simulation$knnGraphs, simulation$labels)
```

---

multinom_EM *An EM algorithm to fit a multinomial with maximum likelihood*

Description

An EM algorithm to fit a multinomial with maximum likelihood

Usage

```
multinom_EM(X, simMM, min_iter = 10, max_iter = 1000, logLik_threshold = 0.01)
```

Arguments

X	A vector of commopent sizes
simMM	A matrix of floats (n_cluster, n_cluster) for the similarity matrix between clusters. simMM[i,j] means the proportion of cluster i will be assigned to cluster j, hence colSums(simMM) are ones.
min_iter	integer(1). number of minimum iterations
max_iter	integer(1). number of maximum iterations
logLik_threshold	A float. The threshold of logLikelihood increase for detecting convergence

Value

a list containing mu, a vector for estimated latent proportion of each cluster, logLik, a float for the estimated log likelihood, simMM, the input of simMM, codeX, the input of X, X_prop, the proportion of clusters in the input X, predict_X_prop, and the predicted proportion of clusters based on mu and simMM.

Examples

```
X = c(100, 300, 1500, 500, 1000)
simMM = create_simMat(5, confuse_rate=0.2)
multinom_EM(X, simMM)
```

`Ren2021` *Count matrix and metadata of a large COVID-19 scRNA-seq data cohort.*

Description

A data containing the count matrix, metadata from a large COVID-19 cohort. Count matrices are calculated based on the number of cells in each cell type. The information in the design matrix is collected in the original paper.

Usage

```
Ren2021
```

Format

A list with 7 items:

countM the count matrix for all samples comming from different condition

designM the corresponding metadata related to each sample

source the source of this dataset

Source

https://www.nature.com/articles/nature24489

`simulation` *Simulated dataset with two conditions*

Description

A data containing the count matrices, the similarity matrix and other variables used to generate the similarity matrix from a simulated single cell RNA sequencing data with two conditions. Dirichlet distribution was used to generate a proportion vector for cell types based on the defined true proportions. Multinomial distribution was used to generate simulated cell numbers. Cells were selected from cell pools based on cell numbers and gene expression matrices were processed by Seurat. The count matrix were calculated by the clustering results, and the similarity matrix was calculated using knn graph. The labels are the groud true annotation of each cell.

Usage

```
simulation
```

Format

A list with 5 items:

numb_cond1 the count matrix of condition 1

numb_cond2 the count matrix of condition 2

knn_mat the similariy matrix

knnGraphs the knn graphs information used to calculate the similarity matrix.

labels the clusters' label for each simulated single cell

---

simulator_base *Composition size simulator*

Description

Directly simulating the composition size from a Dirichlet-Multinomial distribution with replicates for two conditions.

Usage

```
simulator_base(
  totals_cond1,
  totals_cond2,
  dirichlet_s1,
  dirichlet_s2,
  similarity_mat = NULL
)
```

Arguments

totals_cond1	A vector of integers (n_rep1,) for the total samples in each replicate in codition 1
totals_cond2	A vector of integers (n_rep2,) for the total samples in each replicate in codition 2
dirichlet_s1	A vector of floats (n_cluster,) for the composition concentration in condition 1
dirichlet_s2	A vector of floats (n_cluster,) for the composition concentration in condition 2
similarity_mat	A matrix of floats (n_cluster, n_cluster) for the similarity matrix between each cluster pair

Value

a list of two matrices for composition sizes in each replicate and each cluster in both conditions.

Examples

```
K <- 2
totals1 = c(100, 800, 1300, 600)
totals2 = c(250, 700, 1100)
diri_s1 = rep(1, K) * 20
diri_s2 = rep(1, K) * 20
confuse_rate = 0.2
simil_mat = create_simMat(2, 0.2)
sim_dat <- simulator_base(totals1, totals2, diri_s1, diri_s2, simil_mat)
```

`svm_simMat` *Calculate stochastic transition matrix between clusters from a data frame including information about clustering*

Description

The transition probability from cluster i to j is calculated based on the information used to cluster cells. It is estimated by the misclassification rate from cluster i to j comparing the original labels with the labels predicted by support vector machine with 5-fold cross validation.

Usage

```
svm_simMat(dataframe)
```

Arguments

`dataframe`	a data frame contains the information used for clustering and the original label of each cell. The original labels should have the column name 'clusterRes'.

Value

a similarity matrix estimated by 5-fold cross validation support vector machine.

Examples

```
data(Kang2017)
svm_mat = svm_simMat(Kang2017$svmDF)
```

Index