Package ‘TCGAbiolinks’

April 23, 2016

Type Package

Title TCGAbiolinks: An R/Bioconductor package for integrative analysis with TCGA data

Version 1.0.10

Date 2015-12-13

Author Antonio Colaprico, Tiago Chedraoui Silva, Catharina Olsen, Luciano Garofano, Davide Garolini, Claudia Cava, Thais Sabedot, Tathiane M. Malta, Stefano M. Pagnotta, Isabella Castiglioni, Michele Ceccarelli, Gianluca Bontempi, Houtan Noushmehr

Maintainer Antonio Colaprico <`antonio.colaprico@ulb.ac.be`>, Tiago Chedraoui Silva <`tiagochst@usp.br`>

Depends R (>= 3.2)

Imports downloader (>= 0.4), GGally, grDevices, graphics, GenomicRanges, XML, Biobase, affy, heatmap.plus, xtable, data.table, EDASeq (>= 2.0.0), RCurl, edgeR (>= 3.0.0), rjson, plyr, CNTools, cghMCR, biomaRt, coin, gplots, ggplot2, survival, stringr (>= 1.0.0), IRanges, scales, rvest, stats, utils, dnet, igraph, supraHex, S4Vectors, SummarizedExperiment, BiocGenerics, GenomicFeatures, TxDb.Hsapiens.UCSC.hg19.knownGene, limma, knitr, devtools, genefilter, ConsensusClusterPlus, RColorBrewer, doParallel, dplyr, parallel, xml2

Description 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.

License GPL (>= 3)

biocViews DNAMethylation, DifferentialMethylation, GeneRegulation, GeneExpression, MethylationArray, DifferentialExpression, Pathways, Network, Survival

Suggests testthat, png, BiocStyle, rmarkdown

VignetteBuilder knitr

LazyData true

RoxygenNote 5.0.1

NeedsCompilation no

R topics documented:

TCGAanalyze_Clustering 3
TCGAanalyze_copynumber 4
TCGAanalyze_DEA 4
TCGAanalyze_DEA_Affy 5
TCGAanalyze_DMR 6
TCGAanalyze_EA 8
TCGAanalyze_EAcomplete 9
TCGAanalyze_Filtering 10
TCGAanalyze_LevelTab 11
TCGAanalyze_Normalization 12
TCGAanalyze_Preprocessing 13
TCGAanalyze_survival 13
TCGAanalyze_SurvivalKM 15
TCGAbiolinks 16
TCGAdownload 16
TCGAprepare 17
TCGAprepare_Affy 19
TCGAprepare_elmer 20
TCGAquery 20
TCGAquery_clinic 22
TCGAquery_clinicFilt 23
TCGAquery_integrate 25
TCGAquery_Investigate 25
TCGAquery_maf 26
TCGAquery_MatchedCoupledSampleTypes 27
TCGAquery_samplesfilter 27
TCGAquery_SampleTypes 28
TCGAquery_Social 29

TCGAquery_subtype 29
TCGAquery_Version 30
TCGAvisualize_BarPlot 31
TCGAvisualize_EAbarplot 32
TCGAvisualize_Heatmap 33
TCGAvisualize_meanMethylation 34
TCGAvisualize_PCA 35
TCGAvisualize_starburst 36
TCGAvisualize_SurvivalCoxNET 38
TCGAvisualize_Tables 40
TCGAVisualize_volcano 40

Index **42**

`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

```
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 asssignments). ConsensusClusterPlus also produces images.

TCGAanalyze_copynumber *Identifying Segment Gain Or Loss (SGOL)*

Description

Identifying Segment Gain Or Loss (SGOL) using findMCR package Example: query <- TCGA-query("acc","genome_wide_snp_6", level = 3) TCGAdownload(query,"trash") data <- TCGAprepare(query,"trash", type = "nocnv_hg19")

Usage

```
TCGAanalyze_copynumber(data = NULL, threshold = 0)
```

Arguments

`data`	Copy number data from TCGAPrepare
`threshold`	Gain/Loss threshold

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

```
## Not run:
 query <- TCGAquery("acc", platform = "genome_wide_snp_6", level = 3)
 TCGAdownload(query,path = "trash", samples = "TCGA-OR-A5JH-01A-11D-A309-01")
 segment_data <- TCGAprepare(query, dir = "trash",  type = "nocnv_hg19",
                     samples = "TCGA-OR-A5JH-01A-11D-A309-01")
 TCGAanalyze_copynumber(segment_data)

## End(Not run)
```

TCGAanalyze_DEA *Differentially expression analysis (DEA) using edgeR package.*

Description

TCGAanalyze_DEA allows user to perform Differentially expression analysis (DEA), using edgeR package 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.

Usage

```
TCGAanalyze_DEA(mat1, mat2, Cond1type, Cond2type, method = "exactTest",
  fdr.cut = 1, logFC.cut = 0)
```

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
`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)
`method`	is 'glmLRT' (1) or 'exactTest' (2). (1) Fit a negative binomial generalized log-linear model to the read counts for each gene (2) Compute genewise exact tests for differences in the means between two groups of negative-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

Value

table with DEGs containing for each gene logFC, logCPM, pValue,and FDR

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],"Normal", "Tumor")
```

`TCGAanalyze_DEA_Affy` *Differentially expression analysis (DEA) using limma package.*

Description

Differentially expression analysis (DEA) using limma package.

Usage

```
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

```
## Not run:
to add example

## End(Not run)
```

`TCGAanalyze_DMR` *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 collumns with the results from the object.

Usage

```
TCGAanalyze_DMR(data, groupCol = NULL, group1 = NULL, group2 = NULL,
  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, diffmean.cut = 0.2, paired = FALSE,
  adj.method = "BH", overwrite = FALSE, cores = 1, save = TRUE,
  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
`plot.filename`	Filename. Default: volcano.pdf, volcano.svg, volcano.png
`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
`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
`diffmean.cut`	diffmean threshold. Default: 0.2
`paired`	Wilcoxon paired parameter. Default: FALSE
`adj.method`	Adjusted method for the p-value calculation
`overwrite`	Overwrite the pvalues and diffmean values if already in the object for both groups? Default: FALSE
`cores`	Number of cores to be used in the non-parametric test Default = groupCol.group1.group2.rda
`save`	Save object with results? Default: TRUE
`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

```
nrows <- 200; ncols <- 20
counts <- matrix(runif(nrows * ncols, 1, 1e4), 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"), 5),
                   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)
SummarizedExperiment::colData(data)$group <- c(rep("group1",ncol(data)/2),
                         rep("group2",ncol(data)/2))
hypo.hyper <- TCGAanalyze_DMR(data, p.cut = 0.85,"group","group1","group2")
```

TCGAanalyze_EA *Enrichment analysis of a gene-set with GO [BP,MF,CC] and pathways.*

Description

The rational behind a 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. ie the confidence that 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 peform the analysis tests the null hypothesis that, for any particular ontology term, there is no diffeerence 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

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

Arguments

GeneName	is the name of gene signatures list
RegulonList	is a gene signature (lisf 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 informations about genes such as ID, Gene, Description, Location and Family.
GOtype	is type of gene ontology Biological process (BP), Molecular Function (MF), Cellular componet (CC)
FDRThresh	pvalue corrected (FDR) as threshold to selected significant BP, MF,CC, or pathways. (default FDR < 0.01)

Value

Table with enriched GO or pathways by selected gene signature.

Examples

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

## End(Not run)
```

`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

```
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

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

Usage

```
TCGAanalyze_Filtering(tabDF, method, qnt.cut = 0.25, var.func = IQR,
  var.cutoff = 0.75, eta = 0.05, foldChange = 1)
```

Arguments

`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`	is threshold selected as mean for filtering
`var.func`	is function used as the per-feature filtering statistic. See genefilter documentation
`var.cutoff`	is a numeric value. See genefilter documentation
`eta`	is a paramter for filter1. default eta = 0.05.
`foldChange`	is a paramter for filter2. default foldChange = 1.

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],
"Normal", "Tumor")
dataDEGsFilt <- dataDEGs[abs(dataDEGs$logFC) >= 1,]
dataTP <- dataFilt[,samplesTP]
dataTN <- dataFilt[,samplesNT]
dataDEGsFiltLevel <- TCGAanalyze_LevelTab(dataDEGsFilt,"Tumor","Normal",
dataTP,dataTN)
```

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 istance 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
`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_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 pearson 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, filename = NULL,
  width = 500, height = 500)
```

Arguments

`object`	of 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
`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

Examples

```
query <- TCGAquery(tumor = "lgg")
```

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, cutoff = 0,
  main = "Kaplan-Meier Overall Survival Curves",
  ylab = "Probability of survival", xlab = "Time since diagnosis (days)",
  filename = "survival.pdf", color = NULL, height = 8, width = 12,
  print.value = 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
`cutoff`	xlim This parameter will be a limit in the x-axis. That means, that patients with days_to_deth > cutoff will be set to Alive.
`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 of the lines.
`height`	Image height
`width`	Image width
`print.value`	Print pvalue in the plot? Default: TRUE

Value

Survival plot

Examples

```
days_to_death <- floor(runif(200, 1, 1000))
vital_status <- c(rep("Dead",200))
groups <- c(rep(c("G1","G2"),c(100,100)))
df <- data.frame(days_to_death,vital_status,groups)
TCGAanalyze_survival(df,clusterCol="groups")
## Not run:
clinical <- TCGAquery_clinic("gbm","clinical_patient")
TCGAanalyze_survival(clinical,"gender", filename = "surv.pdf", legend="Gender")

## End(Not run)
```

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 complte 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,
  ThreshTop = 0.67, ThreshDown = 0.33, p.cut = 0.05)
```

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_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.
`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

Value

table with survival genes pvalues from KM.

Examples

```
query <- TCGAquery(tumor = "lgg")
## Not run:
clinical_patient_Cancer <- TCGAquery_clinic("brca","clinical_patient")
dataBRCAcomplete <- log2(BRCA_rnaseqv2)
# Selecting only 10 genes for example
dataBRCAcomplete <- dataBRCAcomplete[1:10,]
tabSurvKM<-TCGAanalyze_SurvivalKM(clinical_patient_Cancer,dataBRCAcomplete,
Genelist = rownames(dataBRCAcomplete), Survresult = FALSE,ThreshTop=0.67,ThreshDown=0.33)

## End(Not run)
```

`TCGAbiolinks` *Download data of samples from TCGA*

Description

TCGAbiolinks allows you to Download data of samples from TCGA

Details

The functions you're likely to need from **TCGAbiolinks** is `TCGAdownload`, `TCGAquery`. Otherwise refer to the vignettes to see how to format the documentation.

`TCGAdownload` *Download the data from TCGA using as reference the output from TCGAquery*

Description

The TCGAdownload function will download the data using as reference the the lines of the TCGA-query search result.

There is an option to download the entire tar.gz folder or download specific files using the *type* parameter or the *samples* parameter

The outpufiles will be saved into the path parameters. If this path does not exists the package will try to create the directories.

By default, if a sample was already downloaded the function will not download again, unless the force parameter is set to `TRUE`

Usage

```
TCGAdownload(data = NULL, path = ".", type = NULL, samples = NULL,
  force = FALSE)
```

Arguments

`data`	The TCGAquery output
`path`	Directory to save the downloaded data
`type`	Filter the files that will be downloaded by type. Example:"rsem.genes.results"
`samples`	List of samples to download data
`force`	Download files even if it was already downladed? Default: `FALSE`

Value

Download TCGA data into the given path

See Also

`TCGAquery` for searching the data to download

`TCGAprepare` for preparing the data for the user into a Summarized experiment object, or a matrix.

Other data.functions: `TCGAprepare`, `TCGAquery`

Examples

```
samples <- c("TCGA-26-1442-01A-01R-1850-01")
query <- TCGAquery(tumor = "gbm",
                   platform = "IlluminaHiSeq_RNASeqV2",
                   level = "3",
                   samples = samples)
TCGAdownload(query,path = "RNA",
             samples = samples,
             type ="rsem.genes.results")
```

`TCGAprepare` *Read the data from level 3 the experiments and prepare it for downstream analysis into a SummarizedExperiment object.*

Description

This function will read the data from level 3 the experiments and prepare it for downstream analysis into a SummarizedExperiment object.

The samples are always refered by their barcode.

If you want to save the data into an rda file, please use the *save* parameter that will save an rda object with the *filename* parameter. If no filename was set, the filename will be the concatenation of platform and Sys.time.

List of accepted platforms:

- AgilentG4502A_07_1/AgilentG4502A_07_2/AgilentG4502A_07_3
- Genome_Wide_SNP_6

- H-miRNA_8x15K/H-miRNA_8x15Kv2
- HG-U133_Plus_2
- HT_HG-U133A
- HumanMethylation27
- HumanMethylation450
- IlluminaDNAMethylation_OMA002_CPI
- IlluminaDNAMethylation_OMA003_CPI
- IlluminaGA_RNASeq
- IlluminaGA_RNASeqV2
- IlluminaHiSeq_RNASeq
- IlluminaHiSeq_RNASeqV2
- IlluminaHiSeq_TotalRNASeqV2

ReturnThe default output is a SummarizedExperiment object.

Usage

```
TCGAprepare(query, dir = NULL, samples = NULL, type = NULL,
  save = FALSE, filename = NULL, add.mutation.genes = FALSE,
  reannotate = FALSE, summarizedExperiment = TRUE, add.subtype = FALSE)
```

Arguments

`query`	TCGAquery output
`dir`	Directory with the files downloaded by TCGAdownload
`samples`	List of samples to prepare the data
`type`	Filter the files to prepare.
`save`	Save a rda object with the prepared object? Default: `FALSE`
`filename`	Name of the saved file
`add.mutation.genes`	Integrate information about genes mutation? DEFAULT: FALSE
`reannotate`	Reannotate genes? Source http://grch37.ensembl.org/. DEFAULT: FALSE. (For the moment only working for methylation data)
`summarizedExperiment`	Output as SummarizedExperiment? Default: `FALSE`
`add.subtype`	Add subtype information from tcgaquery_subtype? Default: `FALSE`

Value

A SummarizedExperiment object (If SummarizedExperiment = `FALSE`, a data.frame)

See Also

TCGAquery for searching the data to download

TCGAdownload for downloading the data from the search

Other data.functions: TCGAdownload, TCGAquery

Examples

```
sample <- "TCGA-06-0939-01A-01D-1228-05"
query <- TCGAquery(tumor = "GBM",samples = sample, level = 3)
TCGAdownload(query,path = "exampleData",samples = sample)
data <- TCGAprepare(query, dir="exampleData")
```

TCGAprepare_Affy *Prepare CEL files into an AffyBatch.*

Description

Prepare CEL files into an AffyBatch.

Usage

```
TCGAprepare_Affy(ClinData, PathFolder, TabCel)
```

Arguments

ClinData	write
PathFolder	write
TabCel	write

Value

Normalizd Expression data from Affy eSets

Examples

```
## Not run:
to add example

## End(Not run)
```

TCGAprepare_elmer — *Prepare the data for ELEMR package*

Description

Prepare the data for ELEMR package

Usage

```
TCGAprepare_elmer(data, platform, met.na.cut = 0.2, save = FALSE)
```

Arguments

data	A data frame or summarized experiment from TCGAPrepare
platform	platform of the data. Example: "HumanMethylation450", "IlluminaHiSeq_RNASeqV2"
met.na.cut	Define the percentage of NA that the line should have to remove the probes for humanmethylation platforms.
save	Save object? Default: FALSE. Names of the files will be: "Exp_elmer.rda" (object Exp) and "Met_elmer.rda" (object Met)

Value

Matrix prepared for fetch.mee function

Examples

```
df <- data.frame(runif(200, 1e5, 1e6),runif(200, 1e5, 1e6))
rownames(df) <- sprintf("?|%03d", 1:200)
df <- TCGAprepare_elmer(df,platform="IlluminaHiSeq_RNASeqV2")
```

TCGAquery — *Searches TCGA open-access data providing also latest version of the files.*

Description

The TCGAquery searches TCGA open-access data through a preprocessed table with the latest version of the files.

Some parameters as tumor, platorm, level, center, barcode can be used to search the data.

TCGAquery will return a matrix with the results found, that will be used in the other function TCGAdownload, TCGAprepare.

Usage

```
TCGAquery(tumor = NULL, platform = NULL, samples = NULL, center = NULL,
  level = NULL, version = NULL)
```

Arguments

`tumor` Disease Examples:

OV	BRCA	CESC	ESCA	PCPG
LUSC	LGG	SKCM	KICH	CHOL
GBM	UCEC	PRAD	PAAD	THYM
KIRC	THCA	SARC	LAML	TGCT
COAD	KIRP	HNSC	ACC	UVM
READ	BLCA	DLBC	UCS	FPPP
LUAD	LIHC	STAD	MESO	CNTL

`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
Genome_Wide_SNP_6	bio
tissue_images	Mixed_DNASeq_automated
HumanMethylation27	Mixed_DNASeq_Cont_curated
IlluminaHiSeq_RNASeqV2	Mixed_DNASeq_Cont

`samples` List of samples. Ex:c('TCGA-04-06','TCGA-61-1743-01A-01D-0649-04')

`center` center name

`level` '1' '2' '3'

`version` List of vector with tumor/plaform/version to get old samples,

Value

A dataframe with the results of the query (lastest version of the files)

See Also

`TCGAdownload` for downloading the data from the search

`TCGAprepare` for preparing the data for the user into a Summarized experiment object, or a matrix.

Other data.functions: `TCGAdownload`, `TCGAprepare`

Examples

```
query <- TCGAquery(tumor = "gbm")

query <- TCGAquery(tumor = c("gbm","lgg"),
                   platform = c("HumanMethylation450","HumanMethylation27"))

query <- TCGAquery(tumor = "gbm",
                   platform = "HumanMethylation450",
                   level = "3")

query <- TCGAquery(samples = "TCGA-61-1743-01A-01D-0649-04",
                   tumor = "OV",
                   platform = "CGH-1x1M_G4447A",
                   level = 3)

# Get all LGG IlluminaHiSeq_RNASeqV2 data, but change with data version 11

query <- TCGAquery(tumor = "LGG", platform = "IlluminaHiSeq_RNASeqV2", level = "3",
                   version = list(c("IlluminaHiSeq_RNASeqV2","LGG",11)))
```

`TCGAquery_clinic` *Get the clinical information*

Description

Get the clinical information

Usage

```
TCGAquery_clinic(tumor, clinical_data_type, samples)
```

Arguments

`tumor` a character vector indicating cancer type Examples:

OV	BRCA	CESC	ESCA	PCPG
LUSC	LGG	SKCM	KICH	CHOL
GBM	UCEC	PRAD	PAAD	THYM

KIRC	THCA	SARC	LAML	TGCT
COAD	KIRP	HNSC	ACC	UVM
READ	BLCA	DLBC	UCS	FPPP
LUAD	LIHC	STAD	MESO	CNTL

For information about cancer types: https://tcga-data.nci.nih.gov/tcga/

`clinical_data_type`
a character vector indicating the types of clinical data Example:

biospecimen_aliquot	biospecimen_analyte
biospecimen_cqcf	biospecimen_diagnostic_slides
biospecimen_normal_control	biospecimen_portion
biospecimen_protocol	biospecimen_sample
biospecimen_shipment_portion	biospecimen_slide
biospecimen_tumor_sample	clinical_cqcf
clinical_drug	clinical_follow_up_v1.5
clinical_follow_up_v2.1	clinical_follow_up_v4.0
clinical_follow_up_v4.0_nte	clinical_nte
clinical_omf_v4.0	clinical_patient
clinical_radiation	

`samples` List of barcodes to get the clinical data

Value

clinic data

Examples

```
data <- TCGAquery_clinic("LGG","clinical_drug")
data <- TCGAquery_clinic(clinical_data_type = "clinical_patient",
samples = c("TCGA-06-5416-01A-01D-1481-05",
            "TCGA-2G-AAEW-01A-11D-A42Z-05",
            "TCGA-2G-AAEX-01A-11D-A42Z-05"))
```

`TCGAquery_clinicFilt` *Filter samples using clinical data*

Description

This function will return the samples that matches all filters. Filters available: HER, ER,gender,PR, stage.

Usage

```
TCGAquery_clinicFilt(barcode, clinical_patient_data, HER = NULL, ER = NULL,
  gender = NULL, PR = NULL, stage = NULL)
```

Arguments

`barcode`	List of barcodes
`clinical_patient_data`	clinical_patient_data obtained with clinic function Ex: clinical_patient_data <- TCGAquery_clinic("LGG","clinical_patient")
`HER`	her2 neu immunohistochemistry receptor status: "Positive" or "Negative"
`ER`	Estrogen receptor status: "Positive" or "Negative"
`gender`	"MALE" or "FEMALE"
`PR`	Progesterone receptor status: "Positive" or "Negative"
`stage`	Pathologic Stage: "stage_IX", "stage_I", "stage_IA", "stage_IB", "stage_IIX", "stage_IIA", "stage_IIB", "stage_IIIX","stage_IIIA", "stage_IIIB", "stage_IIIC", "stage_IV" -

Value

List of samples that matches the filters

Examples

```
# clin <- TCGAquery_clinic("BRCA","clinical_patient")
clin <- clinBRCA
bar <- c("TCGA-G9-6378-02A-11R-1789-07", "TCGA-CH-5767-04A-11R-1789-07",
         "TCGA-G9-6332-60A-11R-1789-07", "TCGA-G9-6336-01A-11R-1789-07",
         "TCGA-G9-6336-11A-11R-1789-07", "TCGA-G9-7336-11A-11R-1789-07",
         "TCGA-G9-7336-04A-11R-1789-07", "TCGA-G9-7336-14A-11R-1789-07",
         "TCGA-G9-7036-04A-11R-1789-07", "TCGA-G9-7036-02A-11R-1789-07",
         "TCGA-G9-7036-11A-11R-1789-07", "TCGA-G9-7036-03A-11R-1789-07",
         "TCGA-G9-7036-10A-11R-1789-07", "TCGA-BH-A1ES-10A-11R-1789-07",
         "TCGA-BH-A1F0-10A-11R-1789-07", "TCGA-BH-A0BZ-02A-11R-1789-07",
         "TCGA-B6-A0WY-04A-11R-1789-07", "TCGA-BH-A1FG-04A-11R-1789-08",
         "TCGA-D8-A1JS-04A-11R-2089-08", "TCGA-AN-A0FN-11A-11R-8789-08",
         "TCGA-AR-A2LQ-12A-11R-8799-08", "TCGA-AR-A2LH-03A-11R-1789-07",
         "TCGA-BH-A1F8-04A-11R-5789-07", "TCGA-AR-A24T-04A-55R-1789-07",
         "TCGA-AO-A0J5-05A-11R-1789-07", "TCGA-BH-A0B4-11A-12R-1789-07",
         "TCGA-B6-A1KN-60A-13R-1789-07", "TCGA-AO-A0J5-01A-11R-1789-07",
         "TCGA-AO-A0J5-01A-11R-1789-07", "TCGA-G9-6336-11A-11R-1789-07",
         "TCGA-G9-6380-11A-11R-1789-07", "TCGA-G9-6380-01A-11R-1789-07",
         "TCGA-G9-6340-01A-11R-1789-07","TCGA-G9-6340-11A-11R-1789-07")

TCGAquery_clinicFilt(c("TCGA-3C-AALK","TCGA-A2-A04Q","TCGA-A4-A04Q"),clin,
HER="Positive", gender="FEMALE",ER="Positive")
```

`TCGAquery_integrate` *Filtering common samples among platforms from TCGAquery for the same tumor*

Description

In order to help the user to have an overview of the number of samples in common we created the function ‘TCGAquery_integrate‘ that will receive the data frame returned from ‘TCGAquery‘ and produce a matrix n platforms x n platforms with the values of samples in commun.

Usage

```
TCGAquery_integrate(query)
```

Arguments

`query`	is the output of TCGAquery

Value

table with common samples among platforms from TCGAquery

Examples

```
query <- TCGAquery(tumor = 'brca',level = 3)
matSamples <- TCGAquery_integrate(query)
```

`TCGAquery_Investigate` *Find most studied TF in pubmed related to a specific cancer, disease, or tissue*

Description

Find most studied TF in pubmed related to a specific cancer, disease, or tissue

Usage

```
TCGAquery_Investigate(tumor, dataDEGsFiltLevelTF, topgenes)
```

Arguments

`tumor`	is character such as cancer, disease, or tissue eg. BRCA or breast
`dataDEGsFiltLevelTF`	is a table output from TCGAanalyze_LevelTab with only TFs
`topgenes`	is the number of top genes (eg. 10) in the rownames(dataDEGsFiltLevelTF) where find in pubmed if those genes or TFs are already related to that cancer or disease

Value

table with number of pubmed's publications related to tfs and disease selected

Examples

```
query <- TCGAquery(tumor = "lgg")
## Not run:
TFs <- EAGenes[EAGenes$Family =="transcription regulator",]
TFs_inDEGs <- intersect(TFs$Gene, dataDEGsFiltLevel$mRNA )
dataDEGsFiltLevelTFs <- dataDEGsFiltLevel[TFs_inDEGs,]
dataDEGsFiltLevelTFs <- dataDEGsFiltLevelTFs[order(dataDEGsFiltLevelTFs$Delta,decreasing = TRUE),]
# Find Pubmed of TF studied related to cancer
tabDEGsTFPubmed <- TCGAquery_Investigate("breast", dataDEGsFiltLevelTFs, topgenes = 1)

## End(Not run)
```

`TCGAquery_maf` *Get last maf file for the tumor*

Description

Filtering sample output from TCGAquery

Usage

```
TCGAquery_maf(tumor = NULL, center = NULL, archive.name = NULL)
```

Arguments

`tumor`	tumor type to filter the search
`center`	Center name to filter the search
`archive.name`	Archive name to filter the search

Value

list of samples for a tumor

Examples

```
## Not run:
 query <- TCGAquery(tumor = 'lgg')

## End(Not run)
```

`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_samplesfilter`

Filtering sample output from TCGAquery

Description

Filtering sample output from TCGAquery

Usage

```
TCGAquery_samplesfilter(query)
```

Arguments

`query`	metaData output from TCGAquery

Value

list of samples for a tumor

Examples

```
query <- TCGAquery(tumor = 'brca',level = 3)
querySamples <- TCGAquery_samplesfilter(query)
```

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 theses 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_Social` *Finds the number of downloads of a package on CRAN or BIOC and find questions in website ("bioconductor.org", "biostars.org", "stackoverflow).*

Description

Finds the number of downloads of a package on CRAN or BIOC

Usage

```
TCGAquery_Social(siteToFind = NULL, listPackage = NULL, KeyInfo = NULL)
```

Arguments

`siteToFind`	website ("bioconductor.org", "biostars.org", "stackoverflow) related to TCGA or a package
`listPackage`	list of package to find in bioconductor
`KeyInfo`	is a key to find in biostars related to TGGA or package

Value

table with number of downloads about a package

Examples

```
TCGAquery_Social("bioconductor.org","BiocCheck")
```

`TCGAquery_subtype` *Retrieve molecular subtypes for a given tumor*

Description

TCGAquery_subtype Retrieve molecular subtypes for a given tumor

Usage

```
TCGAquery_subtype(tumor)
```

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")
```

`TCGAquery_Version` *Shows a summary (version, date, number of samples, size of the data) of all versions of data for a given tumor and platform.*

Description

As every new version of data has diferent samples and size the TCGAquery_Version function shows a summary (version, date, number of samples, size of the data) of all versions of data for a given tumor and platform.

Usage

```
TCGAquery_Version(tumor = NULL, platform = NULL)
```

Arguments

`tumor`	a character string indicating the cancer type for which to download data. Options include ACC, BLCA, BRCA, CESC, COAD, DLBC, ESCA, GBM, HNSC, KICH, KIRC, KIRP, LAML, LGG, LIHC, LUAD, LUSC, OV, PAAD, PRAD, READ, SARC, SKCM, STAD, THCA, UCEC, UCS. Look at https://tcga-data.nci.nih.gov/tcga/ for Available Cancer Types.
`platform`	illuminahiseq_rnaseq, agilentg4502a_07_3, illuminahiseq_rnaseqv2, humanmethylation27, humanmethylation450, illuminaga_mirnaseq, genome_wide_snp_6

Value

Data frame with version, date, number of samples,size of the platform and tumor

Examples

```
TCGAquery_Version('GBM','illuminahiseq_rnaseqv2')
```

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

Description

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

Usage

```
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

Examples

```
query <- TCGAquery(tumor = "lgg")
```

TCGAvisualize_EAbarplot *barPlot for a complete Enrichment Analysis*

Description

TCGAvisualize_EAbarplot plots the result from TCGAanalyze_EAcomplete in a complete barPlot

Usage

```
TCGAvisualize_EAbarplot(tf, GOMFTab, GOBPTab, GOCCTab, PathTab, nBar, nRGTab)
```

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.

Value

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

Examples

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

```
TCGAvisualize_Heatmap(cancer, DFfilt, DFclin, DFsubt, data_Hc2, cbPalette,
  filename = NULL)
```

Arguments

`cancer`	tumor selected for the analysis
`DFfilt`	write
`DFclin`	write
`DFsubt`	write
`data_Hc2`	write
`cbPalette`	write
`filename`	write. default = NULL

Value

Heatmap plotted in pdf or png file.

Examples

```
query <- TCGAquery(tumor = "lgg")
## Not run:
# from case study n.2 LGG to test the function
DFfilt <- datFilt
DFclin = dataClin
DFsubt = dataSubt
data_Hc2 = data_Hc2
TCGAvisualize_Heatmap(DFfilt,
DFclin,
DFsubt,
data_Hc2)

## End(Not run)
```

TCGAvisualize_meanMethylation *Mean methylation boxplot*

Description

Creates a mean methylation boxplot for groups (groupCol), subgroups will be highlited 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, 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)
```

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
`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

Value

Save the pdf survival plot

Examples

```
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")
TCGAvisualize_meanMethylation(data,groupCol  = "group", subgroupCol="subgroup")
```

`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)
```

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

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
TCGAvisualize_PCA(dataFilt,dataDEGsFiltLevel, ntopgenes = 200)
```

TCGAvisualize_starburst

Create starburst plot

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,
  names = FALSE, filename = "starburst.pdf",
  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,
  label = c("Not Significant", "Up regulated & Hypo methylated",
  "Down regulated & Hypo methylated", "hypo methylated", "hyper methylated",
  "Up regulated", "Down regulated", "Up regulated & Hyper methylated",
  "Down regulated & Hyper methylated"), xlim = NULL, ylim = NULL)
```

Arguments

`met`	SummarizedExperiment with methylation data obtained from the TCGAPrepare. 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.
`names`	Add the names of the significant genes? Default: FALSE
`filename`	The filename of the file (it can be pdf, svg, png, etc)
`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

Details

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

Value

Save a starburst plot

Examples

```
nrows <- 20000; ncols <- 20
counts <- matrix(runif(nrows * ncols, 1, 1e4), nrows)
ranges <- GenomicRanges::GRanges(rep(c("chr1", "chr2"), c(5000, 15000)),
                  IRanges::IRanges(floor(runif(20000, 1e5, 1e6)), width=100),
                   strand=sample(c("+", "-"), 20000, TRUE),
                   probeID=sprintf("ID%03d", 1:20000),
                   Gene_Symbol=sprintf("ID%03d", 1:20000))
colData <- S4Vectors::DataFrame(Treatment=rep(c("ChIP", "Input"), 5),
                   row.names=LETTERS[1:20],
                   group=rep(c("group1","group2"),c(10,10)))
data <- SummarizedExperiment::SummarizedExperiment(
         assays=S4Vectors::SimpleList(counts=counts),
         rowRanges=ranges,
         colData=colData)
met <- data
exp <- data.frame(row.names=sprintf("ID%03d", 1:20000),
                 logFC=runif(20000, -0.2, 0.2),
                 FDR=runif(20000, 0.01, 1))
```

```
SummarizedExperiment::rowRanges(met)$diffmean.g1.g2 <- c(runif(20000, -0.1, 0.1))
SummarizedExperiment::rowRanges(met)$p.value.g1.g2 <- c(runif(20000, 0, 1))
SummarizedExperiment::rowRanges(met)$p.value.adj.g1.g2 <- c(runif(20000, 0, 1))
result <- TCGAvisualize_starburst(met,exp,
                                  exp.p.cut = 0.05, met.p.cut = 0.05,
                                  group1="g1",group2="g2",
                                  diffmean.cut=0.8,
                                  names=TRUE)
```

TCGAvisualize_SurvivalCoxNET

Survival analysis with univariate Cox regression package (dnet)

Description

TCGAvisualize_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 build with community of genes with similar range of pvalues from Cox regression (same color) and that interaction among those genes is already validated in literatures using the STRING database (version 9.1). TCGAvisualize_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

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

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_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.
`org.Hs.string`	an igraph object that contains a functional protein association network in human. The network is extracted from the STRING database (version 10).
`scoreConfidence`	restrict to those edges with high confidence (eg. score>=700)
`titlePlot`	is the title to show in the final plot.

Details

TCGAvisualize_SurvivalCoxNET allow user to perform the complete workflow using coxph and 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 pvaules
2. fit a Cox proportional hazards model and ANOVA (Chisq test)
3. Network comunites
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. visualise the subnetwork

Value

net IGRAPH with related Cox survival genes in community (same pval and color) and with interactions from STRING database. query <- TCGAquery(tumor = "lgg")

`TCGAvisualize_Tables` *Visaulize results in format of latex tables.*

Description

Visaulize results in format of latex tables.

Usage

```
TCGAvisualize_Tables(Table, rowsForPage, TableTitle, LabelTitle, withrows, size)
```

Arguments

`Table`	write
`rowsForPage`	write
`TableTitle`	write
`LabelTitle`	write
`withrows`	write
`size`	size selected for font, 'small', 'tiny'

Value

table in latex format to use in beamer presentation or sweave files

Examples

```
library(stringr)
tabDEGsTFPubmed$PMID <- str_sub(tabDEGsTFPubmed$PMID,0,30)
TCGAvisualize_Tables(Table = tabDEGsTFPubmed,
rowsForPage = 5,
TableTitle = "pip",
LabelTitle = "pip2",
withrows = FALSE,
size = "small")
```

`TCGAVisualize_volcano` *Creates a volcano plot for DNA methylation or expression*

Description

Creates a volcano plot from the expression and methylation analysis.

Usage

```
TCGAVisualize_volcano(x, y, filename = "volcano.pdf",
  ylab = expression(paste(-Log[10], " (FDR corrected -P values)")),
  xlab = NULL, title = NULL, legend = NULL, label = NULL, xlim = NULL,
  ylim = NULL, color = c("black", "red", "green"), names = NULL,
  x.cut = 0, y.cut = 0.01, height = 5, width = 10)
```

Arguments

`x`	x-axis data
`y`	y-axis data
`filename`	Filename. Default: volcano.pdf, volcano.svg, volcano.png
`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
`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
`color`	vector of colors to be used in graph
`names`	Names to be ploted if significant. Should be the same size of x and y
`x.cut`	x-axis threshold. Default: 0.0
`y.cut`	p-values threshold. Default: 0.01
`height`	Figure height
`width`	Figure width

Details

Creates a volcano plot from the expression and methylation analysis. Please see the vignette for more information Observation: This function automatically is called by TCGAanalyse_DMR

Value

Saves the volcano plot in the current folder

Examples

```
x <- runif(200, 1e5, 1e6)
y <- runif(200, 1e5, 1e6)
TCGAVisualize_volcano(x,y)
```

Index

TCGAanalyze_Clustering, 3
TCGAanalyze_copynumber, 4
TCGAanalyze_DEA, 4
TCGAanalyze_DEA_Affy, 5
TCGAanalyze_DMR, 6
TCGAanalyze_EA, 8
TCGAanalyze_EAcomplete, 9
TCGAanalyze_Filtering, 10
TCGAanalyze_LevelTab, 11
TCGAanalyze_Normalization, 12
TCGAanalyze_Preprocessing, 13
TCGAanalyze_survival, 13
TCGAanalyze_SurvivalKM, 15
TCGAbiolinks, 16
TCGAbiolinks-package (TCGAbiolinks), 16
TCGAdownload, *16*, 16, *19*, *22*
TCGAprepare, *17*, 17, *22*
TCGAprepare_Affy, 19
TCGAprepare_elmer, 20
TCGAquery, *16*, *17*, *19*, 20
TCGAquery_clinic, 22
TCGAquery_clinicFilt, 23
TCGAquery_integrate, 25
TCGAquery_Investigate, 25
TCGAquery_maf, 26
TCGAquery_MatchedCoupledSampleTypes, 27
TCGAquery_samplesfilter, 27
TCGAquery_SampleTypes, 28
TCGAquery_Social, 29
TCGAquery_subtype, 29
TCGAquery_Version, 30
TCGAvisualize_BarPlot, 31
TCGAvisualize_EAbarplot, 32
TCGAvisualize_Heatmap, 33
TCGAvisualize_meanMethylation, 34
TCGAvisualize_PCA, 35
TCGAvisualize_starburst, 36
TCGAvisualize_SurvivalCoxNET, 38
TCGAvisualize_Tables, 40
TCGAVisualize_volcano, 40