Package 'MOVICS'

September 18, 2020

Title Multi-Omics integration and VIsualization in Cancer Subtyping

Version 0.99.1

Description This package provides a unified interface for 10 state-of-the-art multi-omics clustering algorithms, and forms a pipeline for most commonly used downstream analyses in cancer subtyping researches and to create feature rich customizable visualzations with minimal effort.

Depends R (>= 4.0.2)

License GPL-3 + file LICENSE

Encoding UTF-8

LazyData true

Suggests testthat,
knitr,
rmarkdown,
kableExtra

Remotes
github::danro9685/CIMLR@R, github::Lothelab/CMScaller@master, github::jokergoo/ComplexHeatmap@master

Imports CIMLR,
ClassDiscovery,
ConsensusClusterPlus,
IntNMF,
PINSPlus,
SNFtool,
coca,
dplyr,
ggplot2,
iClusterPlus,
mogsa,
vegan,
circlize,
survival,
survminer,
ggpmisc,
tibble,
limma,
DESeq2,
edgeR,
officer,
aricode,

ggalluvial,
flexclust,
reshape2,
clusterProfiler,
GSVA,
grid,
cowplot,
jstable,
impute,
CMScaller,
car,
genefilter,
ggpubr,
preprocessCore,
ridge,
sva,
grDevices,
maftools,
patchwork,
ComplexHeatmap (>= 2.5.4),
pamr,
clusterRepro

VignetteBuilder knitr

RoxygenNote 7.1.1

biocViews Clustering, DifferentialExpression, GeneSetEnrichment, MultipleComparison, SomaticMutation, CopyNumberVariation, Survival

URL https://github.com/xlucpu/MOVICS

BugReports https://github.com/xlucpu/MOVICS/issues

Collate 'compAgree.R'
'compClinvar.R'
'compDrugsen.R'
'compFGA.R'
'compMut.R'
'compSurv.R'
'compTMB.R'
'getCIMLR.R'
'getCOCA.R'
'getClustNum.R'
'getConsensusClustering.R'
'getConsensusMOIC.R'
'getElites.R'
'getIntNMF.R'
'getLRAcluster.R'
'getMOIC.R'
'getMoCluster.R'
'getMoHeatmap.R'
'getNEMO.R'
'getPINSPlus.R'
'getSNF.R'
'getStdiz.R'

'getiClusterBayes.R'
'runDEA.R'
'runGSEA.R'
'runMarker.R'
'runNTP.R'
'runPAM.R'
'runKappa.R'
'pRRophetic.R'
'LRAcluster.R'
'quiet.R'
'dataset.R'
'global.R'

R topics documented:

cgp2016ExprRma *GDSC 2016 gene expression data*

Description

A matrix frame storing the GDSC 2016 gene expression data.

Usage

```
cgp2016ExprRma
```

Format

A matrix with 17419 rows and 1018 variables.

contains "cgp2016ExprRma" the 2016 gene expression data. Data was obtained from http://www.cancerrxgene.org/gdsc1000/GDSC1000_WebResources/Data/preprocessed/Cell_line_RMA_proc_basalExp.txt.zip. Cosmic Ids (in the column names) were mapped to cell line names using data from this file: ftp://ftp.sanger.ac.uk/pub/project/cancerrxgene/releases/release-6.0/Cell_Lines_Details.xlsx

Source

https://osf.io/5xvsg/

compAgree *Comparison of agreement between two subtypes*

Description

Compute the Rand Index, Jaccard Index, Fowlkes-Mallows, and Normalized Mutual Information for agreement of two partitions, and generate alluvial diagrams for visualization.

Usage

```
compAgree(
  moic.res = NULL,
  subt2comp = NULL,
  doPlot = TRUE,
  clust.col = c("#2EC4B6", "#E71D36", "#FF9F1C", "#BDD5EA", "#FFA5AB", "#011627",
    "#023E8A", "9D4EDD"),
  box.width = 0.1,
  fig.name = NULL,
  fig.path = getwd(),
  width = 6,
  height = 5
)
```

Arguments

moic.res	An object returned by 'getMOIC()' with one specified algorithm or 'get%algorithm_name%' or 'getConsensusMOIC()' with a list of multiple algorithms.
subt2comp	A data.frame of subtypes that need to compare with current subtype with row-names for samples and columns for other subtypes.
doPlot	A logic value to indicate if generating alluvial diagram to show the agreement of different subtypes.
clust.col	A string vector storing colors for each cluster.
box.width	A numeric valur to indicate the width for box in alluvial diagram.
fig.name	A string value to indicate the name of the figure
fig.path	A string value to indicate the output path for storing the figure
width	A numeric value to indicate the width of alluvial diagram.
height	A numeric value to indicate the height of alluvial diagram.

Value

A figure of agreement (.pdf) if doPlot = TRUE and a data.frame storing four agreement measurements, including Rand Index (RI), Adjusted Mutual Information (AMI), Jaccard Index (JI), and Fowlkes-Mallows (FM).

Examples

```
# There is no example and please refer to vignette.
```

---

compClinvar *Comparison of clinical variables*

Description

Create a table summarizing all baseline variables (both continuous and categorical) stratifying by current identified Subtypes and performing statistical tests. The object gives a table that is easy to use in medical research papers.

Usage

```
compClinvar(
  moic.res = NULL,
  var2comp = NULL,
  strata = "Subtype",
  factorVars = NULL,
  nonnormalVars = NULL,
  exactVars = NULL,
  includeNA = FALSE,
  doWord = TRUE,
  tab.name = NULL,
  res.path = getwd(),
  ...
)
```

Arguments

moic.res	An object returned by 'getMOIC()' with one specified algorithm or 'get%algorithm_name%' or 'getConsensusMOIC()' with a list of multiple algorithms.
var2comp	A data.frame of clinical variables that need to compare among current subtypes with rownames for samples and columns for variable names.
strata	A string value to indicate the stratifying (subtype) variable; 'Subtype' by default.
factorVars	A string vectors to indicate the categorical variables. If omitted, only factors are considered categorical variables.
nonnormalVars	A string vector to specify the variables for which the p-values should be those of nonparametric tests. By default all p-values are from normal assumption-based tests (oneway.test)., Default: NULL
exactVars	A string vector to specify the variables for which the p-values should be those of exact tests. By default all p-values are from large sample approximation tests (chisq.test)., Default: NULL
includeNA	A logic value to indicate if NA should be handled as a regular factor level rather than missing. NA is shown as the last factor level in the table if includeNA = T. Only effective for categorical variables., Default: FALSE
doWord	A logic value to indicate if transformating the .txt outfile to a .docx WORD file (.txt file will be also kept).
tab.name	A string value to indicate the name of the output table.
res.path	A string value to indicate the path for saving the table.
...	Additionnal parameters pass to jstable::CreateTableOne2

Value

A summarizing table with stastitical testing results.

Examples

```
# There is no example and please refer to vignette.
```

---

compDrugsen *Comparison of drug sensitivity*

Description

This function estimates the IC50 of specific drugs for each Subtype by developing a ridge regression predictive model based on all/specific cell lines derived from Genomics of Drug Sensitivity in Cancer (GDSC, https://www.cancerrxgene.org/).

Usage

```
compDrugsen(
  moic.res = NULL,
  norm.expr = NULL,
  drugs = c("Cisplatin", "Paclitaxel"),
  tissueType = "all",
  test.method = "nonparametric",
```

```
  clust.col = c("#2EC4B6", "#E71D36", "#FF9F1C", "#BDD5EA", "#FFA5AB", "#011627",
    "#023E8A", "9D4EDD"),
  prefix = NULL,
  seed = 123456,
  fig.path = getwd(),
  width = 5,
  height = 5
)
```

Arguments

`moic.res`	An object returned by 'getMOIC()' with one specified algorithm or 'get%algorithm_name%' or 'getConsensusMOIC()' with a list of multiple algorithms.
`norm.expr`	A matrix of normalized expression data with rows for genes and columns for samples; FPKM or TPM without log2 transformation is recommended.
`drugs`	A string vector to indicate the names of the drugs for which you would like to predict sensitivity, one of A.443654, A.770041, ABT.263, ABT.888, AG.014699, AICAR, AKT.inhibitor.VIII, AMG.706, AP.24534, AS601245, ATRA, AUY922, Axitinib, AZ628, AZD.0530, AZD.2281, AZD6244, AZD6482, AZD7762, AZD8055, BAY.61.3606, Bexarotene, BI.2536, BIBW2992, Bicalutamide, BI.D1870, BIR-B.0796, Bleomycin, BMS.509744, BMS.536924, BMS.708163, BMS.754807, Bortezomib, Bosutinib, Bryostatin.1, BX.795, Camptothecin, CCT007093, CC-T018159, CEP.701, CGP.082996, CGP.60474, CHIR.99021, CI.1040, Cisplatin, CMK, Cyclopamine, Cytarabine, Dasatinib, DMOG, Docetaxel, Doxorubicin, EHT.1864, Elesclomol, Embelin, Epothilone.B, Erlotinib, Etoposide, FH535, FTI.277, GDC.0449, GDC0941, Gefitinib, Gemcitabine, GNF.2, GSK269962A, GSK.650394, GW.441756, GW843682X, Imatinib, IPA.3, JNJ.26854165, JNK.9L, JNK.Inhibitor.VIII, JW.7.52.1, KIN001.135, KU.55933, Lapatinib, Lenalido-mide, LFM.A13, Metformin, Methotrexate, MG.132, Midostaurin, Mitomycin.C, MK.2206, MS.275, Nilotinib, NSC.87877, NU.7441, Nutlin.3a, NVP.BEZ235, NVP.TAE684, Obatoclax.Mesylate, OSI.906, PAC.1, Paclitaxel, Parthenolide, Pazopanib, PD.0325901, PD.0332991, PD.173074, PF.02341066, PF.4708671, PF.562271, PHA.665752, PLX4720, Pyrimethamine, QS11, Rapamycin, RDEA119, RO.3306, Roscovitine, Salubrinal, SB.216763, SB590885, Shikonin, SL.0101.1, Sorafenib, S.Trityl.L.cysteine, Sunitinib, Temsirolimus, Thapsigargin, Tipifarnib, TW.37, Vinblastine, Vinorelbine, Vorinostat, VX.680, VX.702, WH.4.023, WO2009093972, WZ.1.84, X17.AAG, X681640, XMD8.85, Z.LLNle.CHO, ZM.447439; two common chemodrugs (i.e., Cisplatin and Paclitaxel) will be analyzed by default if no indication.
`tissueType`	A string value to specify if you would like to traing the models on only a subset of the CGP cell lines (based on the tissue type from which the cell lines originated); Allowed values contain c("all", "aero_digestive_tract", "blood", "bone", "breast", "digestive_system", "lung", "nervous_system", "skin", "urogenital_system") and "all" by default.
`test.method`	A string value to indicate the method for statistical testing. Allowed values contain c('nonparametric', 'parametric'); nonparametric means two-sample wilcoxon rank sum test for two subtypes and Kruskal-Wallis rank sum test for multiple subtypes; parametric means two-sample t-test when only two subtypes are identified, and anova for multiple subtypes comparison.
`clust.col`	A string vector storing colors for annotating each Subtype.
`prefix`	A string value to indicate the prefix of the output plot.

seed	A integer value to indicate the seed for reproducing ridge regression.
fig.path	A string value to indicate the output path for storing the boxviolin plot.
width	A numeric value to indicate the width of boxviolin plot.
height	A numeric value to indicate the height of boxviolin plot.

Value

Data.frame(s) storing the estimated IC50 of specified drugs per sample within each Subtype.

References

Geeleher P, Cox N, Huang R S. (2014). pRRophetic: an R package for prediction of clinical chemotherapeutic response from tumor gene expression levels. PLoS One, 9(9):e107468.

Geeleher P, Cox N J, Huang R S. (2014). Clinical drug response can be predicted using baseline gene expression levels and in vitro drug sensitivity in cell lines. Genome Biol, 15(3):1-12.

Examples

```
# There is no example and please refer to vignette.
```

---

compFGA *Comparison of fraction genome altered*

Description

This function calculates Fraction Genome Altered (FGA), Fraction Genome Gained (FGG), and Fraction Genome Lost (FGL) seperately, and compares them among curent subtypes identified from multi-omics integrative clustering algorithms.

Usage

```
compFGA(
  moic.res = NULL,
  segment = NULL,
  iscopynumber = FALSE,
  cnathreshold = 0.2,
  test.method = "nonparametric",
  barcolor = c("#008B8A", "#F2042C", "#21498D"),
  clust.col = c("#2EC4B6", "#E71D36", "#FF9F1C", "#BDD5EA", "#FFA5AB", "#011627",
    "#023E8A", "9D4EDD"),
  fig.path = getwd(),
  fig.name = NULL,
  width = 8,
  height = 4
)
```

Arguments

moic.res	An object returned by 'getMOIC()' with one specified algorithm or 'get%algorithm_name%' or 'getConsensusMOIC()' with a list of multiple algorithms.
segment	A data frame containing segmented copy number and columns must exactly include the following elements: c('sample','chrom','start','end','value'). Column of 'value' should be segments value when iscopynumber = FALSE but copy-number value when iscopynumber = TRUE. Copy-number will be converted to segments by log2(copy-number/2).
iscopynumber	A logical value to indicate if the fifth column of segment input is copy-number. If segment file derived from CNV calling provides copy number instead of segment_mean value, this argument must be switched to TRUE. FALSE by default.
cnathreshold	A numeric value to indicate the cutoff for identifying copy-number gain or loss. 0.2 by default.
test.method	A string value to indicate the method for statistical testing. Allowed values contain c('nonparametric', 'parametric'); nonparametric means two-sample wilcoxon rank sum test for two subtypes and Kruskal-Wallis rank sum test for multiple subtypes; parametric means two-sample t-test when only two subtypes are identified, and anova for multiple subtypes comparison.
barcolor	A string vector to indicate the mapping color for bars of FGA, FGG and FGL.
clust.col	A string vector storing colors for each subtype.
fig.path	A string value to indicate the output path for storing the barplot.
fig.name	A string value to indicate the name of the barplot.
width	A numeric value to indicate the width of barplot.
height	A numeric value to indicate the height of barplot.

Value

A list contains the following components:

summary a table summarizing the measurements of FGA, FGG, and FGL per sample

FGA.p.value a nominal p value quantifying the difference of FGA among current subtypes

FGG.p.value a nominal p value quantifying the difference of FGG among current subtypes

FGL.p.value a nominal p value quantifying the difference of FGL among current subtypes

test.method a string value indicating the statistical testing method to calculate p values

References

Cerami E, Gao J, Dogrusoz U, et al. (2012). The cBio Cancer Genomics Portal: An Open Platform for Exploring Multidimensional Cancer Genomics Data. Cancer Discov, 2(5):401-404.

Gao J, Aksoy B A, Dogrusoz U, et al. (2013). Integrative analysis of complex cancer genomics and clinical profiles using the cBioPortal. Sci Signal, 6(269):pl1-pl1.

Examples

```
# There is no example and please refer to vignette.
```

compMut *Comparison of mutational frequency*

Description

This function is used to compare mutational frequency among different multi-omics integerative clusters to test the independency between subtypes and mutational status. An oncoprint will be also generated with significant mutations.

Usage

```
compMut(
  moic.res = NULL,
  mut.matrix = NULL,
  freq.cutoff = 0.05,
  test.method = "fisher",
  p.adj.method = "BH",
  doWord = TRUE,
  doPlot = TRUE,
  innerclust = TRUE,
  res.path = getwd(),
  tab.name = NULL,
  fig.path = getwd(),
  fig.name = NULL,
  annCol = NULL,
  annColors = NULL,
  mut.col = "#21498D",
  bg.col = "#dcddde",
  p.cutoff = 0.05,
  p.adj.cutoff = 0.05,
  clust.col = c("#2EC4B6", "#E71D36", "#FF9F1C", "#BDD5EA", "#FFA5AB", "#011627",
    "#023E8A", "9D4EDD"),
  width = 8,
  height = 4
)
```

Arguments

moic.res	An object returned by 'getMOIC()' with one specified algorithm or 'get%algorithm_name%' or 'getConsensusMOIC()' with a list of multiple algorithms.
mut.matrix	A binary matrix storing binary mutation data with entries of 0 and 1 only.
freq.cutoff	A numeric value to indicate the frequency cutoff for mutation data. Specifically, only features that mutated in over than such proportion would be included in testing; 0.05 by default.
test.method	A string value to indicate statistical method for independency testing. Allowed values contain c('fisher', 'chisq'); fisher by default.
p.adj.method	A string value to indicate the correction method for multiple comparision. Allowed values contain c('holm', 'hochberg', 'hommel', 'bonferroni', 'BH', 'BY', 'fdr'); BH by default.

doWord	A logic value to indicate if transformating the .txt outfile to a .docx WORD file (.txt file will be also kept).
doPlot	A logic value to indicate if generating oncoprint.
innerclust	A logic value to indicate if perform clustering within each subtype; TRUE by default.
res.path	A string value to indicate the path for saving the table.
tab.name	A string value to indicate the name of the output table.
fig.path	A string value to indicate the output path for storing the oncoprint.
fig.name	A string value to indicate the name of the oncoprint.
annCol	A data.frame storing annotation information for samples.
annColors	A list of string vectors for colors matched with annCol.
mut.col	A string vector to indicate the mutation color for oncoprint.
bg.col	A string vector to indicate the background color for oncoprint.
p.cutoff	A numeric value to indicate the nominal p value cutoff for significant mutations shown in the oncoprint; 0.05 by default.
p.adj.cutoff	A numeric value to indicate the adjusted p value cutoff for significant mutations shown in the oncoprint; 0.05 by default.
clust.col	A string vector storing colors for annotating each subtype.
width	A numeric value to indicate the width of output figure.
height	A numeric value to indicate the height of output figure.

Value

A figure of mutational oncoprint (.pdf) if doPlot = TRUE, a data.frame storing the difference of mutational frequency among different subtypes and a corresponding table in WORD format if doWord = TRUE.

References

Gu Z, Eils R, Schlesner M (2016). Complex heatmaps reveal patterns and correlations in multidimensional genomic data. Bioinformatics, 32(18):2847–2849.

Examples

```
# There is no example and please refer to vignette.
```

---

compSurv *Comparison of prognosis by Kaplan-Meier survival curve*

Description

This function calculates Kaplan-meier estimator and generate survival curves with log-rank test to detect prognostic difference among identified subtypes. If more than 2 subtypes are identified, pair-wise comparisons will be performed with an additional table printed on the survival curve.

Usage

```
compSurv(
  moic.res = NULL,
  surv.info = NULL,
  convt.time = "d",
  surv.cut = NULL,
  clust.col = c("#2EC4B6", "#E71D36", "#FF9F1C", "#BDD5EA", "#FFA5AB", "#011627",
    "#023E8A", "9D4EDD"),
  p.adjust.method = "BH",
  surv.median.line = "none",
  fig.name = NULL,
  fig.path = getwd()
)
```

Arguments

moic.res	An object returned by 'getMOIC()' with one specified algorithm or 'get%algorithm_name%' or 'getConsensusMOIC()' with a list of multiple algorithms.
surv.info	A data.frame with rownames of observations and with at least two columns of 'futime' for survival time and 'fustat' for survival status (0: censoring; 1: event)
convt.time	A string value to indicate how to convert the survival time; value of 'd' for days, 'm' for months and 'y' for years.
surv.cut	A numeric value to indicate the x-axis cutoff for showing the maximal survival time.
clust.col	A string vector storing colors for each subtype.
p.adjust.method	A string value for indicating method for adjusting p values (see p.adjust). Allowed values include one of c('holm', 'hochberg', 'hommel', 'bonferroni', 'BH', 'BY', 'fdr', 'none').
surv.median.line	A string value for drawing a horizontal/vertical line at median survival. Allowed values include one of c('none', 'hv', 'h', 'v'). v: vertical, h:horizontal.
fig.name	A string value to indicate the output path for storing the kaplan-meier curve.
fig.path	A string value to indicate the name of the kaplan-meier curve.

Value

A figure of multi-omics Kaplan-Meier curve (.pdf) and a list with the following components:

fitd an object returned by survdiff.

fid an object returned by survfit.

overall.p a nominal p.value calculated by Kaplain-Meier estimator with log-rank test.

pairwise.p an object of class "pairwise.htest" which is a list containing the p values (see pairwise_survdiff); (*only returned when more than 2 subtypes are identified*).

Examples

```
# There is no example and please refer to vignette.
```

compTMB *Comparsion of total mutation burden*

Description

This function calculates Total Mutation Burden (TMB) compares them among curent subtypes identified from multi-omics integrative clustering algorithms.

Usage

```
compTMB(
  moic.res = NULL,
  maf = NULL,
  rmDup = TRUE,
  rmFLAGS = FALSE,
  nonSyn = NULL,
  exome.size = 38,
  clust.col = c("#2EC4B6", "#E71D36", "#FF9F1C", "#BDD5EA", "#FFA5AB", "#011627",
    "#023E8A", "9D4EDD"),
  test.method = "nonparametric",
  show.size = TRUE,
  fig.path = getwd(),
  fig.name = NULL,
  width = 6,
  height = 6
)
```

Arguments

moic.res	An object returned by 'getMOIC()' with one specified algorithm or 'get%algorithm_name%' or 'getConsensusMOIC()' with a list of multiple algorithms.
maf	A data frame of MAF file that has been already read with at least 10 columns as following: c('Tumor_Sample_Barcode', 'Hugo_Symbol', 'Chromosome', 'Start_Position', 'End_Position', 'Variant_Classification', 'Variant_Type', 'Reference_Allele', 'Tumor_Seq_Allele1', 'Tumor_Seq_Allele2')
rmDup	A logical value to indicate if removing repeated variants in a particuar sample, mapped to multiple transcripts of same Gene. TRUE by default.
rmFLAGS	A logical value to indicate if removing possible FLAGS. These FLAGS genes are often non-pathogenic and passengers, but are frequently mutated in most of the public exome studies, some of which are fishy. Examples of such genes include TTN, MUC16, etc. FALSE by default.
nonSyn	A string vector to indicate a list of variant claccifications that should be considered as non-synonymous and the rest will be considered synonymous (silent) variants. Default value of NULL uses Variant Classifications with High/Moderate variant consequences, including c('Frame_Shift_Del', 'Frame_Shift_Ins', 'Splice_Site', 'Translation_Start_Site', 'Nonsense_Mutation', 'Nonstop_Mutation', 'In_Frame_Del', 'In_Frame_Ins', 'Missense_Mutation'). See details at http://asia.ensembl.org/Help/Glossary?id=535
exome.size	An integer value to indicate the estimation of exome size. 38 by default (see https://genomemedicine.biomedcentral.com/articles/10.1186/s13073-017-0424-2).

clust.col	A string vector storing colors for annotating each Subtype.
test.method	A string value to indicate the method for statistical testing. Allowed values contain c('nonparametric', 'parametric'); nonparametric means two-sample wilcoxon rank sum test for two subtypes and Kruskal-Wallis rank sum test for multiple subtypes; parametric means two-sample t-test when only two subtypes are identified, and anova for multiple subtypes comparison.
show.size	A logical value to indicate if showing the sample size within each subtype at the top of the figure. TRUE by default.
fig.path	A string value to indicate the output path for storing the boxviolin plot.
fig.name	A string value to indicate the name of the boxviolin plot.
width	A numeric value to indicate the width of boxviolin plot.
height	A numeric value to indicate the height of boxviolin plot.

Value

A figure of TMB and TiTv distribution (.pdf) and a list with the following components:

TMB.dat a data.frame storing the TMB per sample within each subtype.

TMB.median a data.frame storing the median of TMB for each subtype.

titv.dat a data.frame storing the fraction contributions of TiTv per sample within each subtype.

maf.nonsilent a data.frame storing the information for non-synonymous mutations.

maf.silent a data.frame storing the information for synonymous mutations.

maf.FLAGS a data.frame storing the information for FLAGS mutations ifrmFLAGS = TRUE.

FLAGS.count a data.frame storing the summarization per FLAGS ifrmFLAGS = TRUE.

References

Mayakonda A, Lin D, Assenov Y, Plass C, Koeffler PH (2018). Maftools: efficient and comprehensive analysis of somatic variants in cancer. Genome Res, 28(11): 1747-1756.

Shyr C, Tarailo-Graovac M, Gottlieb M, Lee JJ, van Karnebeek C, Wasserman WW. (2014). FLAGS, frequently mutated genes in public exomes. BMC Med Genomics, 7(1): 1-14.

Chalmers Z R, Connelly C F, Fabrizio D, et al. (2017). Analysis of 100,000 human cancer genomes reveals the landscape of tumor mutational burden. Genome Med, 9(1):34.

Examples

```
# There is no example and please refer to vignette.
```

---

drugData2016 *GDSC 2016 drug data*

Description

A data frame storing the GDSC 2016 drug data.

Usage

```
drugData2016
```

Format

A data frame with 224510 rows and 13 variables.

contains "drugData2016" the 2016 drug IC50 data, downloaded from http://www.cancerrxgene.org/translation/drug/download#ic50)

Source

https://osf.io/5xvsg/

`getCIMLR` *Get subtypes from CIMLR*

Description

This function wraps the CIMLR (Cancer Integration via Multikernel Learning) algorithm and provides standard output for 'getMoHeatmap()' and 'getConsensusMOIC()'.

Usage

```
getCIMLR(
  data = NULL,
  N.clust = NULL,
  type = rep("gaussian", length(data)),
  cores.ratio = 0,
  verbose = TRUE
)
```

Arguments

`data`	List of matrices.
`N.clust`	Number of clusters.
`type`	Data type corresponding to the list of matrics, which can be gaussian, binomial or possion.
`cores.ratio`	Ratio of the number of cores to be used when computing the multi-kernel.
`verbose`	A logic value to indicate if supressing progression.

Value

A list with the following components:

`fit` an object returned by CIMLR.

`clust.res` a data.frame storing sample ID and corresponding clusters.

`feat.res` the results of features selection process.

`mo.method` a string value indicating the method used for multi-omics integrative clustering.

References

Ramazzotti D, Lal A, Wang B, Batzoglou S, Sidow A (2018). Multi-omic tumor data reveal diversity of molecular mechanisms that correlate with survival. Nat Commun, 9(1):4453.

Examples

```
# There is no example and please refer to vignette.
```

`getClustNum` *Get estimation of optimal clustering number*

Description

This function provides two measurements (i.e., clustering prediction index [CPI] and Gap-statistics) and aims to search the optimal number for multi-omics integrative clustering. In short, the peaks reach by the red (CPI) and blue (Gap-statistics) lines should be referred to determine 'N.clust'.

Usage

```
getClustNum(
  data = NULL,
  is.binary = rep(FALSE, length(data)),
  try.N.clust = 2:8,
  center = TRUE,
  scale = TRUE,
  fig.path = getwd(),
  fig.name = "optimal_number_cluster"
)
```

Arguments

`data`	List of matrices.
`is.binary`	A logicial vector to indicate if the subdata is binary matrix of 0 and 1 such as mutation.
`try.N.clust`	A integer vector to indicate possible choices of number of clusters.
`center`	A logical value to indicate if the variables should be centered. TRUE by default.
`scale`	A logical value to indicate if the variables should be scaled. FALSE by default.
`fig.path`	A string value to indicate the output figure path.
`fig.name`	A string value to indicate the name of the figure.

Value

A figure that helps to choose the optimal clustering number (argument of 'N.clust') for 'get

`CPI` possible cluster number identified by clustering prediction index

`Gapk` possible cluster number identified by Gap-statistics

References

Chalise P, Fridley BL (2017). Integrative clustering of multi-level omic data based on non-negative matrix factorization algorithm. PLoS One, 12(5):e0176278.

Tibshirani, R., Walther, G., Hastie, T. (2001). Estimating the number of data clusters via the Gap statistic. J R Stat Soc Series B Stat Methodol, 63(2):411-423.

Examples

```
# There is no example and please refer to vignette.
```

`getCOCA` *Get subtypes from COCA*

Description

This function wraps the COCA (Cluster-of-Clusters Analysis) algorithm and provides standard output for 'getMoHeatmap()' and 'getConsensusMOIC()'.

Usage

```
getCOCA(
  data = NULL,
  N.clust = NULL,
  type = rep("gaussian", length(data)),
  methods = "hclust",
  distances = "euclidean"
)
```

Arguments

`data`	List of matrices.
`N.clust`	Number of clusters.
`type`	Data type corresponding to the list of matrics, which can be gaussian, binomial or possion.
`methods`	A string vector storing the names of clustering methods to be used to cluster the observations in each subdataset.
`distances`	A string vector storing the name of distances to be used in the clustering step for each subdataset.

Value

A list with the following components:

`fit` an object returned by coca.

`clust.res` a data.frame storing sample ID and corresponding clusters.

`clust.dend` a dendrogram of sample clustering.

`mo.method` a string value indicating the method used for multi-omics integrative clustering.

References

Hoadley KA, Yau C, Wolf DM, et al (2014). Multiplatform analysis of 12 cancer types reveals molecular classification within and across tissues of origin. Cell, 158(4):929-944.

Examples

```
# There is no example and please refer to vignette.
```

`getConsensusClustering` *Get subtypes from ConsensusClustering*

Description

This function wraps the Consensus Clustering algorithm and provides standard output for 'getMoHeatmap()' and 'getConsensusMOIC()'.

Usage

```
getConsensusClustering(
  data = NULL,
  N.clust = NULL,
  type = rep("gaussian", length(data)),
  norMethod = "none",
  reps = 500,
  pItem = 0.8,
  pFeature = 0.8,
  clusterAlg = "hc",
  innerLinkage = "ward.D",
  finalLinkage = "ward.D",
  distance = "pearson",
  plot = NULL,
  writeTable = F,
  title = file.path(getwd(), "consensuscluster"),
  seed = 123456,
  verbose = F
)
```

Arguments

`data`	List of matrices.
`N.clust`	Number of clusters.
`type`	Data type corresponding to the list of matrics, which can be gaussian, binomial or possion.
`norMethod`	A string vector indicate the normalization method for consensus clustering.
`reps`	An integer value to indicate the number of subsamples.
`pItem`	A numerical value to indicate the proportion of items to sample.
`pFeature`	A numerical value to indicate the proportion of features to sample.
`clusterAlg`	A string value to indicate the cluster algorithm.
`innerLinkage`	A string value to indicate the heirachical linakge method for subsampling.
`finalLinkage`	A string value to indicate the heirarchical method for consensus matrix.
`distance`	A string value to indicate the distance function.
`plot`	A string value to indicate the output format for heatmap.
`writeTable`	A logical value to indicate if writing output and log to csv.
`title`	A string value for output directory.
`seed`	A numerical value to set random seed for reproducible results.
`verbose`	A logical value to indicate if printing messages to the screen to indicate progress.

Value

A list with the following components:

`fit` an object returned by ConsensusClusterPlus.

`clust.res` a data.frame storing sample ID and corresponding clusters.

`clust.dend` a dendrogram of sample clustering.

`mo.method` a string value indicating the method used for multi-omics integrative clustering.

References

Monti S, Tamayo P, Mesirov J, et al (2003). Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data. Mach Learn, 52:91-118.

Examples

```
# There is no example and please refer to vignette.
```

`getConsensusMOIC` *Get subtypes from consensus clustering of multiple multi-omics integrative clustering algorithms*

Description

Since this R package integrates 10 mainstream multi-omics clustering algorithms, we borrow the idea of consensus clustering for later integration of the clustering results derived from different algorithms, so as to improve the clustering robustness. The simplest way to run 'getConsensusMOIC()' is to pass a list of object returned by 'get%algorithm_name%()' or by 'getMOIC()' with specific argument of 'methodslist'.

Usage

```
getConsensusMOIC(
  moic.res.list = NULL,
  distance = "euclidean",
  linkage = "ward.D",
  mapcolor = c("#000004FF", "#56106EFF", "#BB3754FF", "#F98C0AFF", "#FCFFA4FF"),
  clust.col = c("#2EC4B6", "#E71D36", "#FF9F1C", "#BDD5EA", "#FFA5AB", "#011627",
    "#023E8A", "9D4EDD"),
  showID = FALSE,
  fig.path = getwd(),
  fig.name = "consensusheatmap",
  width = 5.5,
  height = 5
)
```

Arguments

moic.res.list	A list of object returned by 'getMOIC()'.
distance	A string value of distance measurement for hierarchical clustering; 'euclidean' by default.
linkage	A string value of clustering method for hierarchical clustering; 'ward.D' by default.
mapcolor	A string vector for heatmap mapping color.
clust.col	A string vector storing colors for annotating each cluster at the top of heatmap.
showID	A logic value to indicate if showing the sample ID.
fig.path	A string value to indicate the output path for storing the consensus heatmap.
fig.name	A string value to indicate the name of the consensus heatmap.
width	A numeric value to indicate the width of output figure.
height	A numeric value to indicate the height of output figure.

Value

A list contains the following components:

consensus.hm an object returned by pheatmap

similarity a similary matrix for pair-wise samples with entries ranging from 0 to 1

clust.res a data.frame storing sample ID and corresponding clusters

clust.dend a dendrogram of sample clustering

mo.method a string value indicating the method used for multi-omics integrative clustering

References

Gu Z, Eils R, Schlesner M (2016). Complex heatmaps reveal patterns and correlations in multidimensional genomic data. Bioinformatics, 32(18):2847-2849.

Examples

```
# There is no example and please refer to vignette.
```

---

getElites *Get elites for clustering*

Description

This function provides several methods to help selecting elites from input features, which aims to reduce data dimention for multi-omics integrative clustering analysis.

Usage

```
getElites(
  dat = NULL,
  surv.info = NULL,
  method = "mad",
  na.action = "rm",
  doLog2 = FALSE,
  p.cutoff = 0.05,
  elite.pct = NULL,
  elite.num = NULL,
  scaleFlag = FALSE,
  centerFlag = FALSE
)
```

Arguments

dat	A data.frame of one omics data, can be continous or binary data.
surv.info	A data.frame with rownames of observations and with at least two columns of 'futime' for survival time and 'fustat' for survival status (0: censoring; 1: event)
method	A string value to indicate the filtering method for selecting elites. Allowed values contain c('mad', 'sd', 'cox', 'freq'). 'mad' means median absolute deviation, 'sd' means standard deviation, 'cox' means univariate Cox proportional hazards regression which needs surv.info also, 'freq' only works for binary data.
na.action	A string value to indicate the action for handling NA missing value. Allowed values contain c('rm', 'impute'). 'rm' means removal of all features containing any missing values, 'impute' means imputation for missing values by k-nearest neighbors
doLog2	A logic value to indicate if performing log2 transformation for data before calculating statistics (e.g., sd, mad and cox). FALSE by default.
p.cutoff	A numeric cutoff for nominal p value derived from univariate Cox proportional hazards regression; 0.05 by default.
elite.pct	A numberic cutoff of percentage for selecting elites. NOTE: epite.pct works for all methods except for 'cox', but two scenarios exist. 1) when using method of 'mad' or 'sd', features will be descending sorted by mad or sd, and top elites.pct * feature size of elites (features) will be selected; 2) when using method of 'freq' for binary data, frequency for value of 1 will be calculated for each feature, and features that have value of 1 in greater than elites.pct * sample size will be considered elites. This argument will be discarded if elite.num is provided simultaneously. Set this argument with 1 and leave elite.num NULL will return all the features as elites after dealing with NA values.
elite.num	A integer cutoff of exact number for selecting elites. NOTE: elite.num works for all methods except for 'cox', but two scenarios exist. 1) when using method of 'mad' or 'sd', features will be descending sorted by mad or sd, and top elite.num of elites (features) will be selected; 2) when using method of 'freq' for binary data, frequency for value of 1 will be calculated for each feature, and features that have value of 1 in greater than elite.num of sample size will be considered elites.
scaleFlag	A logic value to indicate if scaling the data after filtering. FALSE by default.
centerFlag	A logic value to indicate if centerring the data after filtering. FALSE by default.

Value

A list containing the following components:

`elite.dat` a data.frame containing data for selected elites (features).

`unicox.res` a data.frame containing results for univariate Cox proportional hazards regression if `method == cox`

Examples

```
# There is no example and please refer to vignette.
```

`getiClusterBayes` *Get subtypes from iClusterBayes*

Description

This function wraps the iClusterBayes (Integrative clustering by Bayesian latent variable model) algorithm and provides standard output for 'getMoHeatmap()' and 'getConsensusMOIC()'.

Usage

```
getiClusterBayes(
  data = NULL,
  N.clust = NULL,
  type = rep("gaussian", length(data)),
  n.burnin = 18000,
  n.draw = 12000,
  prior.gamma = rep(0.5, length(data)),
  sdev = 0.05,
  thin = 3
)
```

Arguments

`data`	List of matrices with maximum of 6 subdatasets.
`N.clust`	Number of clusters.
`type`	Data type corresponding to the list of matrics, which can be gaussian, binomial or possion.
`n.burnin`	An integer value to indicate the number of MCMC burnin.
`n.draw`	An integer value to indicate the number of MCMC draw.
`prior.gamma`	A numerical vector to indicate the prior probability for the indicator variable gamma of each subdataset.
`sdev`	A numerical value to indicate the standard deviation of random walk proposal for the latent variable.
`thin`	A numerical value to thin the MCMC chain in order to reduce autocorrelation.

Value

A list with the following components:

`fit` an object returned by iClusterBayes.

`clust.res` a data.frame storing sample ID and corresponding clusters.

`feat.res` the results of features selection process.

`mo.method` a string value indicating the method used for multi-omics integrative clustering.

References

Mo Q, Shen R, Guo C, Vannucci M, Chan KS, Hilsenbeck SG (2018). A fully Bayesian latent variable model for integrative clustering analysis of multi-type omics data. Biostatistics, 19(1):71-86.

Examples

```
# There is no example and please refer to vignette.
```

`getIntNMF` *Get subtypes from IntNMF*

Description

This function wraps the IntNMF (Integrative Clustering via Non-negative Matrix Factorization) algorithm and provides standard output for 'getMoHeatmap()' and 'getConsensusMOIC()'.

Usage

```
getIntNMF(
  data = NULL,
  N.clust = NULL,
  type = rep("gaussian", length(data)),
  is.binary = rep(FALSE, length(data))
)
```

Arguments

`data`	List of matrices.
`N.clust`	Number of clusters.
`type`	Data type corresponding to the list of matrics, which can be gaussian, binomial or possion.
`is.binary`	A logical vector to indicate if the input for a subset is binary of 0 and 1.

Value

A list with the following components:

`fit` an object returned by nmf.mnnals.

`clust.res` a data.frame storing sample ID and corresponding clusters.

`mo.method` a string value indicating the method used for multi-omics integrative clustering.

References

Chalise P, Fridley BL (2017). Integrative clustering of multi-level omic data based on non-negative matrix factorization algorithm. PLoS One, 12(5):e0176278.

Examples

```
# There is no example and please refer to vignette.
```

`getLRAcluster` *Get subtypes from LRAcluster*

Description

This function wraps the LRAcluster (Integrated cancer omics data anlsysi by low rank approximation) algorithm and provides standard output for 'getMoHeatmap()' and 'getConsensusMOIC()'.

Usage

```
getLRAcluster(
  data = NULL,
  N.clust = NULL,
  type = rep("gaussian", length(data)),
  clusterAlg = "ward.D"
)
```

Arguments

`data`	List of matrices.
`N.clust`	Number of clusters.
`type`	Data type corresponding to the list of matrics, which can be gaussian, binomial or possion; 'gaussian' by default.
`clusterAlg`	A string value to indicate the cluster algorithm for similarity matrix; 'ward.D' by default.

Value

A list with the following components:

`fit` an object returned by LRAcluster.

`clust.res` a data.frame storing sample ID and corresponding clusters.

`clust.dend` a dendrogram of sample clustering.

`mo.method` a string value indicating the method used for multi-omics integrative clustering.

References

Wu D, Wang D, Zhang MQ, Gu J (2015). Fast dimension reduction and integrative clustering of multi-omics data using low-rank approximation: application to cancer molecular classification. BMC Genomics, 16(1):1022.

Examples

```
# There is no example and please refer to vignette.
```

`getMoCluster` *Get subtypes from MoCluster*

Description

This function wraps the MoCluster (Multiple omics data integrative clustering) algorithm and provides standard output for 'getMoHeatmap()' and 'getConsensusMOIC()'.

Usage

```
getMoCluster(
  data = NULL,
  N.clust = NULL,
  type = rep("gaussian", length(data)),
  ncomp = NULL,
  method = "CPCA",
  option = "lambda1",
  k = 10,
  center = TRUE,
  scale = TRUE,
  clusterAlg = "ward.D"
)
```

Arguments

`data`	List of matrices.
`N.clust`	Number of clusters.
`type`	Data type corresponding to the list of matrics, which can be gaussian, binomial or possion.
`ncomp`	An integer value to indicate the number of components to calculate. To calculate more components requires longer computational time.
`method`	A string value can be one of CPCA, GCCA and MCIA; CPCA by default.
`option`	A string value could be one of c('lambda1', 'inertia', 'uniform') to indicate how the different matrices should be normalized.
`k`	A numeric value to indicate the absolute number (if k >= 1) or the proportion (if 0 < k < 1) of non-zero coefficients for the variable loading vectors. It could be a single value or a vector has the same length as x so the sparsity of individual matrix could be different.
`center`	A logical value to indicate if the variables should be centered. TRUE by default.
`scale`	A logical value to indicate if the variables should be scaled. TRUE by default.
`clusterAlg`	A string value to indicate the cluster algorithm for distance.

Value

A list with the following components:

`fit` an object returned by mbpca.

`clust.res` a data.frame storing sample ID and corresponding clusters.

`feat.res` the results of features selection process.

`clust.dend` a dendrogram of sample clustering.

`mo.method` a string value indicating the method used for multi-omics integrative clustering.

References

Meng C, Helm D, Frejno M, Kuster B (2016). moCluster: Identifying Joint Patterns Across Multiple Omics Data Sets. J Proteome Res, 15(3):755-765.

Examples

```
# There is no example and please refer to vignette.
```

`getMoHeatmap` *Get multi-omics comprehensive heatmap*

Description

This function vertically concatenates multiple heatmap derived from each omics data. 'getMoHeatmap' supports customized column annotation and is able to mark the selected features if indicated.

Usage

```
getMoHeatmap(
  data = NULL,
  is.binary = c(FALSE, FALSE, FALSE, FALSE, FALSE, FALSE),
  row.title = c("Data1", "Data2", "Data3", "Data4", "Data5", "Data6"),
  legend.name = c("Data1", "Data2", "Data3", "Data4", "Data5", "Data6"),
  clust.res = NULL,
  clust.dend = NULL,
  show.col.dend = TRUE,
  show.colnames = FALSE,
  show.row.dend = c(TRUE, TRUE, TRUE, TRUE, TRUE, TRUE),
  show.rownames = c(FALSE, FALSE, FALSE, FALSE, FALSE, FALSE),
  clust.dist.row = c("pearson", "pearson", "pearson", "pearson", "pearson", "pearson"),
  clust.method.row = c("ward.D", "ward.D", "ward.D", "ward.D", "ward.D", "ward.D"),
  clust.col = c("#2EC4B6", "#E71D36", "#FF9F1C", "#BDD5EA", "#FFA5AB", "#011627",
    "#023E8A", "9D4EDD"),
  color = rep(list(c("#00FF00", "#000000", "#FF0000")), length(data)),
  annCol = NULL,
  annColors = NULL,
  annRow = NULL,
  width = 6,
  height = 4,
  fig.path = getwd(),
  fig.name = "moheatmap"
)
```

Arguments

data	A list of data frame or matrix storing multiple omics data with rows for features and columns for samples.
is.binary	A logicial vector to indicate if the subdata is binary matrix of 0 and 1 such as mutation.
row.title	A string vector to assign titles for each subdata.
legend.name	A string vector to assign legend title for each subdata.
clust.res	A clust.res object returned by 'getMOIC()' with one specified algorithm or 'get%algorithm_name%' or 'getConsensusMOIC()' with a list of multiple algorithms.
clust.dend	A dendrogram object returned returned by 'getMOIC()' with one specified algorithm or 'get%algorithm_name%' or 'getConsensusMOIC()' with a list of multiple algorithms.
show.col.dend	A logical vector to indicate if showing the dendrogram for column at the top of heatmap.
show.colnames	A logical vector to indicate if showing the names for column at the bottom of heatmap.
show.row.dend	A logical vector to indicate if showing the dendrogram for row of each subdata.
show.rownames	A logical vector to indicate if showing the names for row of each subdata.
clust.dist.row	A string vector to assign distance method for clustering each subdata at feature dimension.
clust.method.row	A string vector to assign clustering method for clustering each subdata at feature dimension.
clust.col	A string vector storing colors for annotating each subtype at the top of heatmap.
color	A list of string vectors storing colors for each subheatmap of subdata.
annCol	A data.frame storing annotation information for samples with exact the same sample order with data parameter.
annColors	A list of string vectors for colors matched with annCol.
annRow	A list of string vectors to indicate which features belong to which subdata should be annotated specifically in subheatmap.
width	An integer value to indicate the width for each subheatmap with unit of cm.
height	An integer value to indicate the height for each subheatmap with unit of cm.
fig.path	A string value to indicate the output path for storing the comprehensive heatmap.
fig.name	A string value to indicate the name of the comprehensive heatmap.

Value

A pdf of multi-omics comprehensive heatmap

References

Gu Z, Eils R, Schlesner M (2016). Complex heatmaps reveal patterns and correlations in multidimensional genomic data. Bioinformatics.

Examples

```
# There is no example and please refer to vignette.
```

getMOIC *Get subtypes from multi-omics integrative clustering*

Description

Using 'getMOIC()', users can choose one out of the ten algorithms embedded in 'MOVICS'. Users can implement multi-omics clustering in a simplest way of which the only requirement is to specify and at least specify a list of matrices (argument of 'data'), a number of cluster (argument of 'N.clust'), and clustering method (argument of 'methodslist') in 'getMOIC()'. It is possible to pass various arguments that are specific to each method. Of course, users can also directly call different algorithms by using functions start with 'get' and end with the name of the algorithm (e.g., 'getSNF'; please refer to '?get

Usage

```
getMOIC(
  data = NULL,
  methodslist = list("SNF", "CIMLR", "PINSPlus", "NEMO", "COCA", "MoCluster",
    "LRAcluster", "ConsensusClustering", "IntNMF", "iClusterBayes"),
  N.clust = NULL,
  type = rep("gaussian", length(data)),
  ...
)
```

Arguments

data	List of matrices (Maximum number of matrices is 6).
methodslist	A string list specifying one or multiple methods to run (See Details).
N.clust	Number of clusters.
type	Data type corresponding to the list of matrics, which can be gaussian, binomial or possion.
...	Additionnal parameters for each method (only works when only one method chosen)

Details

Method for integrative clustering will be chosed according to the value of argument 'methodslist':

If `methodslist == "IntNMF"`, Integrative clustering methods using Non-Negative Matrix Factorization

If `methodslist == "SNF"`, Similarity network fusion.

If `methodslist == "LRAcluster"`, Integrated cancer omics data analysis by low rank approximation.

If `methodslist == "PINSPlus"`, Perturbation Clustering for data integration and disease subtyping

If `methodslist == "ConsensusClustering"`, Consensus clustering

If `methodslist == "NEMO"`, Neighborhood based multi-omics clustering

If `methodslist == "COCA"`, Cluster Of Clusters Analysis

If `methodslist == "CIMLR"`, Cancer Integration via Multikernel Learning (Support Feature Selection)

If `methodslist == "MoCluster"`, Identifying joint patterns across multiple omics data sets (Support Feature Selection)

If `methodslist == "iClusterBayes"`, Integrative clustering of multiple genomic data by fitting a Bayesian latent variable model (Support Feature Selection)

Value

A list of results returned by each specified algorithms.

References

Pierre-Jean M, Deleuze J F, Le Floch E, et al. Clustering and variable selection evaluation of 13 unsupervised methods for multi-omics data integration[J]. Briefings in Bioinformatics, 2019.

intNMF: Chalise P, Fridley BL. Integrative clustering of multi-level omic data based on non-negative matrix factorization algorithm. PLoS One. 2017;12(5):e0176278.

iClusterBayes: Mo Q, Shen R, Guo C, Vannucci M, Chan KS, Hilsenbeck SG. A fully Bayesian latent variable model for integrative clustering analysis of multi-type omics data. Biostatistics. 2018;19(1):71-86.

SNF: Wang B, Mezlini AM, Demir F, et al. Similarity network fusion for aggregating data types on a genomic scale. Nat Methods. 2014;11(3):333-337.

Mocluster: Meng C, Helm D, Frejno M, Kuster B. moCluster: Identifying Joint Patterns Across Multiple Omics Data Sets. J Proteome Res. 2016;15(3):755-765.

LRAcluster: Wu D, Wang D, Zhang MQ, Gu J. Fast dimension reduction and integrative clustering of multi-omics data using low-rank approximation: application to cancer molecular classification. BMC Genomics. 2015;16:1022.

CIMLR: Ramazzotti D, Lal A, Wang B, Batzoglou S, Sidow A. Multi-omic tumor data reveal diversity of molecular mechanisms that correlate with survival. Nat Commun. 2018;9(1):4453.

PINSPlus: Nguyen H, Shrestha S, Draghici S, Nguyen T. PINSPlus: a tool for tumor subtype discovery in integrated genomic data. Bioinformatics. 2019;35(16):2843-2846.

ConsensusClustering: Monti S, Tamayo P, Mesirov J, et al. Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data. Machine Learning. 2003;52:91-118.

NEMO: Rappoport N, Shamir R. NEMO: cancer subtyping by integration of partial multi-omic data. Bioinformatics. 2019;35(18):3348-3356.

COCA: Hoadley KA, Yau C, Wolf DM, et al. Multiplatform analysis of 12 cancer types reveals molecular classification within and across tissues of origin. Cell. 2014;158(4):929-944.

Examples

```
# There is no example and please refer to vignette.
```

getNEMO *Get subtypes from NEMO*

Description

This function wraps the NEMO (Neighborhood based multi-omics clustering) algorithm and provides standard output for ‘getMoHeatmap()‘ and ‘getConsensusMOIC()‘.

Usage

```
getNEMO(
  data = NULL,
  N.clust = NULL,
  type = rep("gaussian", length(data)),
  num.neighbors = NA
)
```

Arguments

data	List of matrices.
N.clust	Number of clusters.
type	Data type corresponding to the list of matrics, which can be gaussian, binomial or possion.
num.neighbors	The number of neighbors to use for each omic.

Value

A list with the following components:

fit an object returned by nemo.clustering.

clust.res a data.frame storing sample ID and corresponding clusters.

mo.method a string value indicating the method used for multi-omics integrative clustering.

References

Rappoport N, Shamir R (2019). NEMO: cancer subtyping by integration of partial multi-omic data. Bioinformatics, 35(18):3348-3356.

Examples

```
# There is no example and please refer to vignette.
```

`getPINSPlus` *Get subtypes from PINSPlus*

Description

This function wraps the PINSPlus (Perturbation Clustering for data INtegration and disease Subtyping) algorithm and provides standard output for 'getMoHeatmap()' and 'getConsensusMOIC()'.

Usage

```
getPINSPlus(
  data = NULL,
  N.clust = NULL,
  type = rep("gaussian", length(data)),
  norMethod = "none",
  clusteringMethod = "kmeans",
  iterMin = 50,
  iterMax = 500
)
```

Arguments

`data`	List of matrices.
`N.clust`	Number of clusters
`type`	Data type corresponding to the list of matrics, which can be gaussian, binomial or possion.
`norMethod`	A string vector indicate the normalization method for consensus clustering.
`clusteringMethod`	The name of built-in clustering algorithm that PerturbationClustering will use. Currently supported algorithm are kmeans, pam and hclust. Default value is "kmeans".
`iterMin`	The minimum number of iterations. Default value is 50
`iterMax`	The maximum number of iterations. Default value is 500.

Value

A list with the following components:

`fit` an object returned by PerturbationClustering.

`clust.res` a data.frame storing sample ID and corresponding clusters.

`mo.method` a string value indicating the method used for multi-omics integrative clustering.

References

Nguyen H, Shrestha S, Draghici S, Nguyen T (2019). PINSPlus: a tool for tumor subtype discovery in integrated genomic data. Bioinformatics, 35(16):2843-2846.

Examples

```
# There is no example and please refer to vignette.
```

getSNF *Get subtypes from SNF*

Description

This function wraps the SNF (Similarity Network Fusion) algorithm and provides standard output for ‘getMoHeatmap()‘ and ‘getConsensusMOIC()‘.

Usage

```
getSNF(
  data = NULL,
  N.clust = NULL,
  type = rep("gaussian", length(data)),
  K = 30,
  t = 20,
  sigma = 0.5
)
```

Arguments

data	List of matrices.
N.clust	Number of clusters.
type	Data type corresponding to the list of matrics, which can be gaussian, binomial or possion.
K	An integer value to indicate the number of neighbors in K-nearest neighbors part of the algorithm.
t	An integer value to indicate the number of interations for the diffusion process.
sigma	A numerical value to indicate the variance for local model.

Value

A list with the following components:

fit an object returned by SNF.

clust.res a data.frame storing sample ID and corresponding clusters.

mo.method a string value indicating the method used for multi-omics integrative clustering.

References

Wang B, Mezlini AM, Demir F, et al (2014). Similarity network fusion for aggregating data types on a genomic scale. Nat Methods, 11(3):333-337.

Examples

```
# There is no example and please refer to vignette.
```

getStdiz *Get standardized omics data*

Description

This function prepare standardized data for generating heatmap. Omics data, especially for expression, should be centered or scaled or z-scored (both centered and scaled). Generally, DNA methylation beta matrix and somatic mutation (0 and 1 binary matrix) should not be normalized. This function also provides an argument of 'halfwidth' for continuous omics data; such argument is used to truncate the 'extremum' after normalization; specifically, normalized values that exceed the halfwidth boundaries will be replaced by the halfwidth, which is vary useful to map colors in heatmap.

Usage

```
getStdiz(
  data = NULL,
  halfwidth = rep(1, length(data)),
  centerFlag = rep(TRUE, length(data)),
  scaleFlag = rep(TRUE, length(data))
)
```

Arguments

data	A list of data.frame or matrix storing raw multiple omics data with rows for features and columns for samples.
halfwidth	A numeric vector to assign marginal cutoff for truncating values in data
centerFlag	A logical vector to indicate if each subdata should be centered
scaleFlag	A logical vector to indicate if each subdata should be scaled

Value

A standardized data.frame containing multi-omics data.

Examples

```
# There is no example and please refer to vignette.
```

runDEA *Run differential expression analysis*

Description

Using choosen algorithm to run differential expression analysis between two classes identified by multi-omics clustering process.

Usage

```
runDEA(
  dea.method = c("deseq2", "edger", "limma"),
  expr = NULL,
  moic.res = NULL,
  prefix = NULL,
  overwt = FALSE,
  sort.p = TRUE,
  verbose = TRUE,
  res.path = getwd()
)
```

Arguments

dea.method	A string value to indicate the algorithm for differential expression analysis. Allowed value contains c('deseq2', 'edger', 'limma'). The former two require RNA-Seq raw count data and the last one requires normalized expression data (FPKM or TPM without log2 transformation is recommended).
expr	A matrix of expression data.
moic.res	An object returned by 'getMOIC()' with one specified algorithm or 'get%algorithm_name%' or 'getConsensusMOIC()' with a list of multiple algorithms.
prefix	A string value to indicate the prefix of output file.
overwt	A logic value to indicate if to overwrite existing results; FALSE by default.
sort.p	A logic value to indicate if to sort adjusted p value for output table; TRUE by default.
verbose	A logic value to indicate if to only output id, log2fc, pvalue, and padj; TRUE by default.
res.path	A string value to indicate the path for saving the results.

Details

runDEA

Value

Several .txt files storing differential expression analysis results by specified algorithm

References

Love, M.I., Huber, W., Anders, S. (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol, 15(12):550-558.

Robinson MD, McCarthy DJ and Smyth GK (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26(1):139-140.

McCarthy DJ, Chen Y, Smyth GK (2012). Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. Nucleic Acids Res. 40(10):4288-4297.

Ritchie, ME, Phipson, B, Wu, D, Hu, Y, Law, CW, Shi, W, and Smyth, GK (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Res, 43(7):e47.

Examples

```
# There is no example and please refer to vignette.
```

---

runGSEA *Run identification of unique functional pathways*

Description

Use gene set enrichment analysis to identify subtype-specific (overexpressed or downexpressed) functional pathways for each subtype identified by multi-omics clustering algorithms.

Usage

```
runGSEA(
  moic.res = NULL,
  dea.method = c("deseq2", "edger", "limma"),
  norm.expr = NULL,
  prefix = NULL,
  dat.path = getwd(),
  res.path = getwd(),
  dirct = "up",
  n.path = 10,
  msigdb.path = NULL,
  nPerm = 1000,
  minGSSize = 10,
  maxGSSize = 500,
  p.cutoff = 0.05,
  p.adj.cutoff = 0.25,
  gsva.method = "gsva",
  norm.method = "mean",
  clust.col = c("#2EC4B6", "#E71D36", "#FF9F1C", "#BDD5EA", "#FFA5AB", "#011627",
    "#023E8A", "9D4EDD"),
  color = NULL,
  fig.name = NULL,
  fig.path = getwd(),
  width = 15,
  height = 10
)
```

Arguments

moic.res	An object returned by 'getMOIC()' with one specified algorithm or 'get%algorithm_name%' or 'getConsensusMOIC()' with a list of multiple algorithms.
dea.method	A string value to indicate the algorithm for differential expression analysis. Allowed value contains c('deseq2', 'edger', 'limma').
norm.expr	A matrix of normalized expression data with rows for genes and columns for samples; FPKM or TPM without log2 transformation is recommended.
prefix	A string value to indicate the prefix of differential expression file (use for searching files).

dat.path	A string value to indicate the path for saving the files of differential expression analysis.
res.path	A string value to indicate the path for saving the results for identifying subtype-specific functional pathways.
dirct	A string value to indicate the direction of identifying significant pathway. Allowed values contain c('up', 'down'); 'up' means up-regulated pathway, and 'down' means down-regulated pathway
n.path	A integer value to indicate how many top pathways sorted by NES should be identified for each subtypes; 10 by default.
msigdb.path	A string value to indicate ABSOULUTE PATH/NAME of MSigDB file (GMT file with gene symbols) downloaded from https://www.gsea-msigdb.org/gsea/msigdb/collections.jsp#H.
nPerm	A integer value to indicate the number of permutations; 1000 by default and 10000 will be better for reproducibility.
minGSSize	A integer value to indicate minimal size of each geneSet for analyzing; 10 by default.
maxGSSize	A integer value to indicate maximal size of each geneSet for analyzing; 500 by default.
p.cutoff	A numeric value to indicate the nominal p value for identifying significant pathways; pvalue < 0.05 by default.
p.adj.cutoff	A numeric value to indicate the adjusted p value for identifying significant pathways; padj < 0.05 by default.
gsva.method	A string value to indicate the method to employ in the estimation of gene-set enrichment scores per sample. By default this is set to gsva (Hänzelmann et al, 2013) and other options are ssgsea (Barbie et al, 2009), zscore (Lee et al, 2008) or plage (Tomfohr et al, 2005). The latter two standardize first expression profiles into z-scores over the samples and, in the case of zscore, it combines them together as their sum divided by the square-root of the size of the gene set, while in the case of plage they are used to calculate the singular value decomposition (SVD) over the genes in the gene set and use the coefficients of the first right-singular vector as pathway activity profile.
norm.method	A string value to indicate how to calculate subtype-specific pathway enrichment scores. Allowed values contain c('mean', 'median'); mean by default.
clust.col	A string vector storing colors for annotating each subtype at the top of heatmap.
color	A string vector storing colors for heatmap.
fig.name	A string value to indicate the name of the pathway heatmap.
fig.path	A string value to indicate the output path for storing the pathway heatmap.
width	A numeric value to indicate the width of output figure.
height	A numeric value to indicate the height of output figure.

Value

A figure of subtype-specific pathway heatmap (.pdf) and a list with the following components:

gsea.list a list storing gses object returned by GSEA for each subtype.

raw.es a data.frame storing raw enrichment score of identified subtype-specific pathways by using specified gsva.method.

`scaled.es` a data.frame storing scaled enrichment score of identified subtype-specific pathways by using specified `gsva.method`.

`grouped.es` a data.frame storing grouped enrichment score (mean or median value among each subtype) by using specified `norm.method`.

References

Barbie, D.A. et al. (2009). Systematic RNA interference reveals that oncogenic KRAS-driven cancers require TBK1. Nature, 462(5):108-112.

Hänzelmann, S., Castelo, R. and Guinney, J. (2013). GSVA: Gene set variation analysis for microarray and RNA-Seq data. BMC Bioinformatics, 14(1):7.

Lee, E. et al. (2008). Inferring pathway activity toward precise disease classification. PLoS Comp Biol, 4(11):e1000217.

Tomfohr, J. et al. (2005). Pathway level analysis of gene expression using singular value decomposition. BMC Bioinformatics, 6(1):1-11.

Yu G, Wang L, Han Y, He Q (2012). clusterProfiler: an R package for comparing biological themes among gene clusters. OMICS, 16(5):284-287.

Gu Z, Eils R, Schlesner M (2016). Complex heatmaps reveal patterns and correlations in multidimensional genomic data. Bioinformatics, 32(18):2847–2849.

Examples

```
# There is no example and please refer to vignette.
```

`runKappa` *Run consistency evaluation using Kappa statistic*

Description

Calculate Kappa statistic to measure the consistency between two appraisements

Usage

```
runKappa(
  subt1 = NULL,
  subt2 = NULL,
  subt1.lab = NULL,
  subt2.lab = NULL,
  fig.path = getwd(),
  fig.name = "constheatmap",
  width = 5,
  height = 5
)
```

Arguments

subt1	A numeric vector to indicate the first appraisement. Order should be exactly the same with subt2 for each sample.
subt2	A numeric vector to indicate the second appraisement. Order should be exactly the same with subt1 for each sample.
subt1.lab	A string value to indicate the label of the first subtype.
subt2.lab	A string value to indicate the label of the second subtype.
fig.path	A string value to indicate the output path for storing the consistency heatmap.
fig.name	A string value to indicate the name of the consistency heatmap.
width	A numeric value to indicate the width of output figure.
height	A numeric value to indicate the height of output figure.

Details

This function evaluates the consistency between two appraisements that targets to the same cohort. For example, the NTP-predicted subtype amd PAM-predicted subtype of external cohort, or the current subtype and predicted subtype of discovery cohort. Therefore, the arguments 'subt1' and 'subt2' can be the 'clust' column of 'clust.res' derived from 'getMOIC()' with one specified algorithm or 'get%algorithm_name%' or 'getConsensusMOIC()' with a list of multiple algorithms or 'runNTP()' or 'runPAM()'. However, subtypes identified from different algorithm (i.e., 'get%algorithm_name1%' and 'get%algorithm_name2%') can not be evaluated because the subtype 1 identified from the first algorithm may not be the same subtype 1 from the second algorithm.

Value

A figure of consistency heatmap (.pdf).

Examples

```
# There is no example and please refer to vignette.
```

---

runMarker *Run identification of unique biomarkers*

Description

This function aims to identify uniquely and significantly expressed (overexpressed or downexpressed) biomarkers for each subtype identified by multi-omics integrative clustering algorithms. Top markers will be chosen to generate a template so as to run nearest template prediction for subtype verification.

Usage

```
runMarker(
  moic.res = NULL,
  dea.method = c("deseq2", "edger", "limma"),
  prefix = NULL,
  dat.path = getwd(),
  res.path = getwd(),
```

```
  p.cutoff = 0.05,
  p.adj.cutoff = 0.05,
  dirct = "up",
  n.marker = 200,
  doplot = TRUE,
  norm.expr = NULL,
  annCol = NULL,
  annColors = NULL,
  clust.col = c("#2EC4B6", "#E71D36", "#FF9F1C", "#BDD5EA", "#FFA5AB", "#011627",
    "#023E8A", "9D4EDD"),
  halfwidth = 3,
  centerFlag = TRUE,
  scaleFlag = TRUE,
  show_rownames = FALSE,
  show_colnames = FALSE,
  color = c("#5bc0eb", "black", "#ECE700"),
  fig.path = getwd(),
  fig.name = NULL,
  width = 8,
  height = 8,
  ...
)
```

Arguments

moic.res	An object returned by 'getMOIC()' with one specified algorithm or 'get%algorithm_name%' or 'getConsensusMOIC()' with a list of multiple algorithms.
dea.method	A string value to indicate the algorithm for differential expression analysis. Allowed value contains c('deseq2', 'edger', 'limma').
prefix	A string value to indicate the prefix of differential expression file (use for searching files).
dat.path	A string value to indicate the path for saving the files of differential expression analysis.
res.path	A string value to indicate the path for saving the results for identifying subtype-specific markers.
p.cutoff	A numeric value to indicate the nominal p value for identifying significant markers; pvalue < 0.05 by default.
p.adj.cutoff	A numeric value to indicate the adjusted p value for identifying significant markers; padj < 0.05 by default.
dirct	A string value to indicate the direction of identifying significant marker. Allowed values contain c('up', 'down'); 'up' means up-regulated marker, and 'down' means down-regulated marker.
n.marker	A integer value to indicate how many top markers sorted by log2fc should be identified for each subtype; 200 by default.
doplot	A logic value to indicate if generating heatmap by using subtype-specific markers. TRUE by default.
norm.expr	A matrix of normalized expression data with rows for genes and columns for samples; FPKM or TPM without log2 transformation is recommended.
annCol	A data.frame storing annotation information for samples.

annColors	A list of string vectors for colors matched with annCol.
clust.col	A string vector storing colors for annotating each subtype at the top of heatmap.
halfwidth	A numeric vector to assign marginal cutoff for truncating values in data
centerFlag	A logical vector to indicate if each subdata should be centered
scaleFlag	A logical vector to indicate if each subdata should be scaled
show_rownames	A logic value to indicate if showing rownames (feature names) in heatmap.
show_colnames	A logic value to indicate if showing colnames (sample ID) in heatmap.
color	A string vector storing colors for heatmap.
fig.path	A string value to indicate the output path for storing the marker heatmap.
fig.name	A string value to indicate the name of the marker heatmap.
width	A numeric value to indicate the width of output figure.
height	A numeric value to indicate the height of output figure.
...	Additional parameters pass to pheatmap.

Value

A figure of subtype-specific marker heatmap (.pdf) if `doPlot = TRUE` and a list with the following components:

`unqlist` a string vector storing the unique marker across all subtypes.

`templates` a data.frame storing the the template information for nearest template prediction, which is used for verification in external cohort.

`dirct` a string value indicating the direction for identifying subtype-specific markers.

References

Gu Z, Eils R, Schlesner M (2016). Complex heatmaps reveal patterns and correlations in multidimensional genomic data. Bioinformatics, 32(18):2847-2849.

Examples

```
# There is no example and please refer to vignette.
```

---

runNTP *Run nearest template prediction*

Description

Using Nearest Template Prediction (NTP) based on predefined templates derived from current identified subtypes to assign potential subtype label on external cohort.

Usage

```
runNTP(
  expr = NULL,
  templates = NULL,
  scale = TRUE,
  center = TRUE,
  nPerm = 1000,
  distance = "cosine",
  seed = 123456,
  verbose = TRUE,
  doPlot = FALSE,
  fig.path = getwd(),
  fig.name = "ntpheatmap",
  width = 5,
  height = 5
)
```

Arguments

expr	A numeric matrix with row features and sample columns; data is recommended to be z-scored.
templates	A data frame with at least two columns; class (coerced to factor) and probe (coerced to character).
scale	A logic value to indicate if the expr should be further scaled. FALSE by default.
center	A logic value to indicate if the expr should be further centered. FALSE by default.
nPerm	An integer value to indicate the permutations for p-value estimation.
distance	A string value to indicate the distance measurement. Allowed values contain c('cosine', 'pearson', 'spearman', 'kendall').
seed	An integer value for p-value reproducibility.
verbose	A logic value to indicate whether console messages are to be displayed.
doPlot	A logic value to indicate whether to produce prediction heatmap.
fig.path	A string value to indicate the output path for storing the nearest template prediction heatmap.
fig.name	A string value to indicate the name of the nearest template prediction heatmap.
width	A numeric value to indicate the width of output figure.
height	A numeric value to indicate the height of output figure.

Value

A figure of predictive heatmap by NTP (.pdf) and a list with the following components:

`ntp.res` a data.frame storing the results of nearest template prediction (see ntp).

`clust.res` similar to 'clust.res' returned by 'getMOIC()' or 'get

`mo.method` a string value indicating the method used for prediction.

References

Hoshida, Y. (2010). Nearest Template Prediction: A Single-Sample-Based Flexible Class Prediction with Confidence Assessment. PLoS ONE 5, e15543.

Examples

```
# There is no example and please refer to vignette.
```

---

runPAM *Run partition around medoids classifier*

Description

Using partition around medoids (PAM) classifier to predict potential subtype label on external cohort and calculate in-group proportions (IGP) statistics.

Usage

```
runPAM(
  train.expr = NULL,
  moic.res = NULL,
  test.expr = NULL,
  gene.subset = NULL
)
```

Arguments

train.expr	A matrix of normalized expression training data with rows for genes and columns for samples; FPKM or TPM without log2 transformation is recommended.
moic.res	An object returned by 'getMOIC()' with one specified algorithm or 'get%algorithm_name%' or 'getConsensusMOIC()' with a list of multiple algorithms.
test.expr	A matrix of normalized expression testing data with rows for genes and columns for samples; FPKM or TPM without log2 transformation is recommended.
gene.subset	A string vector to indicate a subset of genes to be used. #' @return A list with the following components:
	IGP a named numeric vector storing the in-group proportion (see IGP.clusterRepro).
	clust.res similar to 'clust.res' returned by 'getMOIC()' or 'get
	mo.method a string value indicating the method used for prediction.

Details

This function first trains a partition around medoids (PAM) classifier in the discovery (training) cohort to predict the subtype for patients in the external validation (testing) cohort, and each sample in the validation cohort was assigned to a subtype label whose centroid had the highest Pearson correlation with the sample. Finally, the in-group proportion (IGP) statistic will be performed to evaluate the similarity and reproducibility of the acquired subtypes between discovery and validation cohorts.

References

Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu Diagnosis of multiple cancer types by shrunken centroids of gene expression PNAS 99: 6567-6572.

Kapp A V, Tibshirani R. (2007). Are clusters found in one dataset present in another dataset?. Biostatistics, 8(1):9-31.

Examples

```
# There is no example and please refer to vignette.
```

Index