Package ‘TCGAbiolinks’

May 30, 2024

Type  Package
Title  TCGAbiolinks: An R/Bioconductor package for integrative analysis with GDC data
Version  2.32.0
Date  2024-01-01
Author  Antonio Colaprico,
        Tiago Chedraoui Silva,
        Catharina Olsen,
        Luciano Garofano,
        Davide Garolini,
        Claudia Cava,
        Thais Sabedot,
        Thathiane Malta,
        Stefano M. Pagnotta,
        Isabella Castiglioni,
        Michele Ceccarelli,
        Gianluca Bontempi,
        Houtan Noushmehr

Maintainer  Tiago Chedraoui Silva <tiagochst@gmail.com>,
            Antonio Colaprico <axc1833@med.miami.edu>

Depends  R (>= 4.0)
Imports  downloader (>= 0.4), grDevices, biomaRt, dplyr, graphics, tibble, GenomicRanges, XML (>= 3.98.0), data.table, jsonlite
         (>= 1.0.0), plyr, knitr, methods, ggplot2, stringr (>= 1.0.0),
         IRanges, rvest (>= 0.3.0), stats, utils, S4Vectors, R.utils,
         SummarizedExperiment (>= 1.4.0), TCGAbiolinksGUI.data (>=
         1.15.1), readr, tools, tidyr, purrr, xml2, httr (>= 1.2.1)

Description  The aim of TCGAbiolinks is: i) facilitate the GDC open-access
             data retrieval, ii) prepare the data using the appropriate pre-processing
             strategies, iii) provide the means to carry out different standard analyses
             and iv) to easily reproduce earlier research results. In more detail, the package
             provides multiple methods for analysis (e.g., differential expression analysis,
             identifying differentially methylated regions) and methods for visualization
             (e.g., survival plots, volcano plots, starburst plots) in order to easily
develop complete analysis pipelines.

**License** GPL (>= 3)

**biocViews** DNAmethylation, DifferentialMethylation, GeneRegulation, GeneExpression, MethylationArray, DifferentialExpression, Pathways, Network, Sequencing, Survival, Software

**Suggests** jpeg, png, BiocStyle, rmarkdown, devtools, maftools, parmigene, c3net, minet, Biobase, affy, testthat, sesame, AnnotationHub, ExperimentHub, pathview, clusterProfiler, Seurat, ComplexHeatmap, circlize, ConsensusClusterPlus, igraph, supraHex, limma, edgeR, sva, EDASeq, survminer, genefilter, gridExtra, survival, doParallel, parallel, ggrepel (>= 0.6.3), scales, grid, DT

**VignetteBuilder** knitr

**LazyData** true

**URL** https://github.com/BioinformaticsFMRP/TCGAbiolinks

**BugReports** https://github.com/BioinformaticsFMRP/TCGAbiolinks/issues

**RoxygenNote** 7.3.1

**Encoding** UTF-8

**git_url** https://git.bioconductor.org/packages/TCGAbiolinks

**git_branch** RELEASE_3_19

**git_last_commit** c270a75

**git_last_commit_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-05-29

**Contents**

```
TCGAbiolinks-package ........................................... 4
batch.info ......................................................... 5
bcgsc.ca_CHOL_IlluminaHiSeq_DNASeq.1.somatic.maf ............... 5
chol_maf .......................................................... 5
classification ...................................................... 6
clinBRCA ........................................................... 6
clinical.biotab .................................................... 6
colDataPrepare .................................................... 7
dataBRCA ........................................................... 7
dataDEGsFiltLevel ................................................ 7
dataREAD .......................................................... 8
dataREAD_df ......................................................... 8
DE_PCBC_stemSig .................................................. 8
dmc.non.parametric ................................................ 9
dmc.non.parametric.se .......................................... 10
```
EB_PCBC_stemSig ................................................. 11
ECTO_PCBC_stemSig .............................................. 11
gaiaCNVplot ...................................................... 12
gbm.exp.harmonized ........................................... 12
gbm.exp.legacy ................................................... 13
GDCdownload ....................................................... 13
GDCprepare ........................................................ 14
GDCprepare_clinic ............................................... 16
GDCquery ............................................................ 17
GDCquery_ATAC_seq ............................................. 21
GDCquery_clinic ................................................... 22
geneInfo ............................................................ 24
geneInfoHT ........................................................ 25
GenesCutID ........................................................ 26
GeneSplitRegulon .................................................. 26
get.GRCh.bioMart .................................................. 27
getAdjacencyBiogrid ............................................. 27
getDataCategorySummary ...................................... 28
getGDCInfo ........................................................ 29
getGDCprojects ................................................... 29
getGistic ............................................................. 30
getLinkedOmicsData ............................................. 30
getManifest ........................................................ 33
getMC3MAF ........................................................ 33
getNbCases ........................................................ 34
getNbFiles ........................................................ 34
getProjectSummary .............................................. 35
getResults .......................................................... 36
getSampleFilesSummary ......................................... 36
getTSS ............................................................... 37
get_IDs ............................................................... 38
ggbiplot ............................................................. 38
gliomaClassifier .................................................. 40
isServeOK .......................................................... 41
matchedMetExp .................................................... 41
MESO_PCBC_stemSig .............................................. 42
met.gbm.27k ........................................................ 42
msi_results ........................................................ 42
pancan2018 ........................................................ 43
PanCancerAtlas_subtypes ....................................... 43
SC_PCBC_stemSig .................................................. 43
splitAPICall ........................................................ 44
TabSubtypesCol_merged ........................................ 44
tabSurvKMcompleteDEGs ...................................... 45
TCGAanalyze_analyseGRN ...................................... 45
TCGAanalyze_Clustering ........................................ 46
TCGAanalyze_DEA ................................................ 46
TCGAanalyze_DEA_Affy ......................................... 49
The aim of TCGAbiolinks is: i) facilitate the TCGA open-access data retrieval, ii) prepare the data using the appropriate pre-processing strategies, iii) provide the means to carry out different standard analyses and iv) allow the user to download a specific version of the data and thus to easily reproduce earlier research results. In more detail, the package provides multiple methods for analysis (e.g., differential expression analysis, identifying differentially methylated regions) and methods for visualization (e.g., survival plots, volcano plots, starburst plots) in order to easily develop complete analysis pipelines.
**batch.info**

**Description**

The functions you’re likely to need from TCGAbiolinks is GDCdownload, GDCquery. Otherwise refer to the vignettes to see how to format the documentation.

**See Also**

Useful links:

- [https://github.com/BioinformaticsFMRP/TCGAbiolinks](https://github.com/BioinformaticsFMRP/TCGAbiolinks)
- Report bugs at [https://github.com/BioinformaticsFMRP/TCGAbiolinks/issues](https://github.com/BioinformaticsFMRP/TCGAbiolinks/issues)

---

**batch.info**

*TCGA batch information from Biospecimen Metadata Browser*

---

**bcgsc.ca_CHOL.IlluminaHiSeq_DNASeq.1.somatic.maf**

*TCGA CHOL MAF*

---

**chol_maf**

*TCGA CHOL MAF transformed to maftools object*

---

**Description**

TCGA CHOL MAF transformed to maftools object

**Format**

An object of class MAF
<table>
<thead>
<tr>
<th>classification</th>
<th>Result of gliomaclassifier function</th>
</tr>
</thead>
</table>

**Description**

Result of gliomaclassifier function

**Format**

A list of data frames

<table>
<thead>
<tr>
<th>clinBRCA</th>
<th>Clinical data TCGA BRCA</th>
</tr>
</thead>
</table>

**Description**

Clinical data TCGA BRCA

**Format**

A data frame with 1061 rows and 109 variables

<table>
<thead>
<tr>
<th>clinical.biotab</th>
<th>A list of data frames with clinical data parsed from XML (code in vignettes)</th>
</tr>
</thead>
</table>

**Description**

A list of data frames with clinical data parsed from XML (code in vignettes)

**Format**

A list with 7 elements
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>colDataPrepare</code></td>
<td>Create samples information matrix for GDC samples add subtype information</td>
<td>A data frame with 20531 rows (genes) and 50 variables (samples)</td>
</tr>
<tr>
<td><code>dataBRCA</code></td>
<td>TCGA data matrix BRCA</td>
<td></td>
</tr>
<tr>
<td><code>dataDEGsFiltLevel</code></td>
<td>TCGA data matrix BRCA DEGs</td>
<td></td>
</tr>
</tbody>
</table>
**dataREAD**  
*TCGA data SummarizedExperiment READ*

**Description**

TCGA data SummarizedExperiment READ

**Format**

A SummarizedExperiment of READ with 2 samples

**dataREAD_df**  
*TCGA data matrix READ*

**Description**

TCGA data matrix READ

**Format**

A data frame with 20531 rows (genes) and 2 variables (samples)

**DE_PCBC_stemSig**  
*A numeric vector with SC-derived definitive endoderm (DE) signature trained on PCBC’s dataset*

**Description**

A numeric vector with SC-derived definitive endoderm (DE) signature trained on PCBC’s dataset

**Format**

A numeric vector with 12956 genes
Perform non-parametric wilcoxon test

Usage

dmc.non.parametric(
  matrix,
  idx1 = NULL,
  idx2 = NULL,
  paired = FALSE,
  adj.method = "BH",
  alternative = "two.sided",
  cores = 1
)

Arguments

matrix A matrix
idx1 Index columns group1
idx2 Index columns group2
paired Do a paired wilcoxon test? Default: True
adj.method P-value adjustment method. Default:"BH" Benjamini-Hochberg
alternative wilcoxon test alternative
cores Number of cores to be used

Value

Data frame with p-values and diff mean

Examples

nrows <- 200; ncols <- 20
counts <- matrix(
  runif(nrows * ncols, 1, 1e4), nrows,
  dimnames = list(paste0("cg",1:200),paste0("S",1:20))
)
TCGAbiolinks:::dmc.non.parametric(counts,1:10,11:20)
**Calculate pvalues**

**Description**

Calculate pvalues using wilcoxon test

**Usage**

```r
dmc.non.parametric.se(
  data,
  groupCol = NULL,
  group1 = NULL,
  group2 = NULL,
  paired = FALSE,
  adj.method = "BH",
  alternative = "two.sided",
  cores = 1
)
```

**Arguments**

- `data`: SummarizedExperiment obtained from the TCGAPrepare
- `groupCol`: Columns with the groups inside the SummarizedExperiment object. (This will be obtained by the function colData(data))
- `group1`: In case our object has more than 2 groups, you should set the groups
- `group2`: In case our object has more than 2 groups, you should set the groups
- `paired`: Do a paired wilcoxon test? Default: True
- `adj.method`: P-value adjustment method. Default: "BH" Benjamini-Hochberg
- `alternative`: wilcoxon test alternative
- `cores`: Number of cores to be used

**Details**

Verify if the data is significant between two groups. For the methylation we search for probes that have a difference in the mean methylation and also a significant value. Input: A SummarizedExperiment object that will be used to compared two groups with wilcoxon test, a boolean value to do a paired or non-paired test Output: p-values (non-adj/adj) histograms, p-values (non-adj/adj)

**Value**

Data frame with cols p values/p values adjusted

Data frame with two cols p-values/p-values adjusted
EB_PCBC_stemSig

Examples

```r
nrows <- 200; ncols <- 20
counts <- matrix(runif(nrows * ncols, 1, 1e4), nrows,
dimnames = list(paste0("cg",1:200),LETTERS[1:20]))
rowRanges <- GenomicRanges::GRanges(rep(c("chr1", "chr2"), c(50, 150)),
    IRanges::IRanges(floor(runif(200, 1e5, 1e6)), width=100),
    strand=sample(c("+", "-"), 200, TRUE),
    feature_id=sprintf("ID%03d", 1:200))
colData <- S4Vectors::DataFrame(Treatment=rep(c("ChIP", "Input"), 10),
    row.names=LETTERS[1:20],
    group=rep(c("group1","group2"),c(10,10)))
data <- SummarizedExperiment::SummarizedExperiment(
    assays=S4Vectors::SimpleList(counts=counts),
    rowRanges=rowRanges,
    colData=colData)
results <- TCGAbiolinks:::dmc.non.parametric.se(data,"group")
```

EB_PCBC_stemSig  
A numeric vector with stem cell (SC)-derived embryoid bodies (EB) signature trained on PCBC’s dataset

Description

A numeric vector with stem cell (SC)-derived embryoid bodies (EB) signature trained on PCBC’s dataset

Format

A numeric vector with 12956 genes

ECTO_PCBC_stemSig  
A numeric vector with SC-derived ectoderm (ECTO) signature trained on PCBC’s dataset

Description

A numeric vector with SC-derived ectoderm (ECTO) signature trained on PCBC’s dataset

Format

A numeric vector with 12956 genes
**gaiaCNVplot**

*Creates a plot for GAIA output (all significant aberrant regions.)*

**Description**

This function is an auxiliary function to visualize GAIA output (all significant aberrant regions.)

**Usage**

```r
gaiaCNVplot(calls, threshold = 0.01)
```

**Arguments**

- **calls**: A matrix with the following columns: Chromosome, Aberration Kind, Region Start, Region End, Region Size, and score.
- **threshold**: Score threshold (orange horizontal line in the plot)

**Value**

A plot with all significant aberrant regions.

**Examples**

```r
call <- data.frame("Chromosome" = rep(9, 100),
  "Aberration Kind" = rep(c(-2, -1, 0, 1, 2), 20),
  "Region Start [bp]" = 18259823:18259922,
  "Region End [bp]" = 18259823:18259922,
  "score" = rep(c(1, 2, 3, 4), 25))
gaiaCNVplot(call, threshold = 0.01)
call <- data.frame("Chromosome" = rep(c(1, 9), 50),
  "Aberration Kind" = rep(c(-2, -1, 0, 1, 2), 20),
  "Region Start [bp]" = 18259823:18259922,
  "Region End [bp]" = 18259823:18259922,
  "score" = rep(c(1, 2, 3, 4), 25))
gaiaCNVplot(call, threshold = 0.01)
```

**gbm.exp.harmonized**

*A RangedSummarizedExperiment two samples with gene expression data from vignette aligned against hg38*

**Description**

A RangedSummarizedExperiment two samples with gene expression data from vignette aligned against hg38

**Format**

A RangedSummarizedExperiment: 56963 genes, 2 samples
**Description**

A RangedSummarizedExperiment two samples with gene expression data from vignette aligned against hg19

**Format**

A RangedSummarizedExperiment: 21022 genes, 2 samples

---

**GDCdownload**

*Download GDC data*

**Description**

Uses GDC API or GDC transfer tool to download gdc data The user can use query argument The data from query will be save in a folder: project/data.category

**Usage**

```r
gdcdownload(
    query,
    token.file,
    method = "api",
    directory = "GDCdata",
    files.per.chunk = NULL
)
```

**Arguments**

- **query**
  A query for GDCquery function
- **token.file**
  Token file to download controlled data (only for method = "client")
- **method**
  Uses the API (POST method) or gdc client tool. Options "api", "client". API is faster, but the data might get corrupted in the download, and it might need to be executed again
- **directory**
  Directory/Folder where the data was downloaded. Default: GDCdata
- **files.per.chunk**
  This will make the API method only download n (files.per.chunk) files at a time. This may reduce the download problems when the data size is too large. Expected a integer number (example files.per.chunk = 6)
## GDCprepare

**Prepare GDC data**

### Description

Reads the data downloaded and prepare it into an R object.

### Examples

```r
## Not run:
# Download clinical data from XML
query <- GDCquery(project = "TCGA-COAD", data.category = "Clinical")
GDCdownload(query, files.per.chunk = 200)
query <- GDCquery(
  project = "TARGET-AML",
  data.category = "Transcriptome Profiling",
  data.type = "miRNA Expression Quantification",
  workflow.type = "BCGSC miRNA Profiling",
  barcode = c("TARGET-20-PARUDL-03A-01R", "TARGET-20-PASRRB-03A-01R")
)
# data will be saved in:
# example_data_dir/TARGET-AML/harmonized/Transcriptome_Profiling/miRNA_Expression_Quantification
GDCdownload(query, method = "client", directory = "example_data_dir")
query_acc_gbm <- GDCquery(
  project = c("TCGA-ACC", "TCGA-GBM"),
  data.category = "Transcriptome Profiling",
  data.type = "Gene Expression Quantification",
  workflow.type = "STAR - Counts"
)
GDCdownload(
  query = query_acc_gbm,
  method = "api",
  directory = "example",
  files.per.chunk = 50
)
## End(Not run)
```
Usage

GDCprepare(
  query,
  save = FALSE,
  save.filename,
  directory = "GDCdata",
  summarizedExperiment = TRUE,
  remove.files.prepared = FALSE,
  add.gistic2.mut = NULL,
  mutant_variant_classification = c("Frame_Shift_Del", "Frame_Shift_Ins",
                                "Missense_Mutation", "Nonsense_Mutation", "Splice_Site", "In_Frame_Del",
                                "In_Frame_Ins", "Translation_Start_Site", "Nonstop_Mutation")
)

Arguments

query       A query for GDCquery function
save        Save result as RData object?
save.filename Name of the file to be save if empty an automatic will be created
directory   Directory/Folder where the data was downloaded. Default: GDCdata
summarizedExperiment Create a summarizedExperiment? Default TRUE (if possible)
remove.files.prepared Remove the files read? Default: FALSE This argument will be considered only if save argument is set to true
add.gistic2.mut If a list of genes (gene symbol) is given, columns with gistic2 results from GDAC firehose (hg19) and a column indicating if there is or not mutation in that gene (hg38) (TRUE or FALSE - use the MAF file for more information) will be added to the sample matrix in the summarized Experiment object.
mutant_variant_classification List of mutant_variant_classification that will be consider a sample mutant or not. Default: "Frame_Shift_Del", "Frame_Shift_Ins", "Missense_Mutation", "Nonsense_Mutation", "Splice_Site", "In_Frame_Del", "In_Frame_Ins", "Translation_Start_Site", "Nonstop_Mutation"

Value

A summarizedExperiment or a data.frame

Author(s)

Tiago Chedraoui Silva
GDCprepare_clinic

Examples

```r
## Not run:
query <- GDCquery(
  project = "TCGA-KIRP",
  data.category = "Simple Nucleotide Variation",
  data.type = "Masked Somatic Mutation"
)
GDCdownload(query, method = "api", directory = "maf")
maf <- GDCprepare(query, directory = "maf")

## End(Not run)
```

GDCprepare_clinic Parsing clinical xml files

Description

This function receives the query argument and parses the clinical xml files based on the desired information

Usage

GDCprepare_clinic(query, clinical.info, directory = "GDCdata")

Arguments

- `query`: Result from GDCquery, with data.category set to Clinical
- `clinical.info`: Which information should be retrieved. Options Clinical: drug, admin, follow_up, radiation, patient, stage_event or new_tumor_event Options Biospecimen: protocol, admin, aliquot, analyte, bio_patient, sample, portion, slide
- `directory`: Directory/Folder where the data was downloaded. Default: GDCdata

Value

A data frame with the parsed values from the XML

Examples

```r
query <- GDCquery(
  project = "TCGA-COAD",
  data.category = "Clinical",
  data.format = "bcr xml",
  barcode = c("TCGA-RU-A8FL","TCGA-AA-3972")
)
GDCdownload(query)
clinical <- GDCprepare_clinic(query,"patient")
clinical.drug <- GDCprepare_clinic(query,"drug")
```
GDCquery

clinical.radiation <- GDCprepare_clinic(query,"radiation")
clinical.admin <- GDCprepare_clinic(query,"admin")
## Not run:
query <- GDCquery(
  project = "TCGA-COAD",
  data.category = "Biospecimen",
  data.format = "bcr xml",
  data.type = "Biospecimen Supplement",
  barcode = c("TCGA-RU-A8FL","TCGA-AA-3972")
)
GDCdownload(query)
clinical <- GDCprepare_clinic(query,"admin")
clinical.drug <- GDCprepare_clinic(query,"sample")
clinical.radiation <- GDCprepare_clinic(query,"portion")
clinical.admin <- GDCprepare_clinic(query,"slide")
## End(Not run)

GDCquery  Query GDC data

Description
Uses GDC API to search for search, it searches for both controlled and open-access data. For GDC
data arguments project, data.category, data.type and workflow.type should be used Please, see the
vignette for a table with the possibilities.

Usage

GDCquery(
  project,
  data.category,
  data.type,
  workflow.type,
  access,
  platform,
  barcode,
  data.format,
  experimental.strategy,
  sample.type
)

Arguments

project A list of valid project (see list with TCGAbiolinks:::getGDCprojects()$project_id)

- BEATAML1.0-COHORT
- BEATAML1.0-CRENOLANIB
- CGCI-BLGSP
- CPTAC-2
- CPTAC-3
- CTSP-DLBCL1
- FM-AD
- HCMI-CMDC
- MMRF-COMMPASS
- NCICCR-DLBCL
- OHSU-CNL
- ORGANOID-PANCREATIC
- TARGET-ALL-P1
- TARGET-ALL-P2
- TARGET-ALL-P3
- TARGET-AML
- TARGET-CCSK
- TARGET-NBL
- TARGET-OS
- TARGET-RT
- TARGET-WT
- TCGA-ACC
- TCGA-BLCA
- TCGA-BRCA
- TCGA-CESC
- TCGA-CHOL
- TCGA-COAD
- TCGA-DLBC
- TCGA-ESCA
- TCGA-GBM
- TCGA-HNSC
- TCGA-KICH
- TCGA-KIRC
- TCGA-KIRP
- TCGA-LAML
- TCGA-LGG
- TCGA-LIHC
- TCGA-LUAD
- TCGA-LUSC
- TCGA-MESO
- TCGA-OV
- TCGA-PAAD
- TCGA-PCPG
- TCGA-PRAD
- TCGA-READ
• TCGA-SARC
• TCGA-SKCM
• TCGA-STAD
• TCGA-TGCT
• TCGA-THCA
• TCGA-THYM
• TCGA-UCEC
• TCGA-UCS
• TCGA-UVM
• VAREPOP-APOLLO

data.category A valid project (see list with TCGAbiolinks:::getProjectSummary(project)) For the complete list please check the vignette. List for harmonized database:
• Biospecimen
• Clinical
• Copy Number Variation
• DNA Methylation
• Sequencing Reads
• Simple Nucleotide Variation
• Transcriptome Profiling

data.type A data type to filter the files to download For the complete list please check the vignette.

workflow.type GDC workflow type

access Filter by access type. Possible values: controlled, open

platform Example:

CGH- 1x1M_G4447A IlluminaGA_RNASeqV2
AgilentG4502A_07 IlluminaGA_mRNA_DGE
Human1MDuo HumanMethylation450
HG-CGH-415K_G4124A IlluminaGA_miRNASeq
HumanHap550 IlluminaHiSeq_miRNASeq
ABI H-miRNA_8x15K
HG-CGH-244A SOLID_DNASeq
IlluminaDNAmethylation_OMA003_CPI IlluminaGA_DNASeq_automated
IlluminaDNAmethylation_OMA002_CPI HG-U133_Plus_2
HuEx- 1_0-st-v2 Mixed_DNASeq
H-miRNA_8x15Kv2 IlluminaGA_DNASeq_curated
MDA_RPPA_Core IlluminaHiSeq_TotalRNASeqV2
HT_HG-U133A IlluminaHiSeq_DNASeq_automated
diagnostic_images microsat_i
IlluminaHiSeq_RNASeq SOLiD_DNASeq_curated
IlluminaHiSeq_DNASeqC Mixed_DNASeq_curated
IlluminaGA_RNASeq IlluminaGA_DNASeq_Cont_automated
IlluminaGA_DNASeq IlluminaHiSeq_WGBS
pathology_reports IlluminaHiSeq_DNASeq_Cont_automated
Genomic_Wide_SNP_6    bio    
tissue_images    Mixed_DNASeq_automated
HumanMethylation27    Mixed_DNASeq_Cont_curated
IlluminaHiSeq_RNASeqV2    Mixed_DNASeq_Cont

barcode  A list of barcodes to filter the files to download

data.format  Data format filter ("VCF", "TXT", "BAM","SVS","BCR XML","BCR SSF XML","TSV","BCR Auxiliary XML","BCR OMF XML","BCR Biotab","MAF","BCR PPS XML","XLSX")

experimental.strategy  Filter to experimental strategy. Harmonized: WXS, RNA-Seq, miRNA-Seq, Genotyping Array.

sample.type  A sample type to filter the files to download

Value

A data frame with the results and the parameters used

Author(s)

Tiago Chedraoui Silva

Examples

query <- GDCquery(
  project = "TCGA-ACC",
  data.category = "Copy Number Variation",
  data.type = "Copy Number Segment"
)

## Not run:
query <- GDCquery(
  project = "TARGET-AML",
  data.category = "Transcriptome Profiling",
  data.type = "miRNA Expression Quantification",
  workflow.type = "BCGSC miRNA Profiling",
  barcode = c("TARGET-20-PARUDL-03A-01R", "TARGET-20-PASRRB-03A-01R")
)
query <- GDCquery(
  project = "TARGET-AML",
  data.category = "Transcriptome Profiling",
  data.type = "Gene Expression Quantification",
  workflow.type = "STAR - Counts",
  barcode = c("TARGET-20-PADZCG-04A-01R", "TARGET-20-PARJCR-09A-01R")
)
query <- GDCquery(
  project = "TCGA-ACC",
  data.category = "Copy Number Variation",
  data.type = "Masked Copy Number Segment",
  sample.type = c("Primary Tumor")
)
GDCquery_ATAC_seq

Retrieve open access ATAC-seq files from GDC server

Description

Usage
GDCquery_ATAC_seq(tumor = NULL, file.type = NULL)

Arguments
- tumor: a valid tumor
- file.type: Write maf file into a csv document

Value
A data frame with the maf file information

Examples
query <- GDCquery_ATAC_seq(file.type = "txt")
## Not run:
GDCdownload(query)

## End(Not run)
query <- GDCquery_ATAC_seq(tumor = "BRCA", file.type = "bigWigs")
## Not run:
GDCdownload(query, method = "client")

## End(Not run)
GDCquery_clinic

Get GDC clinical data

Description

GDCquery_clinic will download all clinical information from the API as the one with using the button from each project.

Usage

GDCquery_clinic(project, type = "clinical", save.csv = FALSE)

Arguments

project  A valid project (see list with getGDCprojects()$project_id)
  • BEATAML1.0-COHORT
  • BEATAML1.0-CRENOLANIB
  • CGCI-BLGSP
  • CPTAC-2
  • CPTAC-3
  • CTSP-DLBCL1
  • FM-AD
  • HCMI-CMDC
  • MMRF-COMMPASS
  • NCICCR-DLBCL
  • OHSU-CNL
  • ORGANOID-PANCREATIC
  • TARGET-ALL-P1
  • TARGET-ALL-P2
  • TARGET-ALL-P3
  • TARGET-AML
  • TARGET-CCSK
  • TARGET-NBL
  • TARGET-OS
  • TARGET-RT
  • TARGET-WT
  • TCGA-ACC
  • TCGA-BLCA
  • TCGA-BRCA
  • TCGA-CESC
  • TCGA-CHOL
  • TCGA-COAD
- TCGA-DLBC
- TCGA-ESCA
- TCGA-GBM
- TCGA-HNSC
- TCGA-KICH
- TCGA-KIRC
- TCGA-KIRP
- TCGA-LAML
- TCGA-LGG
- TCGA-LIHC
- TCGA-LUAD
- TCGA-LUSC
- TCGA-MESO
- TCGA-OV
- TCGA-PAAD
- TCGA-PCPG
- TCGA-PRAD
- TCGA-READ
- TCGA-SARC
- TCGA-SKCM
- TCGA-STAD
- TCGA-TGCT
- TCGA-THCA
- TCGA-THYM
- TCGA-UCEC
- TCGA-UCS
- TCGA-UVM
- VAREPOP-APOLLO

**type**
A valid type. Options "clinical", "Biospecimen" (see list with getGDCprojects()$project_id)

**save.csv**
Write clinical information into a csv document

**Value**
A data frame with the clinical information

**Author(s)**
Tiago Chedraoui Silva
Examples

```r
clinical <- GDCquery_clinic(
  project = "TCGA-ACC",
  type = "clinical",
  save.csv = FALSE
)
clinical <- GDCquery_clinic(
  project = "TCGA-ACC",
  type = "biospecimen",
  save.csv = FALSE
)
## Not run:
clinical_cptac_3 <- GDCquery_clinic(
  project = "CPTAC-3",
  type = "clinical"
)
clinical_cptac_2 <- GDCquery_clinic(
  project = "CPTAC-2",
  type = "clinical"
)
clinical_HCMI_CMDC <- GDCquery_clinic(
  project = "HCMI-CMDC",
  type = "clinical"
)
clinical_GCI_HTMCP_CC <- GDCquery_clinic(
  project = "CGCI-HTMCP-CC",
  type = "clinical"
)
clinical <- GDCquery_clinic(
  project = "NCICCR-DLBCL",
  type = "clinical"
)
clinical <- GDCquery_clinic(
  project = "ORGANOID-PANCREATIC",
  type = "clinical"
)
## End(Not run)
```

---

**geneInfo**

geneInfo for normalization of RNAseq data

**Description**

geneInfo for normalization of RNAseq data

**Format**

A data frame with 20531 rows and 2 variables
Description

Code to generate the data in examples

Format

A data frame with 23486 rows and 2 variables

Examples

```r
## Not run:
library(EDASeq)
library(biomaRt)
# get ensembl gene IDs for hg38
ensembl <- useMart("ensembl", dataset = "hsapiens_gene_ensembl")
biomart_getID <- getBM(attributes = c("ensembl_gene_id"), mart = ensembl)
# get gene length and GC content for all IDs
step <- 500
geneInfoHT <- plyr::adply(seq(1,length(biomart_getID$ensembl_gene_id),step),.margins = 1,.fun = function(x){
    begin <- x
    end <- x + step
    if(!any(length(biomart_getID$ensembl_gene_id)) end <- length(biomart_getID$ensembl_gene_id)
    file <- paste0("geneInfoHT_from_",begin,"_to_",end,".rda")
    if(!file.exists(file)){
        df <- getGeneLengthAndGCContent(biomart_getID$ensembl_gene_id[begin:end], org="hsa", mode = c("biomart"))
        save(df,file = file)
    } else {
        df <- get(load(file))
    }
}
)

## End(Not run)
```
### GenesCutID

<table>
<thead>
<tr>
<th>Description</th>
<th>GenesCutID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td>GenesCutID(GeneList)</td>
</tr>
<tr>
<td>Arguments</td>
<td>GeneList</td>
</tr>
<tr>
<td>Value</td>
<td>list of gene symbol without IDs</td>
</tr>
</tbody>
</table>

### GeneSplitRegulon

<table>
<thead>
<tr>
<th>Description</th>
<th>GeneSplitRegulon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td>GeneSplitRegulon(Genelist, Sep)</td>
</tr>
<tr>
<td>Arguments</td>
<td>Genelist, Sep</td>
</tr>
<tr>
<td>Value</td>
<td>GeneSplitRegulon</td>
</tr>
</tbody>
</table>
**get.GRCh.bioMart**

*Get hg19 gene annotation or hg38 (gencode v36)*

**Description**

Get hg19 (from biomart) or hg38 (gencode v36) gene annotation

**Usage**

```r
get.GRCh.bioMart(genome = c("hg19", "hg38"), as.granges = FALSE)
```

**Arguments**

- **genome**: hg38 or hg19
- **as.granges**: Output as GRanges or data.frame

**getAdjacencyBiogrid**

*Get a matrix of interactions of genes from biogrid*

**Description**

Using biogrid database, it will create a matrix of gene interactions. If columns A and row B has value 1, it means the gene A and gene B interacts.

**Usage**

```r
getAdjacencyBiogrid(tmp.biogrid, names.genes = NULL)
```

**Arguments**

- **tmp.biogrid**: Biogrid table
- **names.genes**: List of genes to filter from output. Default: consider all genes

**Value**

A matrix with 1 for genes that interacts, 0 for no interaction.
getDataCategorySummary

Create a Summary table for each sample in a project saying if it contains or not files for a certain data category

Usage

ggetDataCategorySummary(project)

Arguments

project A GDC project

Value

A data frame

Author(s)

Tiago Chedraoui Silva

Examples

summary <- getDataCategorySummary("TCGA-ACC")
getGDCInfo

Description

Check GDC server status using the api https://api.gdc.cancer.gov/status

Usage

getGDCInfo()

Value

Return true all status

Examples

info <- getGDCInfo()

getGDCprojects

Retrieve all GDC projects

Description

getGDCprojects uses the following api to get projects https://api.gdc.cancer.gov/projects

Usage

getGDCprojects()

Value

A data frame with last GDC projects

Examples

projects <- getGDCprojects()
getGistic

Description

Download GISTIC data from firehose from http://gdac.broadinstitute.org/runs/analyses__latest/data/

Usage

getGistic(disease, type = "thresholded")

Arguments

disease: TCGA disease. Option available in http://gdac.broadinstitute.org/runs/analyses__latest/data/
type: Results type: thresholded or data

getLinkedOmicsData

Description

Retrieve linkedOmics data from http://linkedomics.org/

Usage

getLinkedOmicsData(project, dataset)

Arguments

project: A linkedOmics project:
  • TCGA-ACC
  • TCGA-BLCA
  • TCGA-BRCA
  • TCGA-CESC
  • TCGA-CHOL
  • TCGA-COADREAD
  • TCGA-DLBC
  • TCGA-ESCA
  • TCGA-GBM
  • TCGA-GBMLGG
  • TCGA-HNSC
  • TCGA-KICH
  • TCGA-KIPAN
• TCGA-KIRC
• TCGA-KIRP
• TCGA-LAML
• TCGA-LGG
• TCGA-LIHC
• TCGA-LUAD
• TCGA-LUSC
• TCGA-MESO
• TCGA-OV
• TCGA-PAAD
• TCGA-PCPG
• TCGA-PRAD
• TCGA-SARC
• TCGA-SKCM
• TCGA-STAD
• TCGA-STES
• TCGA-TGCT
• TCGA-THCA
• TCGA-THYM
• TCGA-UCEC
• TCGA-UCS
• TCGA-UVM
• CPTAC-COAD

dataset A dataset from the list below
• Annotated mutation
• Clinical
• Glycoproteome (Gene level)
• Glycoproteome (Site level)
• Methylation (CpG-site level, HM27)
• Methylation (CpG-site level, HM450K)
• Methylation (Gene level, HM27)
• Methylation (Gene level, HM450K)
• miRNA (GA, Gene level)
• miRNA (GA, Isoform level)
• miRNA (GA, miRgene level)
• miRNA (Gene level)
• miRNA (HiSeq, Gene level)
• miRNA (HiSeq, miRgene level)
• miRNA (isoform level)
• miRNA (miRgene level)
• Mutation (Gene level)
• Mutation (Site level)
• Mutation raw file (Somatic and MSIndel)
• Phosphoproteome (Gene level)
• Phosphoproteome (Site level)
• Phosphoproteomics (Normal)
• Phosphoproteomics (Tumor)
• Proteome (Gene level)
• Proteome (Gene Level)
• Proteome (JHU, Gene level)
• Proteome (PNNL, Gene level, Normal TMT Unshared Log Ratio)
• Proteome (PNNL, Gene level, Tumor TMT Unshared Log Ratio)
• Proteome (PNNL, Gene level)
• Proteome (VU, Gene level, Label-free Unshared Counts)
• RNAseq (GA, Gene level)
• RNAseq (HiSeq, Gene level)
• RPPA (Analyte level)
• RPPA (Analyte Level)
• RPPA (Gene level)
• RPPA (Gene Level)
• SCNV (Focal level, log-ratio)
• SCNV (Focal level, Thresholded)
• SCNV (Gene level, log ratio)
• SCNV (Gene level, log-ratio)
• SCNV (Gene level, Thresholded)
• SCNV (Segment level)

Value

A matrix with the data

Examples

```r
## Not run:
TCGA_COAD_protein <- getLinkedOmicsData(
  project = "TCGA-COADREAD",
  dataset = "Proteome (Gene level)"
)
TCGA_COAD_RNASeq_hiseq <- getLinkedOmicsData(
  project = "TCGA-COADREAD",
  dataset = "RNAseq (HiSeq, Gene level)"
)
TCGA_COAD_RNASeq_ga <- getLinkedOmicsData(
  project = "TCGA-COADREAD",
  dataset = "RNAseq (GA, Gene level)"
)
TCGA_COAD_RPPA <- getLinkedOmicsData(
  project = "TCGA-COADREAD",
  dataset = "RPPA (Gene level)"
)
```
## End(Not run)

```
getManifest

Get a Manifest from GDCquery output that can be used with GDC-client
```

### Description

Get a Manifest from GDCquery output that can be used with GDC-client

### Usage

```r
getManifest(query, save = FALSE)
```

### Arguments

- `query`: A query for GDCquery function
- `save`: Write Manifest to a txt file (tab separated)

### Examples

```r
query <- GDCquery(
  project = "TARGET-AML",
  data.category = "Transcriptome Profiling",
  data.type = "Gene Expression Quantification",
  workflow.type = "STAR - Counts",
  barcode = c("TARGET-20-PADZCG-04A-01R", "TARGET-20-PARJCR-09A-01R")
)
getManifest(query)
```

---

```
getMC3MAF

Retrieve open access mc3 MAF file from GDC server
```

### Description


### Usage

```r
getMC3MAF()
```

### Value

A data frame with the MAF file information from https://gdc.cancer.gov/about-data/publications/mc3-2017
getNbCases

Get Number of cases in GDC for a project

Description

Get Number of cases in GDC for a project

Usage

getNbCases(project, data.category)

Arguments

project A GDC project
data.category A GDC project data category

Author(s)

Tiago Chedraoui Silva

Examples

## Not run:
getNbCases("TCGA-ACC","Clinical")
getNbCases("CPTAC-2","Clinical")
## End(Not run)

getNbFiles

Get Number of files in GDC for a project

Description

Get Number of files in GDC for a project

Usage

getNbFiles(project, data.category)
**getProjectSummary**

**Arguments**

- `project` A GDC project
- `data.category` A GDC project data category

**Author(s)**

Tiago Chedraoui Silva

**Examples**

```r
## Not run:
getNbFiles("TCGA-ACC","Clinical")
getNbFiles("CPTAC-2","Clinical")

## End(Not run)
```

---

**getProjectSummary**  
*Get Project Summary from GDC*

**Description**

Get Project Summary from GDC

**Usage**

```r
getProjectSummary(project)
```

**Arguments**

- `project` A GDC project

**Author(s)**

Tiago Chedraoui Silva

**Examples**

```r
getProjectSummary("TCGA-ACC")
## Not run:
getProjectSummary("CPTAC-2")

## End(Not run)
```
**getResults**

*Get the results table from query*

**Description**

Get the results table from query, it can select columns with cols argument and return a number of rows using rows argument.

**Usage**

getResults(query, rows, cols)

**Arguments**

- **query**: A object from GDCquery
- **rows**: Rows identifiers (row numbers)
- **cols**: Columns identifiers (col names)

**Value**

Table with query results

**Examples**

```r
query <- GDCquery(
  project = "TCGA-GBM",
  data.category = "Transcriptome Profiling",
  data.type = "Gene Expression Quantification",
  workflow.type = "STAR - Counts",
  barcode = c("TCGA-14-0736-02A-01R-2005-01", "TCGA-06-0211-02A-02R-2005-01")
)
results <- getResults(query)
```

---

**getSampleFilesSummary**

*Retrieve summary of files per sample in a project*

**Description**

Retrieve the number of files under each data_category + data_type + experimental_strategy + platform Almost like https://portal.gdc.cancer.gov/exploration

**Usage**

getSampleFilesSummary(project, files.access = NA)
getTSS

Arguments

project A GDC project
files.access Filter by file access ("open" or "controlled"). Default: no filter

Value

A data frame with the maf file information

Author(s)

Tiago Chedraoui Silva

Examples

summary <- getSampleFilesSummary("TCGA-UCS")
## Not run:
summary <- getSampleFilesSummary(c("TCGA-OV","TCGA-ACC"))
## End(Not run)

description

getTSS to fetch GENCODE gene annotation (transcripts level) from Bioconductor package biomaRt If upstream and downstream are specified in TSS list, promoter regions of GENCODE gene will be generated.

Usage

getTSS(
    genome = c("hg38", "hg19"),
    TSS = list(upstream = NULL, downstream = NULL)
)

Arguments

genome Which genome build will be used: hg38 (default) or hg19.
TSS A list. Contains upstream and downstream like TSS=list(upstream, downstream). When upstream and downstream is specified, coordinates of promoter regions with gene annotation will be generated.
Value

GENCODE gene annotation if TSS is not specified. Coordinates of GENCODE gene promoter regions if TSS is specified.

Examples

# get GENCODE gene annotation (transcripts level)
## Not run:
getTSS <- getTSS()
getTSS <- getTSS(genome.build = "hg38", TSS=list(upstream=1000, downstream=1000))
## End(Not run)

Description

get_IDs allows user to extract metadata from barcodes. The dataframe returned has columns for 'project', 'tss', 'participant', 'sample', "portion", "plate", and "center"

Usage

get_IDs(data)

Arguments

data numeric matrix, each row represents a gene, each column represents a sample

Value

data frame with columns 'project', 'tss', 'participant', 'sample', "portion", "plate", "center", "condition"

Description

Biplot for Principal Components using ggplot2

ggbioplot
Usage

ggbplot(
  pcobj,
  choices = 1:2,
  scale = 1,
  pc.biplot = TRUE,
  obs.scale = 1 - scale,
  var.scale = scale,
  groups = NULL,
  ellipse = FALSE,
  ellipse.prob = 0.68,
  labels = NULL,
  labels.size = 3,
  alpha = 1,
  var.axes = TRUE,
  circle = FALSE,
  circle.prob = 0.69,
  varname.size = 3,
  varname.adjust = 1.5,
  varname.abbrev = FALSE
)

Arguments

pcobj an object returned by prcomp() or princomp()
choices which PCs to plot
scale covariance biplot (scale = 1), form biplot (scale = 0). When scale = 1, the inner product between the variables approximates the covariance and the distance between the points approximates the Mahalanobis distance.
pc.biplot for compatibility with biplot.princomp()
obs.scale scale factor to apply to observations
var.scale scale factor to apply to variables
groups optional factor variable indicating the groups that the observations belong to. If provided the points will be colored according to groups
ellipse draw a normal data ellipse for each group?
ellipse.prob size of the ellipse in Normal probability
labels optional vector of labels for the observations
labels.size size of the text used for the labels
alpha alpha transparency value for the points (0 = transparent, 1 = opaque)
var.axes draw arrows for the variables?
circle draw a correlation circle? (only applies when prcomp was called with scale = TRUE and when var.scale = 1)
circle.prob definition of circle.prob
varname.size size of the text for variable names
gliomaClassifier

```
varname.adjust  adjustment factor the placement of the variable names, >= 1 means farther from
the arrow
varname.abbrev  whether or not to abbreviate the variable names

Value

A ggplot2 plot

Author(s)

Vincent Q. Vu.
```

gliomaClassifier  Gliomar classifier

Description

Classify DNA methylation gliomas using data from https://doi.org/10.1016/j.cell.2015.12.028

Usage

gliomaClassifier(data)

Arguments

data  DNA methylation matrix or Summarized Experiments with samples on columns
and probes on the rows

Value

A list of 3 data frames: 1) Sample final classification 2) Each model final classification 3) Each
class probability of classification

Author(s)

Tiago Chedraoui Silva, Tathiane Malta, Houtan Noushmehr

Examples

```r
## Not run:
query <- GDCquery(
  project= "TCGA-GBM",
  data.category = "DNA methylation",
  barcode = c("TCGA-06-0122","TCGA-14-1456"),
  platform = "Illumina Human Methylation 27",
  legacy = TRUE
)
GDCdownload(query)
data.hg19 <- GDCprepare(query)
```
classification <- gliomaClassifier(data.hg19)

# Comparing results
TCGAquery_subtype("GBM") %>%
dplyr::filter(patient %in% c("TCGA-06-0122","TCGA-14-1456")) %>%
dplyr::select("patient","Supervised.DNA.Methylation.Cluster")

## End(Not run)

---

**isServeOK**  
*Check GDC server status is OK*

**Description**  
Check GDC server status using the api https://api.gdc.cancer.gov/status

**Usage**  
`isServeOK()`

**Value**  
Return true if status is ok

**Examples**  
```r
status <- isServeOK()
```

---

**matchedMetExp**  
*Get GDC primary tumors samples with both DNA methylation (HM450K) and Gene expression data*

**Description**  
For a given TCGA project it gets the primary tumors samples (barcode) with both DNA methylation and Gene expression data from GDC database

**Usage**  
`matchedMetExp(project, n = NULL)`

**Arguments**
- `project`  
  A GDC project
- `n`  
  Number of samples to return. If NULL return all (default)
### $msi\_results$

**Value**

A vector of barcodes

**Examples**

```r
# Get ACC samples with both DNA methylation (HM450K) and gene expression aligned to hg19
samples <- matchedMetExp("TCGA-UCS")
```

<table>
<thead>
<tr>
<th><strong>MESO_PCBC_stemSig</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>A numeric vector with SC-derived mesoderm (MESO) signature trained on PCBC's dataset</td>
</tr>
</tbody>
</table>

**Description**

A numeric vector with SC-derived mesoderm (MESO) signature trained on PCBC's dataset

**Format**

A numeric vector with 12956 genes

<table>
<thead>
<tr>
<th><strong>met_gbm_27k</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>A DNA methylation RangedSummarizedExperiment for 8 samples (only first 20 probes) aligned against hg19</td>
</tr>
</tbody>
</table>

**Description**

A DNA methylation RangedSummarizedExperiment for 8 samples (only first 20 probes) aligned against hg19

**Format**

A RangedSummarizedExperiment: 20 probes, 8 samples

<table>
<thead>
<tr>
<th><strong>msi_results</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>MSI data for two samples</td>
</tr>
</tbody>
</table>

**Description**

MSI data for two samples

**Format**

A data frame: 2 rows, 4 columns
**Description**

A data frame with all TCGA molecular subtypes

**Format**

A data frame with 7,734 lines and 10 columns

---

**PanCancerAtlas_subtypes**

*Retrieve table with TCGA molecular subtypes*

**Description**

PanCancerAtlas_subtypes is a curated table with molecular subtypes for 24 TCGA cancer types

**Usage**

PanCancerAtlas_subtypes()

**Value**

A data.frame with barcode and molecular subtypes for 24 cancer types

**Examples**

molecular.subtypes <- PanCancerAtlas_subtypes()

---

**SC_PCBC_stemSig**

*A numeric vector with stem cell-like signature trained on PCBC’s dataset*

**Description**

A numeric vector with stem cell-like signature trained on PCBC’s dataset

**Format**

A numeric vector with 12956 genes
splitAPICall

**Description**

internal function to break a huge API call into smaller ones so it respects the max character limit of a string.

**Usage**

```r
splitAPICall(FUN, step = 20, items)
```

**Arguments**

- **FUN**: function that calls the API
- **step**: How many items to be evaluated per API call
- **items**: vector of items to be using within the function (list of barcodes, aliquot ids, etc)

**Author(s)**

Tiago Chedraoui Silva

---

**TabSubtypesCol_merged**

TCGA samples with their Pam50 subtypes

**Description**

A dataset containing the Sample Ids from TCGA and PAM50 subtyping attributes of 4768 tumor patients.

**Usage**

```r
TabSubtypesCol_merged
```

**Format**

A data frame with 4768 rows and 3 variables:

- **samples**: Sample ID from TCGA barcodes, character string
- **subtype**: Pam50 classification, character string
- **color**: color, character string ...
Description

TCGAanalyze_analyseGRN perform gene regulatory network.

Usage

TCGAanalyze_analyseGRN(TFs, normCounts, kNum)

Arguments

- **TFs**: a vector of genes.
- **normCounts**: is a matrix of gene expression with genes in rows and samples in columns.
- **kNum**: the number of nearest neighbors to consider to estimate the mutual information. Must be less than the number of columns of normCounts.

Value

an adjacent matrix
TCGAanalyze_Clustering

*Hierarchical cluster analysis*

**Description**

Hierarchical cluster analysis using several methods such as `ward.D'`, `ward.D2`, `single`, `complete`, `average` (= UPGMA), `mcquitty` (= WPGMA), `median` (= WPGMC) or `centroid` (= UPGMC).

**Usage**

```r
TCGAanalyze_Clustering(tabDF, method, methodHC = "ward.D2")
```

**Arguments**

- `tabDF`: is a dataframe or numeric matrix, each row represents a gene, each column represents a sample come from TCGAPrepare.
- `method`: is method to be used for generic cluster such as `hclust` or `consensus`.
- `methodHC`: is method to be used for Hierarchical cluster.

**Value**

- object of class `hclust` if method selected is `hclust`. If method selected is `Consensus` returns a list of length `maxK` (maximum cluster number to evaluate.). Each element is a list containing `consensusMatrix` (numerical matrix), `consensusTree` (hclust), `consensusClass` (consensus class assignments). ConsensusClusterPlus also produces images.

---

**TCGAanalyze_DEA**

*Differential expression analysis (DEA) using edgeR or limma package.*

**Description**

TCGAanalyze_DEA allows user to perform Differentially expression analysis (DEA), using edgeR package or limma to identify differentially expressed genes (DEGs). It is possible to do a two-class analysis.

TCGAanalyze_DEA performs DEA using following functions from edgeR:

1. `edgeR::DGEList` converts the count matrix into an edgeR object.
2. `edgeR::estimateCommonDisp` each gene gets assigned the same dispersion estimate.
3. `edgeR::exactTest` performs pair-wise tests for differential expression between two groups.
4. `edgeR::topTags` takes the output from `exactTest()`, adjusts the raw p-values using the False Discovery Rate (FDR) correction, and returns the top differentially expressed genes.
TCGAanalyze_DEA performs DEA using following functions from limma:

1. `limma::makeContrasts` construct matrix of custom contrasts.
2. `limma::lmFit` Fit linear model for each gene given a series of arrays.
3. `limma::contrasts.fit` Given a linear model fit to microarray data, compute estimated coefficients and standard errors for a given set of contrasts.
4. `limma::eBayes` Given a microarray linear model fit, compute moderated t-statistics, moderated F-statistic, and log-odds of differential expression by empirical Bayes moderation of the standard errors towards a common value.
5. `limma::toptable` Extract a table of the top-ranked genes from a linear model fit.

Usage

```r
TCGAanalyze_DEA(
  mat1,
  mat2,
  metadata = TRUE,
  Cond1type,
  Cond2type,
  pipeline = "edgeR",
  method = "exactTest",
  fdr.cut = 1,
  logFC.cut = 0,
  batch.factors = NULL,
  ClinicalDF = data.frame(),
  paired = FALSE,
  log.trans = FALSE,
  voom = FALSE,
  trend = FALSE,
  MAT = data.frame(),
  contrast.formula = "",
  Condtypes = c()
)
```

Arguments

- `mat1` numeric matrix, each row represents a gene, each column represents a sample with `Cond1type`
- `mat2` numeric matrix, each row represents a gene, each column represents a sample with `Cond2type`
- `metadata` Add metadata
- `Cond1type` a string containing the class label of the samples in `mat1` (e.g., control group)
- `Cond2type` a string containing the class label of the samples in `mat2` (e.g., case group)
- `pipeline` a string to specify which package to use ("limma" or "edgeR")
method is 'glmLRT' (1) or 'exactTest' (2) used for edgeR (1) Fit a negative binomial generalized log-linear model to the read counts for each gene (2) Compute gene-wise exact tests for differences in the means between two groups of negatively binomially distributed counts.

fdr.cut is a threshold to filter DEGs according their p-value corrected

logFC.cut is a threshold to filter DEGs according their logFC

batch.factors is a vector containing strings to specify options for batch correction. Options are "Plate", "TSS", "Year", "Portion", "Center", and "Patients"

ClinicalDF is a dataframe returned by GDCquery_clinic() to be used to extract year data

paired is a boolean to account for paired or non-paired samples. Set to TRUE for paired case

log.trans is a boolean to perform log cpm transformation. Set to TRUE for log transformation

voom is a boolean to perform voom transformation for limma-voom pipeline. Set to TRUE for voom transformation

trend is a boolean to perform limma-trend pipeline. Set to TRUE to go through limma-trend

MAT is a matrix containing expression set as all samples in columns and genes as rows. Do not provide if mat1 and mat2 are used

contrast.formula is a string input to determine coefficients and to design contrasts in a customized way

Condtypes is a vector of grouping for samples in MAT

Value is a table with DEGs containing for each gene logFC, logCPM, pValue, and FDR, also for each contrast

Examples

dataNorm <- TCGAbiolinks::TCGAnalyze_Normalization(dataBRCA, geneInfo)
dataFilt <- TCGAnalyze_Filtering(tabDF = dataBRCA, method = "quantile", qnt.cut = 0.25)
samplesNT <- TCGAquery_SampleTypes(colnames(dataFilt), typesample = c("NT"))
samplesTP <- TCGAquery_SampleTypes(colnames(dataFilt), typesample = c("TP"))
dataDEGs <- TCGAnalyze_DEA(
  mat1 = dataFilt[,samplesNT],
  mat2 = dataFilt[,samplesTP],
  Condt1type = "Normal",
  Condt2type = "Tumor"
)
**TCGAanalyze_DEA_Affy**  
*Differentially expression analysis (DEA) using limma package.*

**Description**
Differentially expression analysis (DEA) using limma package.

**Usage**
```r
TCGAanalyze_DEA_Affy(AffySet, FC.cut = 0.01)
```

**Arguments**
- **AffySet**: A matrix-like data object containing log-ratios or log-expression values for a series of arrays, with rows corresponding to genes and columns to samples.
- **FC.cut**: write

**Value**
List of list with tables in 2 by 2 comparison of the top-ranked genes from a linear model fitted by DEA's limma.

**Examples**
```r
## Not run:
to add example
## End(Not run)
```

**TCGAanalyze_DMC**  
*Differentially methylated regions Analysis*

**Description**
This function will search for differentially methylated CpG sites, which are regarded as possible functional regions involved in gene transcriptional regulation.

In order to find these regions we use the beta-values (methylation values ranging from 0.0 to 1.0) to compare two groups.

Firstly, it calculates the difference between the mean methylation of each group for each probes.
Secondly, it calculates the p-value using the wilcoxon test using the Benjamini-Hochberg adjustment method. The default parameters will require a minimum absolute beta values delta of 0.2 and a false discovery rate (FDR)-adjusted Wilcoxon rank-sum P-value of < 0.01 for the difference.

After these analysis, we save a volcano plot (x-axis:diff mean methylation, y-axis: significance) that will help the user identify the differentially methylated CpG sites and return the object with the calculus in the rowRanges.

If the calculus already exists in the object it will not recalculated. You should set overwrite parameter to TRUE to force it, or remove the columns with the results from the object.
Usage

TCGAanalyze_DMC(
    data,
    groupCol = NULL,
    group1 = NULL,
    group2 = NULL,
    alternative = "two.sided",
    diffmean.cut = 0.2,
    paired = FALSE,
    adj.method = "BH",
    plot.filename = "methylation_volcano.pdf",
    ylab = expression(paste(-Log[10], " (FDR corrected P-values)")),
    xlab = expression(paste("DNA Methylation difference (", beta, "," ","-values)")),
    title = NULL,
    legend = "Legend",
    color = c("black", "red", "darkgreen"),
    label = NULL,
    xlim = NULL,
    ylim = NULL,
    p.cut = 0.01,
    probe.names = FALSE,
    cores = 1,
    save = TRUE,
    save.directory = ".",
    filename = NULL
)

Arguments

data SummarizedExperiment obtained from the TCGAPrepare

groupCol Columns with the groups inside the SummarizedExperiment object. (This will
    be obtained by the function colData(data))

group1 In case our object has more than 2 groups, you should set the name of the group

group2 In case our object has more than 2 groups, you should set the name of the group

alternative wilcoxon test alternative

diffmean.cut diffmean threshold. Default: 0.2

paired Wilcoxon paired parameter. Default: FALSE

adj.method Adjusted method for the p-value calculation

plot.filename Filename. Default: volcano.pdf, volcano.svg, volcano.png. If set to FALSE,
    there will be no plot.

ylab y axis text

xlab x axis text

title main title. If not specified it will be "Volcano plot (group1 vs group2)

legend Legend title
TCGAanalyze_DMC

color vector of colors to be used in graph
label vector of labels to be used in the figure. Example: c("Not Significant","Hypermethylated in group1", "Hypomethylated in group1")
xlim x limits to cut image
ylim y limits to cut image
p.cut p values threshold. Default: 0.01
probe.names is probe.names
cores Number of cores to be used in the non-parametric test Default = groupCol.group1.group2.rda
save Save object with results? Default: TRUE
save.directory Directory to save the files. Default: working directory
filename Name of the file to save the object.

Value

Volcano plot saved and the given data with the results (diffmean.group1.group2,p.value.group1.group2, p.value.adj.group1.group2,status.group1.group2) in the rowRanges where group1 and group2 are the names of the groups.

Examples

```r
nrows <- 200; ncols <- 20
counts <- matrix(
  runif(nrows * ncols, 1, 1e4), nrows,
  dimnames = list(paste0("cg",1:200),paste0("S",1:20))
)
rowRanges <- GenomicRanges::GRanges(
  rep(c("chr1", "chr2"), c(50, 150)),
  IRanges::IRanges(floor(runif(200, 1e5, 1e6)), width = 100),
  strand = sample(c("+", "-"), 200, TRUE),
  feature_id = sprintf("ID%03d", 1:200)
)
names(rowRanges) <- paste0("cg",1:200)
colData <- S4Vectors::DataFrame(
  Treatment = rep(c("ChIP", "Input"), 5),
  row.names = paste0("S",1:20),
  group = rep(c("group1","group2"),c(10,10))
)
data <- SummarizedExperiment::SummarizedExperiment(
  assays=S4Vectors::SimpleList(counts=counts),
  rowRanges = rowRanges,
  colData = colData
)
SummarizedExperiment::colData(data)$group <- c(rep("group 1",ncol(data)/2),
  rep("group 2",ncol(data)/2))
hypo.hyper <- TCGAanalyze_DMC(data, p.cut = 0.85,"group","group 1","group 2")
SummarizedExperiment::colData(data)$group2 <- c(rep("group_1",ncol(data)/2),
  rep("group_2",ncol(data)/2))
hypo.hyper <- TCGAanalyze_DMC(
  data = data,
  p.cut = 0.85,"group","group_1","group_2")
```

```r
p.cut = 0.85,
groupCol = "group2",
group1 = "group_1",
group2 = "group_2"
)
```

**Description**

The rationale behind an enrichment analysis (gene set, pathway etc) is to compute statistics of whether the overlap between the focus list (signature) and the gene set is significant, i.e., the confidence that the overlap between the list is not due to chance. The Gene Ontology project describes genes (gene products) using terms from three structured vocabularies: biological process, cellular component, and molecular function. The Gene Ontology Enrichment component, also referred to as the GO Terms component, allows the genes in any such "changed-gene" list to be characterized using the Gene Ontology terms annotated to them. It asks, whether for any particular GO term, the fraction of genes assigned to it in the "changed-gene" list is higher than expected by chance (is over-represented), relative to the fraction of genes assigned to that term in the reference set. In statistical terms it performs the analysis tests the null hypothesis that, for any particular ontology term, there is no difference in the proportion of genes annotated to it in the reference list and the proportion annotated to it in the test list. We adopted a Fisher Exact Test to perform the EA.

**Usage**

```r
TCGAanalyze_EA(  
  GeneName,  
  RegulonList,  
  TableEnrichment,  
  EAGenes,  
  GOtype,  
  FDRThresh = 0.01,  
  GeneSymbolsTable = FALSE  
)
```

**Arguments**

- **GeneName**
  is the name of gene signatures list
- **RegulonList**
  is a gene signature (list of genes) in which perform EA.
- **TableEnrichment**
  is a table related to annotations of gene symbols such as GO[BP,MF,CC] and Pathways. It was created from DAVID gene ontology on-line.
- **EAGenes**
  is a table with information about genes such as ID, Gene, Description, Location and Family.
**TCGAanalyze_EAcomplete**

**Enrichment analysis for Gene Ontology (GO) [BP, MF, CC] and Pathways**

**Description**

Researchers, in order to better understand the underlying biological processes, often want to retrieve a functional profile of a set of genes that might have an important role. This can be done by performing an enrichment analysis.

We will perform an enrichment analysis on gene sets using the TCGAanalyze_EAcomplete function. Given a set of genes that are up-regulated under certain conditions, an enrichment analysis will find identify classes of genes or proteins that are over-represented using annotations for that gene set.

**Usage**

```r
TCGAanalyze_EAcomplete(TFname, RegulonList)
```

**Arguments**

- `TFname` is the name of the list of genes or TF’s regulon.
- `RegulonList` List of genes such as TF’s regulon or DEGs where to find enrichment.

**Value**

Table with enriched GO or pathways by selected gene signature.

**Examples**

```r
## Not run:
EAGenes <- get("EAGenes")
RegulonList <- rownames(dataDEGsFiltLevel)
ResBP <- TCGAanalyze_EA(
  GeneName="DEA genes Normal Vs Tumor",
  RegulonList = RegulonList,
  TableEnrichment = DAVID_BP_matrix,
  EAGenes = EAGenes,
  GOtype = "DavidBP"
)
## End(Not run)
```
Value

Enrichment analysis GO[BP,MF,CC] and Pathways complete table enriched by genelist.

Examples

Genelist <- c("FN1","COL1A1")
ansEA <- TCGAanalyze_EAcomplete(TFname="DEA genes Normal Vs Tumor",Genelist)
## Not run:
Genelist <- rownames(dataDEGsFiltLevel)
system.time(ansEA <- TCGAanalyze_EAcomplete(TFname="DEA genes Normal Vs Tumor",Genelist))
## End(Not run)

TCGAanalyze_Filtering  Filtering mRNA transcripts and miRNA selecting a threshold.

Description

TCGAanalyze_Filtering allows user to filter mRNA transcripts and miRNA, samples, higher than the threshold defined quantile mean across all samples.

Usage

TCGAanalyze_Filtering(
    tabDF,           # is a dataframe or numeric matrix, each row represents a gene, each column represents a sample come from TCGAPrepare
    method,         # is method of filtering such as 'quantile', 'varFilter', 'filter1', 'filter2'
    qnt.cut = 0.25,  # is threshold selected as mean for filtering
    var.func = IQR,  # is function used as the per-feature filtering statistic. See genefilter documentation
    var.cutoff = 0.75,
    eta = 0.05,      # is a parameter for filter1. default eta = 0.05.
    foldChange = 1   # is a parameter for filter2. default foldChange = 1.
)
TCGAanalyze_LevelTab

Value
A filtered dataframe or numeric matrix where each row represents a gene, each column represents
a sample

Examples

dataNorm <- TCGAbiolinks::TCGAanalyze_Normalization(dataBRCA, geneInfo)
dataNorm <- TCGAanalyze_Normalization(tabDF = dataBRCA,
geneInfo = geneInfo,
method = "geneLength")
dataFilt <- TCGAanalyze_Filtering(tabDF = dataNorm, method = "quantile", qnt.cut = 0.25)

TCGAanalyze_LevelTab

Adding information related to DEGs genes from DEA as mean values in two conditions.

Description
TCGAanalyze_LevelTab allows user to add information related to DEGs genes from Differentially
expression analysis (DEA) such as mean values and in two conditions.

Usage

TCGAanalyze_LevelTab(
  FC_FDR_table_mRNA,
typeCond1,
typeCond2,
TableCond1,
TableCond2,
typeOrder = TRUE
)

Arguments

FC_FDR_table_mRNA Output of dataDEGs filter by abs(LogFC) >=1
typeCond1 a string containing the class label of the samples in TableCond1 (e.g., control
group)
typeCond2 a string containing the class label of the samples in TableCond2 (e.g., case
group)
TableCond1 numeric matrix, each row represents a gene, each column represents a sample
with Cond1type
TableCond2 numeric matrix, each row represents a gene, each column represents a sample
with Cond2type
typeOrder typeOrder
Value

Table with DEGs, log Fold Change (FC), false discovery rate (FDR), the gene expression level for samples in Cond1type, and Cond2type, and Delta value (the difference of gene expression between the two conditions multiplied logFC)

Examples

dataNorm <- TCGAbiolinks::TCGAanalyze_Normalization(dataBRCA, geneInfo)
dataFilt <- TCGAanalyze_Filtering(tabDF = dataBRCA, method = "quantile", qnt.cut = 0.25) samplesNT <- TCGAquery_SampleTypes(colnames(dataFilt), typesample = c("NT")) samplesTP <- TCGAquery_SampleTypes(colnames(dataFilt), typesample = c("TP")) dataDEGs <- TCGAanalyze_DEA(
  dataFilt[,samplesNT],
  dataFilt[,samplesTP],
  Cond1type = "Normal",
  Cond2type = "Tumor"
) dataDEGsFilt <- dataDEGs[abs(dataDEGs$logFC) >= 1,]
dataTP <- dataFilt[,samplesTP] dataTN <- dataFilt[,samplesNT] dataDEGsFiltLevel <- TCGAanalyze_LevelTab(
  FC_FDR_table_mRNA = dataDEGsFilt,
  typeCond1 = "Tumor",
  typeCond2 = "Normal",
  TableCond1 = dataTP,
  TableCond2 = dataTN
)

TCGAanalyze_networkInference

infer gene regulatory networks

Description

TCGAanalyze_networkInference taking expression data as input, this will return an adjacency matrix of interactions

Usage

TCGAanalyze_networkInference(data, optionMethod = "clr")

Arguments

data expression data, genes in columns, samples in rows
optionMethod inference method, chose from aracne, c3net, clr and mrnet

Value

an adjacent matrix
TCGAanalyze_Normalization

normalization mRNA transcripts and miRNA using EDASeq package.

Description

TCGAanalyze_Normalization allows user to normalize mRNA transcripts and miRNA, using EDASeq package.

Normalization for RNA-Seq Numerical and graphical summaries of RNA-Seq read data. Within-lane normalization procedures to adjust for GC-content effect (or other gene-level effects) on read counts: loess robust local regression, global-scaling, and full-quantile normalization (Risso et al., 2011). Between-lane normalization procedures to adjust for distributional differences between lanes (e.g., sequencing depth): global-scaling and full-quantile normalization (Bullard et al., 2010).

For instance returns all mRNA or miRNA with mean across all samples, higher than the threshold defined quantile mean across all samples.

TCGAanalyze_Normalization performs normalization using following functions from EDASeq

1. EDASeq::newSeqExpressionSet
2. EDASeq::withinLaneNormalization
3. EDASeq::betweenLaneNormalization
4. EDASeq::counts

Usage

TCGAanalyze_Normalization(tabDF, geneInfo, method = "geneLength")

Arguments

tabDF Rnaseq numeric matrix, each row represents a gene, each column represents a sample

geneInfo Information matrix of 20531 genes about geneLength and gcContent. Two objects are provided: TCGAbiolinks::geneInfoHT, TCGAbiolinks::geneInfo

method is method of normalization such as 'gcContent' or 'geneLength'

Value

Rnaseq matrix normalized with counts slot holds the count data as a matrix of non-negative integer count values, one row for each observational unit (gene or the like), and one column for each sample.

Examples

dataNorm <- TCGAbiolinks::TCGAanalyze_Normalization(dataBRCA, geneInfo)
**TCGAanalyze_Pathview**  
*Generate pathview graph*

**Description**

TCGAanalyze_Pathview pathway based data integration and visualization.

**Usage**

```r
TCGAanalyze_Pathview(dataDEGs, pathwayKEGG = "hsa05200")
```

**Arguments**

- `dataDEGs`  
- `pathwayKEGG`

**Value**

an adjacent matrix

**Examples**

```r
## Not run:
dataDEGs <- data.frame(mRNA = c("TP53","TP63","TP73"), logFC = c(1,2,3))
TCGAanalyze_Pathview(dataDEGs)
## End(Not run)
```

---

**TCGAanalyze_Preprocessing**  
*Array Array Intensity correlation (AAIC) and correlation boxplot to define outlier*

**Description**

TCGAanalyze_Preprocessing perform Array Array Intensity correlation (AAIC). It defines a square symmetric matrix of spearman correlation among samples. According this matrix and boxplot of correlation samples by samples it is possible to find samples with low correlation that can be identified as possible outliers.
Usage

TCGAanalyze_Preprocessing(
    object,
    cor.cut = 0,
    datatype = names(assays(object))[1],
    filename = NULL,
    width = 1000,
    height = 1000
)

Arguments

object gene expression of class RangedSummarizedExperiment from TCGAprepare
cor.cut is a threshold to filter samples according their spearman correlation in samples
    by samples. default cor.cut is 0
datatype is a string from RangedSummarizedExperiment assay
filename Filename of the image file
width Image width
height Image height

Value

Plot with array array intensity correlation and boxplot of correlation samples by samples

TCGAanalyze_Stemness Generate Stemness Score based on RNASeq (mRNAsi stemness index)
Malta et al., Cell, 2018

Description

TCGAanalyze_Stemness generate the mRNAsi score

Usage

TCGAanalyze_Stemness(stemSig, dataGE, colname.score = "stemness_score")

Arguments

stemSig is a vector of the stemness Signature generated using gelnet package. Please
    check the data from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5902191/
    • SC_PCBC_stemSig - Stemness Score
    • DE_PCBC_stemSig - endoderm score
    • EB_PCBC_stemSig - embryoid bodies score
    • ECTO_PCBC_stemSig - ectoderm score
    • MESO_PCBC_stemSig - mesoderm score
dataGE is a matrix of Gene expression (genes in rows, samples in cols) from TCGA prepare

colname.score Column name of the output. Default "stemness_score"

Value
table with samples and selected score

Examples

# Selecting TCGA breast cancer (10 samples) for example stored in dataBRCA
dataNorm <- TCGAanalyze_Normalization(
  tabDF = dataBRCA,
  geneInfo = geneInfo
)

# quantile filter of genes
dataFilt <- TCGAanalyze_Filtering(
  tabDF = dataNorm,
  method = "quantile",
  qnt.cut = 0.25
)

Stemness_score <- TCGAanalyze_Stemness(
  stemSig = SC_PCBC_stemSig,
  dataGE = dataFilt,
  colname.score = "SC_PCBC_stem_score"
)

ECTO_score <- TCGAanalyze_Stemness(
  stemSig = ECTO_PCBC_stemSig,
  dataGE = dataFilt,
  colname.score = "ECTO_PCBC_stem_score"
)

MESO_score <- TCGAanalyze_Stemness(
  stemSig = MESO_PCBC_stemSig,
  dataGE = dataFilt,
  colname.score = "MESO_PCBC_stem_score"
)

---

TCGAanalyze_survival Creates survival analysis

Description

Creates a survival plot from TCGA patient clinical data using survival library. It uses the fields days_to_death and vital, plus a columns for groups.
Usage

TCGAanalyze_survival(
  data,
  clusterCol = NULL,
  legend = "Legend",
  labels = NULL,
  risk.table = TRUE,
  xlim = NULL,
  main = "Kaplan-Meier Overall Survival Curves",
  ylab = "Probability of survival",
  xlab = "Time since diagnosis (days)",
  filename = "survival.pdf",
  color = NULL,
  height = 8,
  width = 12,
  dpi = 300,
  pvalue = TRUE,
  conf.int = TRUE,
  ...
)

Arguments

data             TCGA Clinical patient with the information days_to_death
clusterCol       Column with groups to plot. This is a mandatory field, the caption will be based in this column
legend           Legend title of the figure
labels           labels of the plot
risk.table       show or not the risk table
xlim             x axis limits e.g. xlim = c(0, 1000). Present narrower X axis, but not affect survival estimates.
main             main title of the plot
ylab             y axis text of the plot
xlab             x axis text of the plot
filename         The name of the pdf file.
color            Define the colors/Pallete for lines.
height           Image height
width            Image width
dpi              Figure quality
pvalue           show p-value of log-rank test
conf.int         show confidence intervals for point estimates of survival curves.
...              Further arguments passed to ggsurvplot.
Value

Survival plot

Examples

```r
# clin <- GDCquery_clinic("TCGA-BRCA","clinical")
clin <- data.frame(
  vital_status = c("alive","alive","alive","dead","alive",
  "alive","dead","alive","dead","alive"),
  days_to_death = c(NA,NA,172,NA,3472,NA,786,NA),
  days_to_last_follow_up = c(3011,965,718,NA,1914,423,NA,656,1417),
  gender = c(rep("male",5),rep("female",5))
)
TCGAanalyze_survival(clin, clusterCol="gender")
TCGAanalyze_survival(clin, clusterCol="gender", xlim = 1000)
TCGAanalyze_survival(clin,
  clusterCol="gender",
  risk.table = FALSE,
  conf.int = FALSE,
  color = c("pink","blue"))
TCGAanalyze_survival(clin,
  clusterCol="gender",
  risk.table = FALSE,
  xlim = c(100,1000),
  conf.int = FALSE,
  color = c("Dark2"))
```

---

**TCGAanalyze_SurvivalKM**

*Survival analysis (SA) univariate with Kaplan-Meier (KM) method.*

---

**Description**

TCGAanalyze_SurvivalKM perform an univariate Kaplan-Meier (KM) survival analysis (SA). It performed Kaplan-Meier survival univariate using complete follow up with all days taking one gene a time from Genelist of gene symbols. For each gene according its level of mean expression in cancer samples, defining two thresholds for quantile expression of that gene in all samples (default ThreshTop=0.67, ThreshDown=0.33) it is possible to define a threshold of intensity of gene expression to divide the samples in 3 groups (High, intermediate, low). TCGAanalyze_SurvivalKM performs SA between High and low groups using following functions from survival package

1. survival::Surv
2. survival::survdiff
3. survival::survfit
Usage

TCGAanalyze_SurvivalKM(
  clinical_patient,
  dataGE,
  Genelist,
  Survresult = FALSE,
  ThreshTop = 0.67,
  ThreshDown = 0.33,
  p.cut = 0.05,
  group1,
  group2
)

Arguments

clinical_patient
  is a data.frame using function 'clinic' with information related to barcode / samples such as bcr_patient_barcode, days_to_death, days_to_last_follow_up, vital_status, etc

dataGE
  is a matrix of Gene expression (genes in rows, samples in cols) from TCGAPrepare

Genelist
  is a list of gene symbols where perform survival KM.

Survresult
  is a parameter (default = FALSE) if is TRUE will show KM plot and results.

ThreshTop
  is a quantile threshold to identify samples with high expression of a gene

ThreshDown
  is a quantile threshold to identify samples with low expression of a gene

p.cut
  p.values threshold. Default: 0.05

group1
  a string containing the barcode list of the samples in in control group

group2
  a string containing the barcode list of the samples in in disease group

Value

table with survival genes pvalues from KM.

Examples

# Selecting only 20 genes for example
dataBRCAcomplete <- log2(dataBRCA[1:20,] + 1)

# clinical_patient_Cancer <- GDCquery_clinic("TCGA-BRCA","clinical")
clinical_patient_Cancer <- data.frame(  
bcr_patient_barcode = substr(colnames(dataBRCAcomplete),1,12),  
vital_status = c(rep("alive",3),"dead",rep("alive",2),rep(c("dead","alive"),2)),  
days_to_death = c(NA,NA,NA,172,NA,NA,3472,NA,786,NA),  
days_to_last_follow_up = c(3011,965,718,NA,1914,423,NA,5,656,1417)
)

group1 <- TCGAquery_SampleTypes(colnames(dataBRCAcomplete), typesample = c("NT"))

group2 <- TCGAquery_SampleTypes(colnames(dataBRCAcomplete), typesample = c("TP"))

tabSurvKM <- TCGAanalyze_SurvivalKM(
  clinical_patient = clinical_patient_Cancer,
  dataGE = dataBRCAcomplete,
  Genelist = rownames(dataBRCAcomplete),
  Survresult = FALSE,
  p.cut = 0.4,
  ThreshTop = 0.67,
  ThreshDown = 0.33,
  group1 = group1, # Control group
  group2 = group2
) # Disease group

# If the groups are not specified group1 == group2 and all samples are used
## Not run:
# tabSurvKM <- TCGAanalyze_SurvivalKM(
#   clinical_patient_Cancer,
#   dataBRCAcomplete,
#   Genelist = rownames(dataBRCAcomplete),
#   Survresult = TRUE,
#   p.cut = 0.2,
#   ThreshTop = 0.67,
#   ThreshDown = 0.33
# )

## End(Not run)

TCGAbatch_Correction

Batch correction using ComBat and Voom transformation using limma package.

Description

TCGAbatch_correction allows user to perform a Voom correction on gene expression data and have it ready for DEA. One can also use ComBat for batch correction for exploratory analysis. If batch.factor or adjustment argument is "Year" please provide clinical data. If no batch factor is provided, the data will be voom corrected only.

TCGAanalyze_DEA performs DEA using following functions from sva and limma:

1. limma::voom Transform RNA-Seq Data Ready for Linear Modelling.
2. sva::ComBat Adjust for batch effects using an empirical Bayes framework.

Usage

TCGAbatch_Correction(
  tabDF,
  batch.factor = NULL,
  adjustment = NULL,
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tabDF</td>
<td>numeric matrix, each row represents a gene, each column represents a sample</td>
</tr>
<tr>
<td>batch.factor</td>
<td>a string containing the batch factor to use for correction. Options are &quot;Plate&quot;, &quot;TSS&quot;, &quot;Year&quot;, &quot;Portion&quot;, &quot;Center&quot;</td>
</tr>
<tr>
<td>adjustment</td>
<td>vector containing strings for factors to adjust for using ComBat. Options are &quot;Plate&quot;, &quot;TSS&quot;, &quot;Year&quot;, &quot;Portion&quot;, &quot;Center&quot;</td>
</tr>
<tr>
<td>ClinicalDF</td>
<td>a dataframe returned by GDCquery_clinic() to be used to extract year data</td>
</tr>
<tr>
<td>UnpublishedData</td>
<td>if TRUE perform a batch correction after adding new data</td>
</tr>
<tr>
<td>AnnotationDF</td>
<td>a dataframe with column Batch indicating different batches of the samples in the tabDF</td>
</tr>
</tbody>
</table>

Value
data frame with ComBat batch correction applied

---

**TCGAprepare_Affy** Prepare CEL files into an AffyBatch.

Description

Prepare CEL files into an AffyBatch.

Usage

TCGAprepare_Affy(ClinData, PathFolder, TabCel)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClinData</td>
<td>write</td>
</tr>
<tr>
<td>PathFolder</td>
<td>write</td>
</tr>
<tr>
<td>TabCel</td>
<td>write</td>
</tr>
</tbody>
</table>

Value

Normalized Expression data from Affy eSets
TCGAquery_MatchedCoupledSampleTypes

Retrieve multiple tissue types from the same patients.

Description

TCGAquery_MatchedCoupledSampleTypes

Usage

TCGAquery_MatchedCoupledSampleTypes(barcode, typeSample)

Arguments

barcode: barcode

typeSample: typesample

Value

a list of samples / barcode filtered by type sample selected

Examples

TCGAquery_MatchedCoupledSampleTypes(c("TCGA-B0-4698-01Z-00-DX1", "TCGA-B0-4698-02Z-00-DX1"), c("TP", "TR"))


TCGAquery_MatchedCoupledSampleTypes(barcode,c("TR", "TP"))
TCGAquery_recount2  
Query gene counts of TCGA and GTEx data from the Recount2 project

Description

TCGArecount2_query queries and downloads data produced by the Recount2 project. User can specify which project and which tissue to query

Usage

TCGAquery_recount2(project, tissue = c())

Arguments

project is a string denoting which project the user wants. Options are "tcga" and "gtex"

Value

List with $subtypes attribute as a dataframe with barcodes, samples, subtypes, and colors. The $filtered attribute is returned as filtered samples with no subtype info

Examples

## Not run:
brain.rec<-TCGAquery_recount2(project = "gtex", tissue = "brain")

## End(Not run)

TCGAquery_SampleTypes  Retrieve multiple tissue types not from the same patients.

Description

TCGAquery_SampleTypes for a given list of samples and types, return the union of samples that are from these type.

Usage

TCGAquery_SampleTypes(barcode, typesample)
Arguments

barcode is a list of samples as TCGA barcodes

typesample a character vector indicating tissue type to query. Example:

TP  PRIMARY SOLID TUMOR
TR  RECURRENT SOLID TUMOR
TB Primary Blood Derived Cancer-Peripheral Blood
TRBM Recurrent Blood Derived Cancer-Bone Marrow
TAP Additional-New Primary
TM Metastatic
TAM Additional Metastatic
THOC Human Tumor Original Cells
TBM Primary Blood Derived Cancer-Bone Marrow
NB Blood Derived Normal
NT Solid Tissue Normal
NBC Buccal Cell Normal
NEBV EBV Immortalized Normal
NBM Bone Marrow Normal

Value

a list of samples / barcode filtered by type sample selected

Examples

# selection of normal samples "NT"
barcode <- c("TCGA-B0-4698-01Z-00-DX1","TCGA-CZ-4863-02Z-00-DX1")
# Returns the second barcode
TCGAquery_SampleTypes(barcode,"TR")
# Returns both barcode
TCGAquery_SampleTypes(barcode,c("TR","TP"))

TCGAquery_subtype Retrieve molecular subtypes for a given tumor

Description

TCGAquery_subtype Retrieve molecular subtypes for a given tumor

Usage

TCGAquery_subtype(tumor)
TCGAtumor_purity

Arguments

tumor is a cancer Examples:

  lgg  gbm  luad stad  brca
coad  read

Value

a data.frame with barcode and molecular subtypes

Examples

dataSubt <- TCGAquery_subtype(tumor = "lgg")

TCGAtumor_purity
Filters TCGA barcodes according to purity parameters

Description

TCGAtumor_purity Filters TCGA samples using 5 estimates from 5 methods as thresholds.

Usage

TCGAtumor_purity(barcodes, estimate, absolute, lump, ihc, cpe)

Arguments

barcodes is a vector of TCGA barcodes
estimate uses gene expression profiles of 141 immune genes and 141 stromal genes
absolute which uses somatic copy-number data (estimations were available for only 11 cancer types)
lump (leukocytes unmethylation for purity), which averages 44 non-methylated immune-specific CpG sites
ihc as estimated by image analysis of haematoxylin and eosin stain slides produced by the Nationwide Childrens Hospital Biospecimen Core Resource
cpe CPE is a derived consensus measurement as the median purity level after normalizing levels from all methods to give them equal means and s.ds

Value

List with $pure_barcodes attribute as a vector of pure samples and $filtered attribute as filtered samples with no purity info
Examples

```r
dataTableSubt <- TCGAtumor_purity("TCGA-60-2721-01A-01R-0851-07",
   estimate = 0.6,
   absolute = 0.6,
   ihc = 0.8,
   lump = 0.8,
   cpe = 0.7)
```

Description

Barplot of subtypes and clinical info in groups of gene expression clustered.

Usage

```r
TCGAvisualize_BarPlot(
   DFfilt, DFclin, DFsubt, data_Hc2, Subtype, cbPalette, filename, width, height, dpi
)
```

Arguments

- `DFfilt`: write
- `DFclin`: write
- `DFsubt`: write
- `data_Hc2`: write
- `Subtype`: write
- `cbPalette`: Define the colors of the bar.
- `filename`: The name of the pdf file
- `width`: Image width
- `height`: Image height
- `dpi`: Image dpi

Value

barplot image in pdf or png file
**Description**

The figure shows canonical pathways significantly overrepresented (enriched) by the DEGs (differentially expressed genes). The most statistically significant canonical pathways identified in DEGs list are listed according to their p value corrected FDR (-Log) (colored bars) and the ratio of list genes found in each pathway over the total number of genes in that pathway (Ratio, red line).

**Usage**

```r
TCGAvisualize_Eabarplot(
  tf, 
  GOMFTab, 
  GOBPTab, 
  GOCCTab, 
  PathTab, 
  nBar, 
  nRGTab, 
  filename = "TCGAvisualize_Eabarplot_Output.pdf", 
  text.size = 1, 
  mfrow = c(2, 2), 
  xlim = NULL, 
  fig.width = 30, 
  fig.height = 15, 
  color = c("orange", "cyan", "green", "yellow")
)
```

**Arguments**

- **tf** is a list of gene symbols
- **GOMFTab** is results from TCGAanalyze_EAcomplete related to Molecular Function (MF)
- **GOBPTab** is results from TCGAanalyze_EAcomplete related to Biological Process (BP)
- **GOCCTab** is results from TCGAanalyze_EAcomplete related to Cellular Component (CC)
- **PathTab** is results from TCGAanalyze_EAcomplete related to Pathways EA
- **nBar** is the number of bar histogram selected to show (default = 10)
- **nRGTab** is the gene signature list with gene symbols.
- **filename** Name for the pdf. If null it will return the plot.
- **text.size** Text size
- **mfrow** Vector with number of rows/columns of the plot. Default 2 rows/2 columns "c(2,2)"
- **xlim** Upper limit of the x-axis.
**fig.width**  Default 30
**fig.height** Default 15
**color** A vector of colors for each barplot. Default: c("orange", "cyan", "green", "yellow")

**Value**
Complete barPlot from Enrichment Analysis showing significant (default FDR < 0.01) BP, CC, MF and pathways enriched by list of genes.

**Examples**

```r
Genelist <- c("FN1", "COL1A1")
ansEA <- TCGAanalyze_EAcomplete(TFname="DEA genes Normal Vs Tumor", Genelist)
TCGAvisualize_EAbarplot(tf = rownames(ansEA$ResBP),
                       GOBPTab = ansEA$ResBP,
                       GOCCTab = ansEA$ResCC,
                       GOMFTab = ansEA$ResMF,
                       PathTab = ansEA$ResPat,
                       nRGTab = Genelist,
                       nBar = 10,
                       filename="a.pdf")
```

```
## Not run:
Genelist <- rownames(dataDEGsFiltLevel)
system.time(ansEA <- TCGAanalyze_EAcomplete(TFname="DEA genes Normal Vs Tumor", Genelist))
# Enrichment Analysis EA (TCGAVisualize)
# Gene Ontology (GO) and Pathway enrichment barPlot
TCGAvisualize_EAbarplot(tf = rownames(ansEA$ResBP),
                       GOBPTab = ansEA$ResBP,
                       GOCCTab = ansEA$ResCC,
                       GOMFTab = ansEA$ResMF,
                       PathTab = ansEA$ResPat,
                       nRGTab = Genelist,
                       nBar = 10)
```

## End(Not run)

---

**TCGAvisualize_Heatmap**

Heatmap with more sensible behavior using heatmap.plus

**Description**
Heatmap with more sensible behavior using heatmap.plus

**Usage**

```r
TCGAvisualize_Heatmap(
  data,
  col.metadata,
  row.metadata,
  data, col.metadata, row.metadata,
)```


```r

# arguments

col.colors = NULL,
row.colors = NULL,
show_column_names = FALSE,
show_row_names = FALSE,
cluster_rows = FALSE,
cluster_columns = FALSE,
sortCol,
extremes = NULL,
rownames.size = 12,
title = NULL,
color.levels = NULL,
values.label = NULL,
filename = "heatmap.pdf",
width = 10,
height = 10,
type = "expression",
scale = "none",
heatmap.legend.color.bar = "continuous"
)

Arguments

data The object to with the heatmap data (expression, methylation)
col.metadata Metadata for the columns (samples). It should have one of the following columns:
barcode (28 characters) column to match with the samples. It will also work with
"bcr_patient_barcode"(12 chars),"patient"(12 chars),"sample"(16 chars) columns
but as one patient might have more than one sample, this could lead to errors in
the annotation. The code will throw a warning in case two samples are from the
same patient.
row.metadata Metadata for the rows genes (expression) or probes (methylation)
col.colors A list of names colors
row.colors A list of named colors
show_column_names Show column names names? Default: FALSE
show_row_names Show row names? Default: FALSE
cluster_rows Cluster rows ? Default: FALSE
cluster_columns Cluster columns ? Default: FALSE
sortCol Name of the column to be used to sort the columns
extremes Extremes of colors (vector of 3 values)
rownames.size Rownames size
title Title of the plot
color.levels A vector with the colors (low level, middle level, high level)
values.label Text of the levels in the heatmap
filename Filename to save the heatmap. Default: heatmap.png
```
width | figure width
---|---
height | figure height
**type** | Select the colors of the heatmap values. Possible values are "expression" (default), "methylation"
**scale** | Use z-score to make the heatmap? If we want to show differences between genes, it is good to make Z-score by samples (force each sample to have zero mean and standard deviation=1). If we want to show differences between samples, it is good to make Z-score by genes (force each gene to have zero mean and standard deviation=1). Possibilities: "row", "col". Default "none"
**heatmap.legend.color.bar** | Heatmap legends values type. Options: "continuous", "discrete"

**Value**

Heatmap plotted in the device

**Examples**

```r
row.mdat <- matrix(c("FALSE","FALSE",
 "TRUE","TRUE",
 "FALSE","FALSE",
 "TRUE","FALSE",
 "FALSE","TRUE"),
 nrow = 5, ncol = 2, byrow = TRUE,
 dimnames = list(
   c("probe1", "probe2","probe3","probe4","probe5"),
   c("duplicated", "Enhancer region")))

dat <- matrix(c(0.3,0.2,0.3,1,1,0.1,1,1,0, 0.8,1,0.7,0.7,0.3,1),
 nrow = 5, ncol = 3, byrow = TRUE,
 dimnames = list(
   c("probe1", "probe2","probe3","probe4","probe5"),
   c("TCGA-DU-6410", "TCGA-DU-AST5", "TCGA-HT-7688")))

mdat <- data.frame(patient=c("TCGA-DU-6410","TCGA-DU-AST5","TCGA-HT-7688"),
 Sex=c("Male","Female","Male"),
 COCCluster=c("coc1","coc1","coc1"),
 IDHtype=c("IDHwt","IDHMut-cod","IDHMut-noncod"))

TCGAvisualize_Heatmap(dat,
 col.metadata = mdat,
 row.metadata = row.mdat,
 row.colors = list(duplicated = c("FALSE" = "pink",
 "TRUE"="green"),
 "Enhancer region" = c("FALSE" = "purple",
 "TRUE"="grey")),
 col.colors = list(Sex = c("Male" = "blue", "Female"="red"),
 COCCluster=c("coc1"="grey"),
 IDHtype=c("IDHwt"="cyan"),
...)
```
TCGAvisualize_meanMethylation

Mean methylation boxplot

Description

Creates a mean methylation boxplot for groups (groupCol), subgroups will be highlighted as shapes if the subgroupCol was set.

Observation: Data is a summarizedExperiment.

Usage

TCGAvisualize_meanMethylation(
  data,
  groupCol = NULL,
  subgroupCol = NULL,
  shapes = NULL,
  print.pvalue = FALSE,
  plot.jitter = TRUE,
  jitter.size = 3,
  filename = "groupMeanMet.pdf",
  ylab = expression(paste("Mean DNA methylation (", beta, "-values")
)),
  xlab = NULL,
  title = "Mean DNA methylation",
  labels = NULL,
  group.legend = NULL,
  subgroup.legend = NULL,
  color = NULL,
  y.limits = NULL,
  sort,
  order,
  legend.position = "top",
  legend.title.position = "top",
  legend.ncols = 3,
  add.axis.x.text = TRUE,
  width = 10,
  height = 10,
  dpi = 600,
  axis.text.x.angle = 90
)
**Arguments**

- `data` : SummarizedExperiment object obtained from TCGAPrepare
- `groupCol` : Columns in colData(data) that defines the groups. If no columns defined a columns called "Patients" will be used
- `subgroupCol` : Columns in colData(data) that defines the subgroups.
- `shapes` : Shape vector of the subgroups. It must have the size of the levels of the subgroups. Example: shapes = c(21,23) if for two levels
- `print.pvalue` : Print p-value for two groups
- `plot.jitter` : Plot jitter? Default TRUE
- `jitter.size` : Plot jitter size? Default 3
- `filename` : The name of the pdf that will be saved
- `ylab` : y axis text in the plot
- `xlab` : x axis text in the plot
- title : main title in the plot
- `labels` : Labels of the groups
- `group.legend` : Name of the group legend. DEFAULT: groupCol
- `subgroup.legend` : Name of the subgroup legend. DEFAULT: subgroupCol

- `color` : vector of colors to be used in graph
- `y.limits` : Change lower/upper y-axis limit
- `sort` : Sort boxplot by mean or median. Possible values: mean.asc, mean.desc, median.asc, median.desc
- `order` : Order of the boxplots
- `legend.position` : Legend position ("top", "right", "left", "bottom")
- `legend.title.position` : Legend title position ("top", "right", "left", "bottom")
- `legend.ncols` : Number of columns of the legend
- `add.axis.x.text` : Add text to x-axis? Default: FALSE
- `width` : Plot width default:10
- `height` : Plot height default:10
- `dpi` : Pdf dpi default:600
- `axis.text.x.angle` : Angle of text in the x axis

**Value**

- Save the pdf survival plot
Examples

```r
nrows <- 200; ncols <- 21
counts <- matrix(runif(nrows * ncols, 0, 1), nrows)
rowRanges <- GenomicRanges::GRanges(rep(c("chr1", "chr2"), c(50, 150)),
    IRanges::IRanges(floor(runif(200, 1e5, 1e6)), width=100),
    strand=sample(c("+", "-"), 200, TRUE),
    feature_id=sprintf("ID%03d", 1:200))
colData <- S4Vectors::DataFrame(Treatment=rep(c("ChIP", "Input","Other"), 7),
    row.names=LETTERS[1:21],
    group=rep(c("group1","group2","group3"),c(7,7,7)),
    subgroup=rep(c("subgroup1","subgroup2","subgroup3"),7))
data <- SummarizedExperiment::SummarizedExperiment(
    assays=S4Vectors::SimpleList(counts=counts),
    rowRanges=rowRanges,
    colData=colData)
TCGAvisualize_meanMethylation(data,groupCol = "group")
# change lower/upper y-axis limit
TCGAvisualize_meanMethylation(data,groupCol = "group", y.limits = c(0,1))
# change lower y-axis limit
TCGAvisualize_meanMethylation(data,groupCol = "group", y.limits = 0)
TCGAvisualize_meanMethylation(data,groupCol = "group", subgroupCol="subgroup")
TCGAvisualize_meanMethylation(data,groupCol = "group")
TCGAvisualize_meanMethylation(data,groupCol = "group",sort="mean.desc",filename="meandesc.pdf")
TCGAvisualize_meanMethylation(data,groupCol = "group",sort="mean.asc",filename="meanasc.pdf")
TCGAvisualize_meanMethylation(data,groupCol = "group",sort="median.asc",filename="medianasc.pdf")
TCGAvisualize_meanMethylation(data,groupCol = "group",sort="median.desc",filename="mediandesc.pdf")
```

Description

Creating a oncoprint

Usage

```r
TCGAvisualize_oncoprint(
    mut,
    genes,
    filename,
    color,
    annotation.position = "bottom",
    annotation,
    height,
    width = 10,
    rm.empty.columns = FALSE,
    show.column.names = FALSE,
)```
show.row.barplot = TRUE,
label.title = "Mutation",
column.names.size = 8,
label.font.size = 16,
rows.font.size = 16,
dist.col = 0.5,
dist.row = 0.5,
information = "Variant_Type",
row.order = TRUE,
col.order = TRUE,
heatmap.legend.side = "bottom",
annotation.legend.side = "bottom"
)

Arguments

mut A dataframe from the mutation annotation file (see TCGAquery_maf from TCGAbiolinks)

genes Gene list

filename name of the pdf

color named vector for the plot

annotation.position Position of the annotation "bottom" or "top"

annotation Matrix or data frame with the annotation. Should have a column bcr_patient_barcode with the same ID of the mutation object

height pdf height

width pdf width

rm.empty.columns If there is no alteration in that sample, whether remove it on the oncoprint

show.column.names Show column names? Default: FALSE

show.row.barplot Show barplot annotation on rows?

label.title Title of the label

column.names.size Size of the fonts of the columns names

label.font.size Size of the fonts

rows.font.size Size of the fonts

dist.col distance between columns in the plot

dist.row distance between rows in the plot

information Which column to use as information from MAF. Options: 1) "Variant_Classification" (The information will be "Frame_Shift_Del", "Frame_Shift_Ins", "In_Frame_Del", "In_Frame_Ins", "Missense_Mutation", "Nonsense_Mutation", "Nonstop_Mutation", "RNA", "Silent", "Splice_Site", "Targeted_Region", "Translation_Start_Site")

2) "Variant_Type" (The information will be INS, DEL, SNP)
row.order Order the genes (rows) Default: TRUE. Genes with more mutations will be in the first rows

col.order Order columns. Default: TRUE.

heatmap.legend.side Position of the heatmap legend

annotation.legend.side Position of the annotation legend

Value

A oncoprint plot

Examples

```r
## Not run:
library(dplyr)
query <- GDCquery(
  project = "TCGA-CHOL",
data.category = "Simple Nucleotide Variation",
  access = "open",
  legacy = FALSE,
data.type = "Masked Somatic Mutation",
  workflow.type = "Aliquot Ensemble Somatic Variant Merging and Masking"
)
GDCdownload(query)
mut <- GDCprepare(query)
TCGAvisualize_oncoprint(mut = mut, genes = mut$Hugo_Symbol[1:10], rm.empty.columns = TRUE)
TCGAvisualize_oncoprint(
  mut = mut, genes = mut$Hugo_Symbol[1:10],
  filename = "onco.pdf",
  color = c("background"="#CCCCCC","DEL"="purple","INS"="yellow","SNP"="brown")
)
clin <- GDCquery_clinic("TCGA-ACC","clinical")
clin <- clin[,c("bcr_patient_barcode","disease","gender","tumor_stage","race","vital_status")]
TCGAvisualize_oncoprint(
  mut = mut, genes = mut$Hugo_Symbol[1:20],
  filename = "onco.pdf",
  annotation = clin,
  color=c("background"="#CCCCCC","DEL"="purple","INS"="yellow","SNP"="brown"),
  rows.font.size=10,
  heatmap.legend.side = "right",
  dist.col = 0,
  label.font.size = 10
)
## End(Not run)
```
TCGAvisualize_PCA

Principal components analysis (PCA) plot

Description

TCGAvisualize_PCA performs a principal components analysis (PCA) on the given data matrix and returns the results as an object of class prcomp, and shows results in PCA level.

Usage

TCGAvisualize_PCA(dataFilt, dataDEGsFiltLevel, ntopgenes, group1, group2)

Arguments

dataFilt A filtered dataframe or numeric matrix where each row represents a gene, each column represents a sample from function TCGAanalyze_Filtering
dataDEGsFiltLevel table with DEGs, log Fold Change (FC), false discovery rate (FDR), the gene expression level, etc, from function TCGAanalyze_LevelTab.
ntopgenes number of DEGs genes to plot in PCA
group1 a string containing the barcode list of the samples in in control group
group2 a string containing the barcode list of the samples in in disease group the name of the group

Value

principal components analysis (PCA) plot of PC1 and PC2

Examples

# normalization of genes
dataNorm <- TCGAbiolinks::TCGAanalyze_Normalization(tabDF = dataBRCA, geneInfo = geneInfo, method = "geneLength")
# quantile filter of genes
dataFilt <- TCGAanalyze_Filtering(tabDF = dataBRCA, method = "quantile", qnt.cut = 0.25)
# Principal Component Analysis plot for ntop selected DEGs
# selection of normal samples "NT"
group1 <- TCGAquery_SampleTypes(colnames(dataFilt), typesample = c("NT"))
# selection of normal samples "TP"
group2 <- TCGAquery_SampleTypes(colnames(dataFilt), typesample = c("TP"))
pca <- TCGAvisualize_PCA(dataFilt, dataDEGsFiltLevel, ntopgenes = 200, group1, group2)
Description

Create Starburst plot for comparison of DNA methylation and gene expression. The log10 (FDR-corrected P value) is plotted for beta value for DNA methylation (x axis) and gene expression (y axis) for each gene.

The black dashed line shows the FDR-adjusted P value of 0.01.

You can set names to TRUE to get the names of the significant genes.

Candidate biologically significant genes will be circled in the plot.

Candidate biologically significant are the genes that respect the expression (logFC.cut), DNA methylation (diffmean.cut) and significance thresholds (exp.p.cut, met.p.cut)

Usage

TCGAvisualize_starburst(
  met,
  exp,
  group1 = NULL,
  group2 = NULL,
  exp.p.cut = 0.01,
  met.p.cut = 0.01,
  diffmean.cut = 0,
  logFC.cut = 0,
  met.platform = c("Illumina Human Methylation 450", "Illumina Human Methylation 27", "Illumina Methylation Epic"),
  genome,
  names = FALSE,
  names.fill = TRUE,
  filename = "starburst.png",
  return.plot = FALSE,
  ylab = expression(atop("Gene Expression", paste(-Log[10], " (FDR corrected P values)"))),
  xlab = expression(atop("DNA Methylation", paste(-Log[10], " (FDR corrected P values)"))),
  title = "Starburst Plot",
  legend = "DNA Methylation/Expression Relation",
  color = NULL,
  xlim = NULL,
Arguments

**met**  A SummarizedExperiment with methylation data obtained from the TCGAPrepare or Data frame from DMR_results file. Expected colData columns: diffmean, p.value.adj and p.value Execute volcanoPlot function in order to obtain these values for the object.

**exp**  Object obtained by DEArnaSEQ function

**group1**  The name of the group 1 Obs: Column p.value.adj.group1.group2 should exist

**group2**  The name of the group 2. Obs: Column p.value.adj.group1.group2 should exist

**exp.p.cut**  expression p value cut-off

**met.p.cut**  methylation p value cut-off

**diffmean.cut**  If set, the probes with diffmean higher than methylation cut-off will be highlighted in the plot. And the data frame return will be subseted.

**logFC.cut**  If set, the probes with expression fold change higher than methylation cut-off will be highlighted in the plot. And the data frame return will be subseted.

**met.platform**  DNA methylation platform "Illumina Human Methylation 450", "Illumina Human Methylation 27", "Illumina Methylation Epic"

**genome**  Genome of reference ("hg38" or "hg19") used to identify nearest probes TSS

**names**  Add the names of the significant genes? Default: FALSE

**names.fill**  Names should be filled in a color box? Default: TRUE

**filename**  The filename of the file (it can be pdf, svg, png, etc)

**return.plot**  If true only plot object will be returned (pdf will not be created)

**ylab**  y axis text

**xlab**  x axis text

**title**  main title

**legend**  legend title

**color**  vector of colors to be used in graph

**label**  vector of labels to be used in graph

**xlim**  x limits to cut image

**ylim**  y limits to cut image

**height**  Figure height

**width**  Figure width

**dpi**  Figure dpi

Details

Input: data with gene expression/methylation expression Output: starburst plot
**Value**

Save a starburst plot

**Examples**

```r
## Not run: library(SummarizedExperiment) met <- TCGAbiolinks:::getMetPlatInfo( genome = "hg38", platform = "Illumina Human Methylation 27") values(met) <- NULL met$probeID <- names(met) nrows <- length(met); ncols <- 20 counts <- matrix(runif(nrows * ncols, 1, 1e4), nrows) colData <- S4Vectors::DataFrame( Treatment = rep(c("ChIP", "Input"), 5), row.names = LETTERS[1:20], group = rep(c("group1","group2"),c(10,10)) ) met <- SummarizedExperiment::SummarizedExperiment( assays = S4Vectors::SimpleList(counts=counts), rowRanges = met, colData = colData ) rowRanges(met)$diffmean.g1.g2 <- c(runif(nrows, -0.1, 0.1)) rowRanges(met)$diffmean.g2.g1 <- -1*(rowRanges(met)$diffmean.g1.g2) rowRanges(met)$p.value.g1.g2 <- c(runif(nrows, 0, 1)) rowRanges(met)$p.value.adj.g1.g2 <- c(runif(nrows, 0, 1)) exp <- TCGAbiolinks:::get.GRCh.bioMart("hg38") exp$logFC <- runif(nrow(exp), -5, 5) exp$FDR <- runif(nrow(exp), 0.01, 1) result <- TCGAvisualize_starburst( met, exp, exp.p.cut = 0.05, met.p.cut = 0.05, logFC.cut = 2, group1 = "g1", group2 = "g2", genome = "hg38", met.platform = "27k", diffmean.cut = 0.0, names = TRUE )
```
TCGAvizualize_SurvivalCoxNET

Survival analysis with univariate Cox regression package (dnet)

Description

TCGAvizualize_SurvivalCoxNET can help an user to identify a group of survival genes that are significant from univariate Kaplan Meier Analysis and also for Cox Regression. It shows in the end a network built with community of genes with similar range of p-values from Cox regression (same color) and the interaction among those genes is already validated in literatures using the STRING database (version 9.1). TCGAvizualize_SurvivalCoxNET perform survival analysis with univariate Cox regression and package (dnet) using following functions wrapping from these packages:

1. survival::coxph
2. igraph::subgraph.edges
3. igraph::layout.fruchterman.reingold
4. igraph::spinglass.community
5. igraph::communities
6. dnet::dRDataLoader
7. dnet::dNetInduce
8. dnet::dNetPipeline
9. dnet::visNet
10. dnet::dCommSignif

Usage

TCGAvizualize_SurvivalCoxNET(
  clinical_patient,
  dataGE,
  Genelist,
  org.Hs.string,
  scoreConfidence = 700,
  titlePlot = "TCGAvizualize_SurvivalCoxNET Example"
)

Arguments

clincial_patient

is a data.frame using function 'clinic' with information related to barcode / samples such as bcr_patient_barcode, days_to_death, days_to_last_followup, vital_status, etc

dataGE

is a matrix of Gene expression (genes in rows, samples in cols) from TCGAprepare

Genelist

is a list of gene symbols where perform survival KM.
\texttt{org.Hs.string} is an igraph object that contains a functional protein association network in human. The network is extracted from the STRING database (version 10).

\texttt{scoreConfidence} restrict to those edges with high confidence (eg. score>=700)

\texttt{titlePlot} is the title to show in the final plot.

\textbf{Details}

\texttt{TCGAvisualize_SurvivalCoxNET} allow user to perform the complete workflow using \texttt{coxph} and \texttt{dnet} package related to survival analysis with an identification of gene-active networks from high-throughput omics data using gene expression and clinical data.

1. Cox regression survival analysis to obtain hazard ratio (HR) and p-values
2. fit a Cox proportional hazards model and ANOVA (Chisq test)
3. Network communities
4. An igraph object that contains a functional protein association network in human. The network is extracted from the STRING database (version 9.1). Only those associations with medium confidence (score>=400) are retained.
5. restrict to those edges with high confidence (score>=700)
6. extract network that only contains genes in pvals
7. Identification of gene-active network
8. visualisation of the gene-active network itself
9. the layout of the network visualisation (fixed in different visuals)
10. color nodes according to communities (identified via a spin-glass model and simulated annealing)
11. node sizes according to degrees
12. highlight different communities
13. visualize the subnetwork

\textbf{Value}

\texttt{net IGRAPH} with related Cox survival genes in community (same pval and color) and with interactions from STRING database.

---

\texttt{TCGAvisualize_volcano}  \textit{Creates a volcano plot for DNA methylation or gene expression}

\textbf{Description}

Creates a volcano plot from the gene expression and DNA methylation analysis.
Usage

TCGAVisualize_volcano(
  x,
  y,
  filename = "volcano.pdf",
  ylab = expression(paste(-Log[10], " (FDR corrected P-values)")),
  xlab = NULL,
  title = "Volcano plot",
  legend = NULL,
  label = NULL,
  xlim = NULL,
  ylim = NULL,
  color = c("black", "red", "green"),
  names = NULL,
  names.fill = TRUE,
  show.names = "significant",
  x.cut = 0,
  y.cut = 0.01,
  height = 5,
  width = 10,
  highlight = NULL,
  highlight.color = "orange",
  names.size = 4,
  dpi = 300
)

Arguments

x  x-axis data (i.e. Diff mean beta-values or Log2FC).
y  FDR adjusted p-value (q-value). This data will be transformed to \(-\log_{10}\) values.
filename  File name: volcano.pdf, volcano.svg, volcano.png. If NULL returns the ggplot object.
ylab  y axis text. Default: \(-\log_{10}\) FDR corrected P-values
xlab  x axis text. Default: No text. Examples of input: expression(paste(Log[2], "FoldChange"))
title  main title. If not specified it will be "Volcano plot (group1 vs group2"
legend  Legend title
label  vector of labels to be used in the figure. Example: c("Not Significant","Hypermethylated in group1", "Hypomethylated in group1")
xlim  x limits to cut image (i.e. c(-4,4))
ylim  y limits to cut image (i.e. c(-1,10))
color  vector of colors to be used in graph
names  Names to be plotted if significant. Should be the same size of x and y
names.fill  Names should be filled in a color box? Default: TRUE
show.names  What names will be showed? Possibilities: "both", "significant", "highlighted"

x.cut  x-axis threshold. Default: 0.0 If you give only one number (e.g. 0.2) the cut-offs will be -0.2 and 0.2. Or you can give different cut-offs as a vector (e.g. c(-0.3,0.4))

y.cut  q-values threshold (i.e. 0.01, 10^-10)

height  Figure height

width  Figure width

highlight  List of genes/probes to be highlighted. It should be in the names argument.

highlight.color  Color of the points highlighted

names.size  Size of the names text

dpi  Figure dpi

Details

Creates a volcano plot from the gene expression and DNA methylation analysis. Please see the vignette for more information

Value

Saves the volcano plot in the current folder

Examples

```r
log2_foldchange <- runif(200, -2, 2)
fdr <- runif(200, 0.01, 1)
TCGAVisualize_volcano(
  x = log2_foldchange,
  y = fdr,
  x.cut = 1.5,
  y.cut = 0.01,
  title = "Title example",
  xlab = expression(paste(Log[2], "FoldChange"))
)

## Not run:
beta_diff <- runif(200, -1, 1)
fdr <- runif(200, 0.01, 1)
TCGAVisualize_volcano(
  x = beta_diff,
  y = fdr,
  x.cut = 1.5,
  y.cut = 0.01,
  title = "Title example",
  xlab = expression(paste("DNA Methylation difference (", beta, "-values)"))
)
TCGAVisualize_volcano(
  x,
  y,
  filename = NULL,
)```
TCGA_MolecularSubtype  

Retrieves molecular subtypes for given TCGA barcodes.

**Description**

TCGA_MolecularSubtype retrieve molecular subtypes from TCGA consortium for a given set of barcodes.

**Usage**

```r
TCGA_MolecularSubtype(barcodes)
```

**Arguments**

- `barcodes` is a vector of TCGA barcodes.
Value

List with $subtypes attribute as a dataframe with barcodes, samples, subtypes, and colors. The $filtered attribute is returned as filtered samples with no subtype info.

Examples

TCGA_MolecularSubtype("TCGA-60-2721-01A-01R-0851-07")

Description

A dataset containing the Sample Ids from TCGA tumor purity measured according to 4 estimates attributes of 9364 tumor patients.

Usage

Tumor.purity

Format

A data frame with 9364 rows and 7 variables:

- **Sample.ID**  Sample ID from TCGA barcodes, character string
- **Cancer.type**  Cancer type, character string
- **ESTIMATE**  uses gene expression profiles of 141 immune genes and 141 stromal genes, 0-1 value
- **ABSOLUTE**  uses somatic copy-number data (estimations were available for only 11 cancer types), 0-1 value
- **LUMP**  (leukocytes unmethylation for purity), which averages 44 non-methylated immune-specific CpG sites, 0-1 value
- **IHC**  as estimated by image analysis of haematoxylin and eosin stain slides produced by the Nationwide Childrens Hospital Biospecimen Core Resource, 0-1 value
- **CPE**  derived consensus measurement as the median purity level after normalizing levels from all methods to give them equal means and s.ds, 0-1 value ...

Source

https://images.nature.com/original/nature-assets/ncomms/2015/151204/ncomms9971/extref/ncomms9971-s2.xlsx
UseRaw_afterFilter

Use raw count from the DataPrep object which genes are removed by normalization and filtering steps.

Description
function to keep raw counts after filtering and/or normalizing.

Usage
UseRaw_afterFilter(DataPrep, DataFilt)

Arguments
- DataPrep: DataPrep object returned by TCGAanalyze_Preprocessing()
- DataFilt: Filtered data frame containing samples in columns and genes in rows after normalization and/or filtering steps

Value
Filtered return object similar to DataPrep with genes removed after normalization and filtering process.

Examples
```r
## Not run:
dataPrep_raw <- UseRaw_afterFilter(dataPrep, dataFilt)

## End(Not run)
```
Index

* datasets
  TabSubtypesCol_merged, 44
  Tumor.purity, 89
* internal
  batch.info, 5
  bcgsc.ca_CHOL.IlluminaHiSeq_DNASeq.1.somatic.maf, 5
  chol.maf, 5
  classification, 6
  clinBRCA, 6
  clinical.biotab, 6
  dataBRCA, 7
  dataDEGsFiltLevel, 7
  dataREAD, 8
  dataREAD_df, 8
  DE_PCBC_stemSig, 8
  dmc.non.parametric, 9
  dmc.non.parametric.se, 10
  EB_PCBC_stemSig, 11
  ECTO_PCBC_stemSig, 11
  gaiaCNVplot, 12
  gbm.exp.harmonized, 12
  gbm.exp.legacy, 13
  GDCdownload, 14
  GDCprepare, 14
  GDCprepare_clinic, 16
  GDCquery, 5
  GDCquery_ATAC_seq, 21
  GDCquery_clinic, 22
  geneInfo, 24
  geneInfoHT, 25
  GenesCutID, 26
  GeneSplitRegulon, 26
  gaiaCNVplot, 12
  gbm.exp.harmonized, 12
  gbm.exp.legacy, 13
  GDCdownload, 14
  GDCprepare, 14
  GDCprepare_clinic, 16
  GDCquery, 5
  GDCquery_ATAC_seq, 21
  GDCquery_clinic, 22
  geneInfo, 24
  geneInfoHT, 25
  GenesCutID, 26
  GeneSplitRegulon, 26
  gaiaCNVplot, 12
  gbm.exp.harmonized, 12
  gbm.exp.legacy, 13
  GDCdownload, 14
  GDCprepare, 14
  GDCprepare_clinic, 16
  GDCquery, 5
  GDCquery_ATAC_seq, 21
  GDCquery_clinic, 22
  geneInfo, 24
  geneInfoHT, 25
  GenesCutID, 26
  GeneSplitRegulon, 26
  gaiaCNVplot, 12
  gbm.exp.harmonized, 12
  gbm.exp.legacy, 13
  GDCdownload, 14
  GDCprepare, 14
  GDCprepare_clinic, 16
  GDCquery, 5
  GDCquery_ATAC_seq, 21
  GDCquery_clinic, 22
  geneInfo, 24
  geneInfoHT, 25
  GenesCutID, 26
  GeneSplitRegulon, 26
  gaiaCNVplot, 12
  gbm.exp.harmonized, 12
  gbm.exp.legacy, 13
  GDCdownload, 14
  GDCprepare, 14
  GDCprepare_clinic, 16
  GDCquery, 5
  GDCquery_ATAC_seq, 21
  GDCquery_clinic, 22
  geneInfo, 24
  geneInfoHT, 25
  GenesCutID, 26
  GeneSplitRegulon, 26
getProjectSummary, 35
getResults, 36
getSampleFilesSummary, 36
getTSS, 37
ggbplot, 38
ggSurvplot, 61
gliomaClassifier, 40

isServeOK, 41

matchedMetExp, 41
MESO_PCBC_stemSig, 42, 59
met.gbm.27k, 42
msi_results, 42

pancan2018, 43
PanCancerAtlas_subtypes, 43

SC_PCBC_stemSig, 43, 59
splitAPICall, 44

TabSubtypesCol_merged, 44
tabSurvKMcompleteDEGs, 45
TCGA_MolecularSubtype, 88
TCGAanalyze_analyseGRN, 45
TCGAanalyze_Clustering, 46
TCGAanalyze_DEA, 46
TCGAanalyze_DEA_Affy, 49
TCGAanalyze_DMC, 49
TCGAanalyze_EA, 52
TCGAanalyze_EAcomplete, 53
TCGAanalyze_Filtering, 54
TCGAanalyze_LevelTab, 55
TCGAanalyze_networkInference, 56
TCGAanalyze_Normalization, 57
TCGAanalyze_Pathview, 58
TCGAanalyze_Preprocessing, 58
TCGAanalyze_Stemness, 59
TCGAanalyze_survival, 60
TCGAanalyze_SurvivalKM, 62
TCGABatch_Correction, 64
TCGAbiolinks, 64
TCGAbiolinks-package, 4
TCGAbatch_Correction, 64
TCGAbioliolinks, 64
TCGAbioliolinks-package, 4
TCGApipeline_Affy, 65
TCGAquery_MatchedCoupledSampleTypes, 66
TCGAquery_recount2, 67
TCGAquery_SampleTypes, 67
TCGAquery_subsetype, 68

TCGA_tumor_purity, 69
TCGAvisualize_BarPlot, 70
TCGAvisualize_EAbarplot, 71
TCGAvisualize_Heatmap, 72
TCGAvisualize_meanMethylation, 75
TCGAvisualize_oncoprint, 77
TCGAvisualize_PCA, 80
TCGAvisualize_PCA, 80
TCGAvisualize_starburst, 81
TCGAvisualize_SurvivalCoxNET, 84
TCGAvisualize_volcano, 85
Tumor.purity, 89

UseRaw_afterFilter, 90