Package 'TCGAutils'

April 16, 2019

Title TCGA utility functions for data management

Version 1.2.2

Description A suite of helper functions for checking and manipulating TCGA data including data obtained from the curatedTCGAData experiment package. These functions aim to simplify and make working with TCGA data more manageable.

Depends R (>= 3.5.0)

Imports AnnotationDbi, BiocGenerics, GenomeInfoDb, GenomicFeatures, GenomicRanges, GenomicDataCommons, IRanges, methods, MultiAssayExperiment, RaggedExperiment (>= 1.5.7), rvest, S4Vectors, stats, stringr, SummarizedExperiment, utils, xml2

Suggests BiocStyle, curatedTCGAData, devtools, impute, knitr, magrittr, mirbase.db, org.Hs.eg.db, readr, RTCGAToolbox (>= 2.7.5), testthat, TxDb.Hsapiens.UCSC.hg19.knownGene,

License Artistic-2.0

Encoding UTF-8

LazyData true

BugReports https://github.com/waldronlab/TCGAutils/issues

biocViews Software, WorkflowStep, Preprocessing

VignetteBuilder knitr

RoxygenNote 6.1.0

git_url https://git.bioconductor.org/packages/TCGAutils

git_branch RELEASE_3_8

git_last_commit b8b085b

git_last_commit_date 2019-02-13

Date/Publication 2019-04-15

Author Marcel Ramos [aut, cre],
Lucas Schiffer [aut],
Sean Davis [ctb],
Levi Waldron [aut]

Maintainer Marcel Ramos <marcel.ramos@roswellpark.org>

R topics documented:

`TCGAutils-package` *TCGAutils: Helper functions for working with TCGA and MultiAssayExperiment data*

Description

TCGAutils is a toolbox to work with TCGA specific datasets. It allows the user to manipulate and translate TCGA barcodes, conveniently convert a list of data files to GRangesList. Take datasets from GISTIC and return a SummarizedExperiment class object. The package also provides functions for working with data from the `curatedTCGAData` experiment data package. It provides convenience functions for extracting subtype metadata data and adding clinical data to existing MultiAssayExperiment objects.

Author(s)

Maintainer: Marcel Ramos `<marcel.ramos@roswellpark.org>`

Authors:

- Lucas Schiffer
- Levi Waldron

Other contributors:

- Sean Davis [contributor]

See Also

Useful links:

- Report bugs at https://github.com/waldronlab/TCGAutils/issues

---

`clinicalNames` *Clinical dataset names in TCGA*

Description

A dataset of names for each of the TCGA cancer codes available. These names were obtained by the clinical datasets from getFirehoseData. They serve to subset the current datasets provided by `curatedTCGAData`.

Usage

```
clinicalNames
```

Format

A CharacterList of names for 33 cancer codes

Value

The clinical dataset column names in TCGA as provided by the `RTCGAToolbox`

`curatedTCGAData-helpers`
Helper functions for managing MultiAssayExperiment from curatedTCGAData

---

Description

Additional helper functions for cleaning and uncovering metadata within a downloaded `MultiAssayExperiment` from `curatedTCGAData`. The `getSubtypeMap` function provides a 2 column `data.frame` with in-data variable names and an interpreted names. The `getClinicalNames` function provides a vector of variable names that exist in the `colData` slot of a downloaded `MultiAssayExperiment` object. These variables are obtained from getFirehoseData by default and tend to be present across most cancer codes.

Usage

```
getSubtypeMap(multiassayexperiment)

getClinicalNames(diseaseCode)

splitAssays(multiassayexperiment, sampleCodes = c("01", "11"))

sampleTables(multiassayexperiment, vial = FALSE)
```

Arguments

multiassayexperiment	A MultiAssayExperiment object
diseaseCode	A TCGA cancer code (e.g., "BRCA")
sampleCodes	A string of sample type codes (refer to data(sampleTypes); default "01", "11")
vial	(logical default FALSE) whether to display vials in the table output

Value

- getSubtypeMap: A data.frame with columns representing actual data variables and explanatory names
- getClinicalNames: A vector of names that correspond to a particular disease code.

splitAssays

Separates samples by indicated sample codes into different assays in a MultiAssayExperiment. Refer to the sampleTypes data object for a list of available codes. This operation generates **n** times the number of assays based on the number of sample codes entered. By default, primary solid tumors ("01") and solid tissue normals ("11") are seperated out.

sampleTables

Display all the available samples in each of the assays

Examples

```
## Not run:
library(curatedTCGAData)

coad <- curatedTCGAData(diseaseCode = "COAD",
    assays = "CNA*", dry.run = FALSE)
getSubtypeMap(coad)

## End(Not run)

getClinicalNames("COAD")
```

---

diseaseCodes *TCGA Cancer Disease Codes Table*

Description

A dataset for obtaining the cancer codes in TCGA for about 13 different types of cancers.

Usage

```
diseaseCodes
```

Format

A data frame with 37 rows and 2 variables:

Study.Abbreviation Disease Code used in TCGA

Available Cancer datasets available via curatedTCGAData

SubtypeData Subtype curation data available via curatedTCGAData

Study.Name The full length study name (i.e., type of cancer)

Value

The TCGA 'diseaseCodes' table

Source

https://gdc.cancer.gov/resources-tcga-users/tcga-code-tables/tcga-study-abbreviations

---

findGRangesCols *Obtain minimum necessary names for the creation of a GRangesList object*

Description

This function attempts to match chromosome, start position, end position and strand names in the given character vector. Modified helper from the GenomicRanges package.

Usage

```
findGRangesCols(df_colnames, seqnames.field = c("seqnames", "seqname",
  "chromosome", "chrom", "chr", "chromosome_name", "seqid", "om"),
  start.field = "start", end.field = c("end", "stop"),
  strand.field = "strand", ignore.strand = FALSE)
```

Arguments

df_colnames	A character vector of names in a dataset
seqnames.field	A character vector of the chromosome name
start.field	A character vector that indicates the column name of the start positions of ranged data
end.field	A character vector that indicates the end position of ranged data
strand.field	A character vector of the column name that indicates the strand type
ignore.strand	logical (default FALSE) whether to ignore the strand field in the data

Value

Index positions vector indicating columns with appropriate names

Examples

```
myDataColNames <- c("Start_position", "End_position", "strand",
                    "chromosome", "num_probes", "segment_mean")
findGRangesCols(myDataColNames)
```

generateMap *Create a sampleMap from an experiment list and phenoData dataframe*

Description

This function helps create a sampleMap in preparation of a `MultiAssayExperiment` object. This especially useful when the sample identifiers are not very different, as in the case of TCGA barcodes. An `idConverter` function can be provided to truncate such sample identifiers and obtain patient identifiers.

Usage

```
generateMap(experiments, colData, idConverter = identity, sampleCol,
  patientCol, ...)
```

Arguments

`experiments`	A named `list` of experiments compatible with the MultiAssayExperiment API
`colData`	A `data.frame` of clinical data with patient identifiers as rownames
`idConverter`	A function to be used against the sample or specimen identifiers to match those in the rownames of the `colData` (default NULL)
`sampleCol`	A single string indicating the sample identifiers column in the colData dataset
`patientCol`	A single string indicating the patient identifiers in colData, "row.names" extracts the colData row names
`...`	Additonal arguments to pass to the 'idConverter' function.

Value

A `DataFrame` class object of mapped samples and patient identifiers including assays

Author(s)

M. Ramos, M. Morgan, L. Schiffer

Examples

```
## Minimal example
expList <- list(assay1 = matrix(1:6, ncol = 2L,
        dimnames = list(paste0("feature", 1:3), c("A-J", "B-J"))),
    assay2 = matrix(1:4, ncol = 2,
        dimnames = list(paste0("gene", 1:2), c("A-L", "B-L"))))

## Mock colData
myPheno <- data.frame(var1 = c("Yes", "No"), var2 = c("High", "Low"),
    row.names = c("a", "b"))

## A look at the identifiers
vapply(expList, colnames, character(2L))
rownames(myPheno)
```

```
## Use 'idConverter' to correspond sample names to patient identifiers
generateMap(expList, myPheno,
    idConverter = function(x) substr(tolower(x), 1L, 1L))
```

`getFileNames` *Find the file names used in RTCGAToolbox*

Description

Part of this function is from the RTCGAToolbox. It aims to extract the file name used inside of the getFirehoseData function. The arguments of the function parallel those in the getFirehoseData function. It is only available for select data types.

Usage

```
getFileNames(disease, runDate = "20160128", CNASNP = FALSE,
  CNVSNP = FALSE, CNASeq = FALSE, CNACGH = FALSE)
```

Arguments

`disease`	The TCGA cancer disease code, e.g., "COAD"
`runDate`	The single `string` used in the `getFirehoseData` function (default "20160128")
`CNASNP`	A `logical` (default = FALSE) vector indicating whether to get the file name from this data type
`CNVSNP`	A `logical` (default = FALSE) vector indicating whether to get the file name from this data type
`CNASeq`	A `logical` (default = FALSE) vector indicating whether to get the file name from this data type
`CNACGH`	A `logical` (default = FALSE) vector indicating whether to get the file name from this data type

Value

A `character` vector of length one indicating the file name

Examples

```
getFileNames("COAD", CNVSNP = TRUE)
```

ID-translation *Translate study identifiers from barcode to UUID and vice versa*

Description

These functions allow the user to enter a character vector of identifiers and use the GDC API to translate from TCGA barcodes to Universally Unique Identifiers (UUID) and vice versa. These relationships are not one-to-one. Therefore, a `data.frame` is returned for all inputs. The UUID to TCGA barcode translation only applies to file and case UUIDs. Two-way UUID translation is available from 'file_id' to 'case_id' and vice versa. Please double check any results before using these features for analysis. Case / submitter identifiers are translated by default, see the `id_type` argument for details. All identifiers are converted to lower case.

Usage

```
UUIDtoBarcode(id_vector, id_type = c("case_id", "file_id"),
  end_point = "participant", legacy = FALSE)

UUIDtoUUID(id_vector, to_type = c("case_id", "file_id"),
  legacy = FALSE)

barcodeToUUID(barcodes, id_type = c("case_id", "file_id"),
  legacy = FALSE)

filenameToBarcode(filenames, legacy = FALSE)
```

Arguments

`id_vector`	A `character` vector of UUIDs corresponding to either files or cases (default assumes case_ids)
`id_type`	Either `case_id` or `file_id` indicating the type of `id_vector` entered (default "case_id")
`end_point`	The cutoff point of the barcode that should be returned, only applies to `file_id` type queries. See details for options.
`legacy`	(logical default FALSE) whether to search the legacy archives
`to_type`	The desired UUID type to obtain, can either be "case_id" or "file_id"
`barcodes`	A `character` vector of TCGA barcodes
`filenames`	A `character` vector of filenames obtained from the GenomicDataCommons

Details

The `end_point` options reflect endpoints in the Genomic Data Commons API. These are summarized as follows:

- participant: This default snippet of information includes project, tissue source site (TSS), and participant number (barcode format: TCGA-XX-XXXX)
- sample: This adds the sample information to the participant barcode (TCGA-XX-XXXX-11X)

- portion, analyte: Either of these options adds the portion and analyte information to the sample barcode (TCGA-XX-XXXX-11X-01X)
- plate, center: Additional plate and center information is returned, i.e., the full barcode (TCGA-XX-XXXX-11X-01X-XXXX-XX)

Only these keywords need to be used to target the specific barcode endpoint. These endpoints only apply to "file_id" type translations to TCGA barcodes (see `id_type` argument).

Value

A `data.frame` of TCGA barcode identifiers and UUIDs

Author(s)

Sean Davis, M. Ramos

Examples

```
## Translate UUIDs >> TCGA Barcode

uuids <- c("0001801b-54b0-4551-8d7a-d66fb59429bf",
"002c67f2-ff52-4246-9d65-a3f69df6789e",
"003143c8-bbbf-46b9-a96f-f58530f4bb82")

UUIDtoBarcode(uuids, id_type = "file_id", end_point = "sample")

UUIDtoBarcode("ae55b2d3-62a1-419e-9f9a-5ddfac356db4", id_type = "case_id")

## Translate file UUIDs >> case UUIDs

uuids <- c("0001801b-54b0-4551-8d7a-d66fb59429bf",
"002c67f2-ff52-4246-9d65-a3f69df6789e",
"003143c8-bbbf-46b9-a96f-f58530f4bb82")

UUIDtoUUID(uuids)

## Translate TCGA Barcode >> UUIDs

fullBarcodes <- c("TCGA-B0-5117-11A-01D-1421-08",
"TCGA-B0-5094-11A-01D-1421-08",
"TCGA-E9-A295-10A-01D-A16D-09")

sample_ids <- TCGAbarcode(fullBarcodes, sample = TRUE)

barcodeToUUID(sample_ids)

participant_ids <- c("TCGA-CK-4948", "TCGA-D1-A17N",
"TCGA-4V-A9QX", "TCGA-4V-A9QM")

barcodeToUUID(participant_ids)

library(GenomicDataCommons)

fquery <- files() %>%
    filter(~ cases.project.project_id == "TCGA-COAD" &
        data_category == "Copy Number Variation" &
```

```
        data_type == "Copy Number Segment")

fnames <- results(fquery)$file_name[1:6]

filenameToBarcode(fnames)
```

`imputeAssay` *This function imputes assays values inside a* `MultiAssayExperiment`

Description

These function allow the user to enter a `MultiAssayExperiment` and impute all the NA values inside assays.

Usage

```
imputeAssay(multiassayexperiment, i = 1, ...)
```

Arguments

`multiassayexperiment`
: A `MultiAssayExperiment` with genes in the rows, samples in the columns

`i`
: A numeric, logical, or character `vector` indicating the assays to perform imputation on (default 1L)

`...`
: Arguments passed on to `impute::impute.knn`

data An expression matrix with genes in the rows, samples in the columns

k Number of neighbors to be used in the imputation (default=10)

rowmax The maximum percent missing data allowed in any row (default 50%). For any rows with more than `rowmax`% missing are imputed using the overall mean per sample.

colmax The maximum percent missing data allowed in any column (default 80%). If any column has more than `colmax`% missing data, the program halts and reports an error.

maxp The largest block of genes imputed using the knn algorithm inside `impute.knn` (default 1500); larger blocks are divided by two-means clustering (recursively) prior to imputation. If `maxp=p`, only knn imputation is done.

rng.seed The seed used for the random number generator (default 362436069) for reproducibility.

Value

MultiAssayExperiment with imputed assays values

Examples

```
library(curatedTCGAData)

gbm <- curatedTCGAData("GBM", "RPPA*", FALSE)

## replace DataFrame with "matrix"
gbm[[1L]] <- as.matrix(assay(gbm[[1L]]))

gbm <- imputeAssay(gbm, i = 1L)
```

`makeGRangesListFromCopyNumber`

Make a GRangesList from TCGA Copy Number data

Description

`makeGRangesListFromCopyNumber` allows the user to convert objects of class `data.frame` or DataFrame to a GRangesList. It includes additional features specific to TCGA data such as, hugo symbols, probe numbers, segment means, and ucsc build (if available).

Usage

```
makeGRangesListFromCopyNumber(df, split.field,
  names.field = "Hugo_Symbol", ...)
```

Arguments

`df`	A `data.frame` or `DataFrame` class object. `list` class objects are coerced to data.frame or DataFrame.
`split.field`	A `character` vector of length one indicating the column to be used as sample identifiers
`names.field`	A `character` vector of length one indicating the column to be used as names for each of the ranges in the data
`...`	Additional arguments to pass on to makeGRangesListFromDataFrame

Value

A GRangesList class object

Examples

```
library(GenomicDataCommons)
library(magrittr)

manif <- files() %>%
    filter(~ cases.project.project_id == "TCGA-COAD" &
        data_type == "Copy Number Segment") %>%
    manifest(size = 1)

fname <- gdcdata(manif$id)
```

```
barcode <- UUIDtoBarcode(names(fname), id_type = "file_id")$cases.submitter_id

cndata <- read.delim(fname[[1L]], nrows = 10L)

cngrl <- makeGRangesListFromCopyNumber(cndata, split.field = "GDC_Aliquot",
    keep.extra.columns = TRUE)

names(cngrl) <- barcode
GenomeInfoDb::genome(cngrl) <- extractBuild(fname[[1L]])
cngrl
```

`makeGRangesListFromExonFiles` *Read Exon level files and create a GRangesList*

Description

This function serves to read exon-level expression data. It works for exon quantification (raw counts and RPKM) and junction quantification (raw counts) files paths and represent such data as a GRangesList. The data can be downloaded via the TCGA Legacy Archive. File name and structure requirements are as follows: The third position delimited by dots (".") in the file name should be the universally unique identifier (UUID). The column containing the ranged information is labeled "exon."

Usage

```
makeGRangesListFromExonFiles(filepaths, sampleNames = NULL,
  fileNames = NULL, rangesColumn = "exon")
```

Arguments

`filepaths`	A `character` vector of valid exon data file paths
`sampleNames`	A `character` vector of TCGA barcodes to be applied if not present in the data (default NULL)
`fileNames`	A `character` vector of file names as downloaded from the Genomic Data Commons Legacy archive (default NULL)
`rangesColumn`	(default "exon") A single string indicating the name of the column in the data containing the ranges information

Value

A GRangesList object

Author(s)

M. Ramos

Examples

```
## Load example file found in package
pkgDir <- system.file("extdata", package = "TCGAutils", mustWork = TRUE)
exonFile <- list.files(pkgDir, pattern = "cation\\.txt$", full.names = TRUE)

filePrefix <- "unc.edu.32741f9a-9fec-441f-96b4-e504e62c5362.1755371."

## Add actual file name manually (due to Windows OS restriction)
makeGRangesListFromExonFiles(exonFile,
    fileNames = paste0(filePrefix, basename(exonFile)),
    sampleNames = "TCGA-AA-3678-01A-01R-0905-07")
```

---

makeSummarizedExperimentFromGISTIC

Create a SummarizedExperiment from FireHose GISTIC

Description

Use the output of `getFirehoseData` to create a SummarizedExperiment. This can be done for three types of data, G-scores thresholded by gene, copy number by gene, and copy number by peak regions.

Usage

```
makeSummarizedExperimentFromGISTIC(gistic, dataType)
```

Arguments

`gistic`	A FirehoseGISTIC-class object
`dataType`	Either one of "ThresholdedByGene", "AllByGene", "Peaks"

Value

A `SummarizedExperiment` object

Author(s)

L. Geistlinger, M. Ramos

Examples

```
library(RTCGAToolbox)
co <- getFirehoseData("COAD", clinical = FALSE, GISTIC = TRUE,
    destdir = tempdir())
makeSummarizedExperimentFromGISTIC(co, "AllByGene")
```

`mergeColData` *Take a MultiAssayExperiment and include curated variables*

Description

This function works on the `colData` of a `MultiAssayExperiment` object to merge curated variable columns or other clinical variables that would like to be added. It is recommended that the user run the scripts in the `MultiAssayExperiment-TCGA` repository that build the "enhanced" type of data but not necessary if using different clinical data. Please see the repository's README for more information.

Usage

```
mergeColData(MultiAssayExperiment, colData)
```

Arguments

`MultiAssayExperiment`	A MultiAssayExperiment object
`colData`	A `DataFrame` or data.frame to merge with clinical data in the MultiAssayExperiment object

Value

A `MultiAssayExperiment` object

Examples

```
library(MultiAssayExperiment)

mergeColData(MultiAssayExperiment(), S4Vectors::DataFrame())
```

`mirToRanges` *Convert SummarizedExperiment elements with microRNA to RangedSummarizedExperiment*

Description

Convert SummarizedExperiment elements with microRNA to RangedSummarizedExperiment

Usage

```
mirToRanges(obj, keep = FALSE)
```

Arguments

`obj`	A MultiAssayExperiment object obtained from curatedTCGAData
`keep`	If FALSE (default), remove the SummarizedExperiment objects that have been converted to RangedSummarizedExperiment

Value

a MultiAssayExperiment where any of the original SummarizedExperiment containing gene symbols as rownames have been replaced or supplemented by a RangedSummarizedExperiment for miR that could be mapped to GRanges, and another SummarizedExperiment for miR that could not be mapped to GRanges.

Author(s)

L. Waldron

See Also

symbolsToRanges

Examples

```
library(curatedTCGAData)

accmae <- curatedTCGAData("ACC", "miRNASeqGene", dry.run = FALSE)

mirToRanges(accmae)
```

`qreduceTCGA` *Simplify curatedTCGAData objects by replacing RaggedExperiment objects*

Description

Simplify curatedTCGAData objects by replacing RaggedExperiment objects

Usage

```
qreduceTCGA(obj, keep = FALSE, suffix = "_simplified")
```

Arguments

`obj`	A MultiAssayExperiment object obtained from curatedTCGAData
`keep`	logical (default FALSE) whether to remove the original RaggedExperiment objects from the returned MultiAssayExperiment
`suffix`	character (default "_simplified") A character string to append to the newly modified assay.

Details

Relies on TxDb.Hsapiens.UCSC.hg19.knownGene and org.Hs.eg.db to map to hg19 NCBI build.

Value

A MultiAssayExperiment object with RaggedExperiments converted to RangedSummarizedExperiment with rows corresponding to gene symbol.

"Mutations" objects become a genes x patients RangedSummarizedExperiment containing 1 if there is a non-silent mutation somewhere in the gene, and 0 otherwise. "CNA" and "CNV" segmented copy number are reduced using a weighted mean in the rare cases of overlapping (non-disjoint) copy number regions.

Author(s)

L. Waldron

Examples

```
library(curatedTCGAData)
library(GenomeInfoDb)

accmae <-
    curatedTCGAData("ACC", c("CNASNP", "Mutation"), dry.run = FALSE)

## Update build to "hg19"
genome(accmae[["ACC_Mutation-20160128"]]) <-
    vapply(genome(accmae[["ACC_Mutation-20160128"]]),
    translateBuild, character(1L))

qreduceTCGA(accmae)
```

---

sampleTypes *Barcode Sample Type Table*

Description

A dataset that contains the mappings for sample codes in the TCGA barcodes.

Usage

```
sampleTypes
```

Format

A data frame with 19 rows and 3 variables:

Code Two digit code number found in the barcode

Definition Long name for the sample type

Short.Letter.Code Letter code for the sample type

Value

The TCGA 'sampleTypes' table

Source

https://gdc.cancer.gov/resources-tcga-users/tcga-code-tables/sample-type-codes

simplifyTCGA *All-in-one simplification of curatedTCGAData objects*

Description

All-in-one simplification of curatedTCGAData objects

Usage

```
simplifyTCGA(obj, keep = FALSE)
```

Arguments

obj	A MultiAssayExperiment from curatedTCGAData
keep	If FALSE (default), remove the original MultiAssayExperiment elements that have simplified versions in the output.

Value

a MultiAssayExperiment with any gene expression, miRNA, copy number, and mutations converted to RangedSummarizedExperiment objects

Author(s)

L. Waldron

See Also

mirToRanges, symbolsToRanges, qreduceTCGA

Examples

```
library(curatedTCGAData)
library(GenomeInfoDb)

accmae <- curatedTCGAData("ACC",
    c("CNASNP", "Mutation", "miRNASeqGene", "GISTICT"),
    dry.run = FALSE)

rex <- accmae[["ACC_Mutation-20160128"]]

## Translate build to "hg19"
tgenome <- vapply(genome(rex), translateBuild, character(1L))
genome(rex) <- tgenome

accmae[["ACC_Mutation-20160128"]] <- rex

simplifyTCGA(accmae)
```

`symbolsToRanges` *Convert SummarizedExperiment elements with gene symbols to RangedSummarizedExperiment*

Description

Convert SummarizedExperiment elements with gene symbols to RangedSummarizedExperiment

Usage

```
symbolsToRanges(obj, keep = FALSE)
```

Arguments

`obj`	A MultiAssayExperiment object obtained from curatedTCGAData
`keep`	If FALSE (default), remove the SummarizedExperiment objects that have been converted to RangedSummarizedExperiment

Details

Any SummarizedExperiment elements with gene symbols as rownames will have ranges added. Symbols where ranges can't be found are put in a new SummarizedExperiment.

Value

a MultiAssayExperiment where any of the original SummarizedExperiment containing gene symbols as rownames have been replaced or supplemented by a RangedSummarizedExperiment for miR that could be mapped to GRanges, and another SummarizedExperiment for miR that could not be mapped to GRanges.

Author(s)

L. Waldron

See Also

mirToRanges

Examples

```
library(MultiAssayExperiment)

data(miniACC)

symbolsToRanges(miniACC)
```

TCGAbarcode *Parse data from TCGA barcode*

Description

This function returns the specified snippet of information obtained from the TCGA barcode.

Usage

```
TCGAbarcode(barcodes, participant = TRUE, sample = FALSE,
  portion = FALSE, plate = FALSE, center = FALSE, index = NULL)
```

Arguments

barcodes	A character vector of TCGA barcodes
participant	Logical (default TRUE) participant identifier chunk
sample	Logical (default FALSE) includes the numeric sample code of the barcode and the vial letter
portion	Logical (default FALSE) includes the portion and analyte codes of the barcode
plate	Logical (default FALSE) returns the plate value
center	Logical (default FALSE) returns a matrix with the plate and center codes
index	A numerical vector of TCGA barcode positions desired when split by the delimiter (i.e., hyphen '-')

Value

A character vector or data matrix of TCGA barcode information

Author(s)

M. Ramos

Examples

```
barcodes <- c("TCGA-B0-5117-11A-01D-1421-08",
"TCGA-B0-5094-11A-01D-1421-08",
"TCGA-E9-A295-10A-01D-A16D-09")

## Patient identifiers
TCGAbarcode(barcodes)

## Sample identifiers
TCGAbarcode(barcodes, sample = TRUE)
```

TCGAbiospec *Extract biospecimen data from the TCGA barcode*

Description

This function uses the full TCGA barcode to return a data frame of the data pertinent to laboratory variables such as vials, portions, analytes, plates and the center.

Usage

```
TCGAbiospec(barcodes)
```

Arguments

barcodes	A character vector of TCGA barcodes

Value

A dataframe with sample type, sample code, portion, plate, and center columns.

Author(s)

M. Ramos

Examples

```
example("TCGAbarcode")
TCGAbiospec(barcodes)
```

TCGAsampleSelect *Select samples from barcodes from lookup table*

Description

The TCGA barcode contains several pieces of information which can be parsed by the TCGAbarcode function. To select a specific type of sample, enter the appropriate sampleCode argument from the lookup table. See lookup table in data("sampleTypes").

Usage

```
TCGAsampleSelect(barcodes, sampleCode)
```

Arguments

barcodes	Standard TCGA barcodes containing patient identifiers, sample, portion, plate, center codes.
sampleCode	Either a character or numeric vector of length one. See the sampleType dataset.

Value

A logical vector of the same length as 'barcodes' indicating matches

Examples

```
example("TCGAbarcode")
TCGAsampleSelect(barcodes, 11)
```

`translateBuild` *Utilities for working with build numbers*

Description

A few functions are available to search for build versions, either from NCBI or UCSC.

- `translateBuild`: translates between UCSC and NCBI build versions
- `extractBuild`: use grep patterns to find the first build within the string input
- `uniformBuilds`: replace build occurrences below a threshold level of occurence with the alternative build

Usage

```
translateBuild(from, to = "UCSC")

extractBuild(string, build = c("UCSC", "NCBI"))

uniformBuilds(builds, cutoff = 0.2)
```

Arguments

`from`	A build version name
`to`	The name of the desired version
`string`	A single character string
`build`	A vector of build version names (default UCSC, NCBI)
`builds`	A character vector of builds
`cutoff`	A threshold value for translating builds below the threshold

Examples

```
translateBuild("GRCh35", "UCSC")


extractBuild(
"SCENA_p_TCGAb29and30_SNP_N_GenomeWideSNP_6_G05_569110.nocnv_grch38.seg.txt"
)


buildvec <- rep(c("GRCh37", "hg19"), times = c(5, 1))
uniformBuilds(buildvec)
```

Index