Package 'NUWA'

May 30, 2022

Type Package

Title A pipeline for robust inference of missing cell markers and deciphering immune cell fractions using mass spectrometry proteomics

Version 0.1.0

Date 2021-2-4

Author Yuhao Xie <xieyuhao@pku.edu.cn>

Maintainer Lihua Cao <lihuacao@bjcancer.org>

Description NUWA is a computational pipeline for robust abundance inference of missing cell type markers in proteomic profiles (by NUWA-ms) and deciphering makeup of immune cell subsets (by NUWA-eDeconv or other popular deconvolution algorithms), which could enable accurate proteomic deconvolution of tissue-infiltrating cell populations. The performance of NUWA pipeline has been systematically evaluated by multiple approaches with validation using scRNA-seq data, see the NUWA manuscript for more details.

License MIT

Encoding UTF-8

LazyData TRUE

RoxygenNote 7.2.0

URL https://github.com/WuOmicsLab/NUWA

Depends R (>= 3.6.1),
e1071 (>= 1.7.3),
preprocessCore

Imports parallel,
abind,
reshape2,
ggplot2,
glmnet,
EPIC,
MCPcounter

Suggests knitr,
roxygen2,
testthat,
rmarkdown

VignetteBuilder knitr

Remotes GfellerLab/EPIC,
ebecht/MCPcounter/Source,
bioc::3.15/preprocessCore,
bioc::3.15/GSEABase,
bioc::3.15/GSVA,
dviraran/xCell

R topics documented:

`barplotCF` *Stacked barplot of cell fractions for samples in different groups.*

Description

Visualize the estimated cell fractions for multiple groups of samples by a stacked bar chart.

Usage

```
barplotCF(mat, groupInfo = NULL, ctCol = NULL)
```

Arguments

`mat`	a numeric matrix of cell fractions for bulk samples, with sample identifiers as rownames and cell type names as colnames.
`groupInfo`	a vector or factor giving the group names for samples in "`mat`". The order of "`groupInfo`" should be same as the sample order in "`mat`" unless it was named by the sample identifiers. You can designate the bar orders of groups by setting "`groupInfo`" to factor format following your order of interest. Missing values (NA) will be removed before plotting.
`ctCol`	a character vector specifying the colors of the different cell types, which should be given in the order of the colnames of "`mat`". Default colors will be used if not provided.

Examples

```
promat <- runif(10 * 7,min = 0, max = 1)
promat <- matrix(promat, nrow = 10)
promat <- promat / rowSums(promat)
rownames(promat) <- paste0("sample_", 1:10)
colnames(promat) <- paste0("ct_", 1:7)
groupinfo <- sample(paste0('Group_', letters[1:3]), 10, replace = T)
barplotCF(promat, groupInfo = groupinfo)
```

`boxplotCF` *Boxplot of cell fractions for samples in different groups.*

Description

Visualize the estimated cell fractions for multiple groups of samples by boxplots.

Usage

```
boxplotCF(
  mat,
  groupInfo,
  groupCol = NULL,
  groupOrder = NULL,
  use.wilcoxon = T,
  use.kruskal = T
)
```

Arguments

`mat`	see the same argument of barplotCF.
`groupInfo`	see the same argument of barplotCF.
`groupCol`	a character vector specifying the colors of the different groups. Default colors will be used if not provided.
`groupOrder`	a character vector specifying the order of the group.
`use.wilcoxon`	logical, if TRUE, two-sided Wilcoxon rank-sum tests will be used to assess significance of cell fractions difference between groups, if FALSE, two-sided t tests will be used. Default is TRUE. Ignored if there are three or more levels of groupInfo.
`use.kruskal`	logical, if TRUE, Kruskal-Wallis tests will be used to assess significance of cell fractions difference between groups. If FALSE, one-way ANOVA analysis will be used. Default is TRUE. Ignored if there are only two levels of groupInfo.

Details

P values texted above the boxes are calculated by two-sided Wilcoxon rank-sum tests or two-sided t tests between two groups. If there are three or more groups, P values are generated by Kruskal-Wallis tests or one-way ANOVA analysis. P values colored in red indicate significant differences ($P < 0.05$).

Examples

```
promat <- runif(10 * 7,min = 0, max = 1)
promat <- matrix(promat, nrow = 10)
rownames(promat) <- paste0("sample_", 1:10)
colnames(promat) <- paste0("ct_", 1:7)
promat <- promat / rowSums(promat)
groupinfo <- sample(paste0('Group_', letters[1:3]), 10, replace = T)
boxplotCF(promat, groupInfo = groupinfo)
```

`buildNetwork` *Build co-expression network for individual marker based on provided training datasets.*

Description

This function builds individual co-expression network for each provided marker, to select proteins with correlated expression relationship of a marker (using samples with marker quantification) in the given training datasets (two or more proteome datasets needed).

Usage

```
buildNetwork(
  trainsets = NULL,
  markers = NULL,
  preprocess = T,
  batchInfoList = NULL,
  nTr = 2,
  corCutoff = 0.3,
  ncores = 16
)
```

Arguments

`trainsets`	a list containing the datasets used to build the network and train regression models later. Each dataset of the list should be a numeric expression matrix with HUGO gene symbols as rownames and sample identifiers as colnames. Data should be non-logarithm scale. Default is NULL, then a list including proteomic expression matrices of the six CPTAC datasets (S025, S029, S037/S045, S043, S044/S050 and S046/S056) will be used.
`markers`	a character vector of interesting markers, which are the candidate cores of the network. If NULL (default), the union of markers from public signature matrices "LM6", "LM22"and "BCIC" will be used.
`preprocess`	logical. If TURE, each matrix in "trainsets" is preprocessed before network building. Default is TRUE. See the Methods section of the NUWA manuscript for more details.
`batchInfoList`	a list containing the batch information corresponding to the training datasets. The batchInfoList should be named using the names of trainsets, or the length of batchInfoList should be equal to the number of trainsets. Each element in the list gives the batch information of one trainset, which is a vector named by the sample identifiers of that trainset.

nTr	a positive integer, we only build network for markers existing in greater than or equal to "nTr" training datasets. Default is 3.
corCutoff	a positive numeric, specifying the absolute Pearson correlation coefficient threshold above which a co-expression will be declared between individual marker and other quantified protein. For each marker, we identify its "coherently correlated proteins", i.e., proteins with the same correlation coefficient sign in all training datasets, and with significant correlation ($P < 0.05$, absolute value of Pearson correlation coefficient greater than "corCutoff") in at least two training datasets. Default is 0.3.
ncores	a positive integer, indicating the number of cores used by this function. If the operating system is windows, then only one core will be used.

Value

A list containing:

`corr` a numeric data frame of Pearson correlation coefficients between markers and other proteins.

`markers` a character vector, containing the markers with co-expression networks.

`trainScaled` a list containing the preprocessed training datasets.

Examples

```
my.net <- buildNetwork(cptacDatasets[1:3])
str(my.net)
```

NUWA.cibersort *A portal function running CIBERSORT after NUWAms analysis.*

Description

Run NUWAms (missing markers inference) and CIBERSORT algorithm (deconvolution) with a signature matrix of interest.

Usage

```
NUWA.cibersort(expr, signature_matrix, cibersortPath, ...)
```

Arguments

expr	a numeric matrix of expression profiles for bulk tissue samples, with HUGO gene symbols as rownames and sample identifiers as colnames. Data must be non-logarithm scale.
signature_matrix	a signature matrix, which is a numeric expression matrix of markers in cell types of interest, with HUGO gene symbols as rownames and cell type identifiers as colnames. Such as LM22, LM6, BCIC, TIC or user provided signature matrix.
cibersortPath	a string specifying the path of CIBERSORT R script, CIBERSORT is only freely available for academic users, please register on https://cibersort.stanford.edu, and download the CIBERSORT source script.
...	additional arguments passed to the NUWAms() function

Value

The results of each built-in NUWA analysis function, is a list containing an expression matrix with missing markers inferred, two matrices used for recall analysis, and a matrix including cell fractions estimated by the algorithm used.

Examples

```
expr <- cptacDatasets$brca[, 1:5]
res_nuwa <- NUWA.cibersort(expr, cibersortPath = cibersortPath, signature_matrix = LM22)
res_nuwa <- NUWA.cibersort(expr, cibersortPath = cibersortPath, signature_matrix = NUWAp26)
res_nuwa <- NUWA.cibersort(expr, cibersortPath = cibersortPath, signature_matrix = my_signature_matrix)
```

NUWA.EPIC *A portal function running EPIC after NUWAms analysis.*

Description

Run NUWAms and EPIC algorithm with a signature matrix of interest.

Usage

```
NUWA.EPIC(expr, signature_matrix, ...)
```

Arguments

`expr`	see the same argument in NUWA.cibersort.
`signature_matrix`	see the same argument in NUWA.cibersort.
`...`	additional arguments passed to the NUWAms() function

Value

see NUWA.cibersort.

Examples

```
expr <- cptacDatasets$brca[, 1:5]
res_nuwa <- NUWA.EPIC(expr, signature_matrix = BCIC)
res_nuwa <- NUWA.EPIC(expr, signature_matrix = TIC)
res_nuwa <- NUWA.EPIC(expr, signature_matrix = my_signature_matrix)
```

NUWA.mcpcounter *A portal function running CIBERSORT after NUWAms analysis.*

Description

Run NUWAms and MCPcounter algorithm with a marker list of interest.

Usage

```
NUWA.mcpcounter(expr, marker_list = NULL, ...)
```

Arguments

expr	see the same argument in NUWA.cibersort.
marker_list	see the same argument in NUWA.xcell, default is MCPcounter markers.
...	additional arguments passed to the NUWAms() function

Value

see NUWA.cibersort.

Examples

```
expr <- cptacDatasets$brca[, 1:5]
res_nuwa <- NUWA.mcpcounter(expr, marker_list = NULL)
res_nuwa <- NUWA.mcpcounter(expr, marker_list = my_markers)
```

NUWA.xcell *A portal function running xCell after NUWAms analysis.*

Description

Run NUWAms and xCell algorithm with a marker list of interest.

Usage

```
NUWA.xcell(expr, marker_list = NULL, ...)
```

Arguments

expr	see the same argument of NUWA.cibersort.
marker_list	a list, whose names are cellular populations' names and elements are character vectors of markers (HUGO symbols), default is xCell64.
...	additional arguments passed to the NUWAms() function

Value

see NUWA.cibersort.

Examples

```
expr <- cptacDatasets$brca[, 1:5]
res_nuwa <- NUWA.xcell(expr, marker_list = NULL)
res_nuwa <- NUWA.xcell(expr, marker_list = my_markers)
```

NUWAeDeconv *Immune cell types deconvolution using expression dataset.*

Description

This function integrates deconvolution results of three algorithm-signature combinations selected from our benchmark analysis, and provides the relative proportions of six immune cell types in mixture samples. See the NUWA manuscript for more details.

Usage

```
NUWAeDeconv(
  expr,
  cibersortPath,
  BCIC_min_marker_num = 6,
  LM6_min_marker_num = 6,
  LM22_min_marker_num = 6,
  quantileNorm_cibersort = T,
  protein = T
)
```

Arguments

`expr`	a numeric matrix of expression profiles for bulk tissue samples, with HUGO gene symbols as rownames and sample identifiers as colnames. Data must be non-logarithm scale.
`cibersortPath`	a string specifying the path of CIBERSORT R script, CIBERSORT is only freely available for academic users, please register on https://cibersort.stanford.edu, and download the CIBERSORT source script.
`BCIC_min_marker_num`	a positive integer, indicating the minimal number of BCIC markers needed to run EPIC. Default is 6.
`LM6_min_marker_num`	a positive integer, indicating the minimal number of LM6 markers needed to run CIBERSORT. Default is 6.
`LM22_min_marker_num`	a positive integer, indicating the minimal number of LM22 markers needed to run CIBERSORT. Default is 6.
`quantileNorm_cibersort`	logical, indciating whether quantile normalization will be performed in CIBERSORT analysis. Only set FALSE for RNA-seq data as recommended on the CIBERSORT website. Default is TRUE.
`protein`	logical, set TRUE for proteomic expression data. If TRUE, signature matrix including 118 markers (union of BCIC and TIC markers) will be used for EPIC analysis. If FALSE, the BCIC markers (n = 65) will be used. Default is TRUE.

Value

A list:

proportion a matrix, the first column is the cell type name, and the remaining columns (one sample per column) are the relative proportion of mRNA or protein coming from the six immune cell types (B, CD4 T, CD8 T, monocyte/macrophage, NK and neutrophils cells).

mergedProp a list containing three merged and sum-to-one normalized proportion matrices predicted by CIBERSORT-LM22, CIBERSORT-LM6 and EPIC-BCIC.

rawRes a list containing the raw proportion matrices generated by CIBERSORT-LM22, CIBERSORT-LM6 and EPIC-BCIC.

usedComb a string vector showing the combinations (from CIBERSORT-LM22, CIBERSORT-LM6 and EPIC-BCIC) used to generate the ensembled prediction of proportions. Disqualification might be caused by insufficient markers.

Examples

```
# You need to provide path to CIBERSORT.R
# cibersortPath = "/mnt/dellfs/pub/tools/ccb/NUWA/CIBERSORT.R"
my.nuwadec = NUWAeDeconv(expr=my.nuwams$finalExpr, cibersortPath= cibersortPath)
```

NUWAms *Infer proteomic expression abundance for missing immune markers, note that union of markers from public signature matrices "LM6", "LM22"and "BCIC" will be used.*

Description

This function is to infer the abundance of missing cell markers using the given co-expression networks of individual marker. It may take a few minutes, if more than 50 samples were included in the analysis.

Usage

```
NUWAms(
  expr,
  network = NULL,
  direction = c("both", "backward", "forward")[1],
  lasso_step_cutoff = 10,
  preprocess = T,
  ncores = 16,
  lambda = c("lambda.1se", "lambda.min")[1]
)
```

Arguments

expr	a numeric matrix of expression profiles for bulk tissue samples, with HUGO gene symbols as rownames and sample identifiers as colnames.
network	a list, consisting of the co-expression networks built by function buildNetwork(). Default is NULL, then the built-in networks using six CPTAC datasets (S025, S029, S037/S045, S043, S044/S050 and S046/S056) for individual marker from LM22, LM6 and BCIC signatures will be used. NULL is required when NUWAeDeconv function will be used followingly.

`direction` a character, indicating the mode used for feature searching in stepwise regression analysis, one of "both", "backward" or "forward". Default is "both".

`lasso_step_cutoff` a positive integer, specifying the minimal number of variables needed to run LASSO regression analysis. If the number is less than "lasso_step_cutoff", stepwise regression models will be constructed. Default is 10.

`preprocess` logical. If TURE, expression data is preprocessed before markers inferring. Default is TRUE. See the Methods section of the NUWA manuscript for more details.

`ncores` a positive integer, indicating the number of cores used by this function. If the operating system is windows, then only one core will be used.

`lambda` a character, indicating which value of lambda will be used in the LASSO analysis. One of "lambda.min" or "lambda.1se". "lambda.min" gives lambda with minimal cross-validation errors, and "lambda.1se" gives the largest value of lambda such that the error is within 1 standard error of the minimal. Default is "lambda.1se".

Value

A list:

`finalExpr` a numeric matrix, the final full dataset of expression with missing markers are inferred.

`predVsTruth` a list with elements comprising the prediction and truth expression matrices of quantified markers, which will be used for the following recall analysis.

`inferenceMat` a numeric matrix, a subset of the full dataset "finalExpr", a expression matrix only including markers inferred by NUWAms() function.

`runtime` a data frame, consisting running time (minutes) used for model building, markers inferring and total time.

Examples

```
expr <- cptacDatasets$brca[, 1:5]
res <- NUWAms(expr)
```

---

recall *Evaluate the inference accuracy of NUWAms by recall analysis.*

Description

This function computes recall value for each marker and sample based on the inference results of NUWAms function or NUWA built-in analysis functions. Recall of a marker was determined as the fraction of a null distribution of similarity less than the observed similarity (Spearman rank correlation) between the inferred and measured abundances of this marker in all samples of a given dataset. Recall of a sample was performed in a similar way.

Usage

```
recall(x, corMethod = c("spearman", "pearson")[1])
```

Arguments

x	a list, the output of NUWAms().
corMethod	a character, the type of correlation coeficient to be used. One of "pearson" or "spearman" (default).

Value

A list:

recallTable a data frame recording the recall values of each marker and each sample.

simNull a list of length 2, consisting two correlation matrices which were used to generate null distributions of similarity (SIMnull) at marker and sample level, respectively. See the Methods section of the NUWA manuscript for more detail.

corMethod a character, the type of correlation method used.

Examples

```
res <- recall(myInfer)
```

---

recall.plot *Plotting the recall values at both marker level and sample level.*

Description

This function outputs a density plot and a scatter points plot to visualize NUWA-ms accuracy by evaluating similarity between observed and inferred abundances for markers with quantification in the inferred dataset.

Usage

```
recall.plot(recallRes, level = c("marker", "sample")[1], ...)
```

Arguments

recallRes	a list, the output of recall function.
level	a character, one of "marker" (at marker level) and "sample" (at marker level). Default is "marker". At marker level, scatter plot showing associations between marker level recall and correlation coefficients for all samples. Dotted lines indicate a recall of 0.8, i.e. 80th percentile. At sample level, density plot showing distributions of correlation coefficients within the same samples or between different samples. Accuracy rate (AR), representing the overall accuracy in the dataset, and the number of comparison are indicated.
...	additional arguments passed to the ggplot2::theme() function.

Value

ggplot object

Examples

```
recall.plot(recallRes, "marker")
```

Index