Package: TCGAutils (via r-universe)

October 30, 2026

Title TCGA utility functions for data management

Version 1.32.2

Date 2025-12-11

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. Exported functions include those that import data from flat files into Bioconductor objects, convert row annotations, and identifier translation via the GDC API.

Depends R (>= 4.5.0)

Imports AnnotationDbi, BiocGenerics, BiocBaseUtils, GenomeInfoDb, GenomicFeatures, GenomicRanges, GenomicDataCommons, glue, IRanges, methods, MultiAssayExperiment, RaggedExperiment, rvest, S4Vectors, Seqinfo, stats, stringr, SummarizedExperiment, utils, xml2

Suggests AnnotationHub, Bioc.gff, BiocFileCache, BiocStyle, curatedTCGAData, ComplexHeatmap, devtools, dplyr, httr, IlluminaHumanMethylation450kanno.ilmn12.hg19, impute, knitr, magrittr, miRNAmeConverter, org.Hs.eg.db, RColorBrewer, readr, rmarkdown, RTCGAToolbox, rtracklayer, R.utils, testthat, TxDb.Hsapiens.UCSC.hg18.knownGene, TxDb.Hsapiens.UCSC.hg19.knownGene

License Artistic-2.0

Roxygen list(markdown = TRUE)

Encoding UTF-8

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

biocViews Software, WorkflowStep, Preprocessing, DataImport

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/pak/sysreqs make libbz2-dev libicu-dev liblzma-dev libpng-dev libxml2-dev libssl-dev libx11-dev xz-utils zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-07-02 19:25:56 UTC

RemoteUrl https://github.com/bioc/TCGAutils

RemoteRef RELEASE_3_23

RemoteSha 85081507639d0cd7c6ce1c6db524798143b37a48

Contents

TCGAutils-package 2
builds 3
clinicalNames 5
curatedTCGAData-helpers 5
diseaseCodes 7
findGRangesCols 8
generateMap 9
getFileName 10
ID-translation 11
imputeAssay 13
makeGRangesListFromCopyNumber 14
makeGRangesListFromExonFiles 16
mergeColData 17
oncoPrintTCGA 18
sampleTypes 19
simplifyTCGA 19
TCGAbarcode 21
TCGAbiospec 23
TCGAprimaryTumors 23
TCGAsampleSelect 24
trimColData 25

Index **26**

`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@sph.cuny.edu> (ORCID)

Authors:

- Marcel Ramos <marcel.ramos@sph.cuny.edu> (ORCID)
- Lucas Schiffer
- Levi Waldron

Other contributors:

- Sean Davis [contributor]
- NCI (GrantNo.: U24CA289073) [funder]

See Also

Useful links:

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

---

`builds` *Utilities for working with* HUMAN *genome builds*

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
- `correctBuild`: Ensure that the build annotation is correct based on the NCBI/UCSC website. If not, use `translateBuild` with the indicated 'style' input
- `isCorrect`: Check to see if the build is exactly as annotated

Usage

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

correctBuild(build, style = c("UCSC", "NCBI"))

isCorrect(build, style = c("UCSC", "NCBI"))

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

uniformBuilds(builds, cutoff = 0.2, na = c("", "NA"))
```

Arguments

from	character() A vector of build versions typically from genome() (e.g., "37"). The build vector must be homogenous (i.e., length(unique(x)) == 1L).
to	character(1) The name of the desired build version (either "UCSC" or "NCBI"; default: "UCSC")
build	A vector of build version names (default UCSC, NCBI)
style	character(1) The annotation style, either 'UCSC' or 'NCBI'
string	A single character string
builds	A character vector of builds
cutoff	numeric(1L) An inclusive threshold tolerance value for missing values and translating builds that are below the threshold
na	character() The values to be considered as missing (default: c("", "NA"))

Details

The correctBuild function takes the input and ensures that the style specified matches the input. Otherwise, it will return the correct style for use with seqlevelsStyle. Currently, the function does not support patched builds (e.g., 'GRCh38.p13') Build names are taken from the website: https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.26/

Value

```
translateBuild: A character vector of translated genome builds

extractBuild: A character string of the build information available

uniformBuilds: A character vector of builds where all builds are
    identical `identical(length(unique(build)), 1L)`

correctBuild: A character string of the 'corrected' build name

isCorrect: A logical indicating if the build is exactly as annotated
```

Examples

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


correctBuild("grch38", "NCBI")
correctBuild("hg19", "NCBI")


isCorrect("GRCh38", "NCBI")

isCorrect("hg19", "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)

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

`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

```
data("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`.

Usage

```
getSubtypeMap(multiassayexperiment)

getClinicalNames(diseaseCode)

TCGAsplitAssays(multiassayexperiment, sampleCodes = NULL, exclusive = FALSE)

sampleTables(multiassayexperiment, vial = FALSE)
```

Arguments

multiassayexperiment	A MultiAssayExperiment object
diseaseCode	A TCGA cancer code (e.g., "BRCA")
sampleCodes	character (default NULL) A string of sample type codes (refer to data(sampleTypes); TCGAsplitAssays section)
exclusive	logical (default FALSE) Whether to return only assays that contain all codes in sampleCodes
vial	(logical default FALSE) whether to display vials in the table output

Details

Note that for getSubtypeMap, the column of in-data variable names may need to go through make.names to be found in the colData of the MultiAssayExperiment.

Value

- getSubtypeMap: A data.frame with explanatory names and their in-data variable names. They may not be present for all cancer types.
- getClinicalNames: A vector of common variable names that may be found across several cancer disease codes.

getSubtypeMap

provides a two column data.frame with interpreted names and in-data variable names. 'Name' usually refers to the colData row names a.k.a. the patientID.

getClinicalNames

provides a vector of common variable names that exist in the colData DataFrame of a curatedTCGAData MultiAssayExperiment object. These variables are directly obtained from the BroadFirehose clinical data (downloaded with getFirehoseData) and tend to be present across cancer disease codes.

TCGAsplitAssays

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, all assays will be split by samples present in the data.

sampleTables

Display all the available samples in each of the assays

Examples

```
library(curatedTCGAData)

gbm <- curatedTCGAData("GBM", c("RPPA*", "CNA*"), version = "2.0.1", FALSE)

getSubtypeMap(gbm)

sampleTables(gbm)

TCGAsplitAssays(gbm, c("01", "10"))

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

```
data("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))
```

`getFileName` *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

```
getFileName(
  disease,
  runDate = "20160128",
  dataType = c("CNASNP", "CNVSNP", "CNAseq", "CNACGH", "Mutation")
)
```

Arguments

`disease`	The TCGA cancer disease code, e.g., "COAD"
`runDate`	The single `string` used in the `getFirehoseData` function (default "20160128")
`dataType`	A single character vector (default "CNASNP") indicating the data type for which to get the source file name

Value

A single `character` file name

Examples

```
getFileName("COAD", dataType = "CNASNP")
```

`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 `from_type` argument for details. All identifiers are converted to lower case.

Usage

```
UUIDtoBarcode(id_vector, from_type = c("case_id", "file_id", "aliquot_ids"))

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

barcodeToUUID(barcodes)

filenameToBarcode(filenames, slides)

UUIDhistory(id, endpoint = .HISTORY_ENDPOINT)
```

Arguments

`id_vector`	character() A vector of UUIDs corresponding to either files or cases (default assumes case_ids)
`from_type`	character(1) Either `case_id` or `file_id` indicating the type of `id_vector` entered (default `"case_id"`)
`to_type`	character(1) The desired UUID type to obtain, can either be `"case_id"` (default) or `"file_id"`
`barcodes`	character() A vector of TCGA barcodes
`filenames`	`character()` A vector of file names usually obtained from a `GenomicDataCommons` query
`slides`	`logical(1L)` **DEFUNCT**: No longer used. See details.
`id`	character(1) A UUID whose history of versions is sought
`endpoint`	character(1) Generally a constant pertaining to the location of the history api endpoint. This argument rarely needs to change.

Details

Based on the file UUID supplied, the appropriate entity_id (TCGA barcode) is returned. In previous versions of the package, the 'end_point' parameter would require the user to specify what type of barcode needed. This is no longer supported as `entity_id` returns the appropriate one.

Slides are identified by the `filenames` input by searching for the `.svs` extension. Slide queries can only be done when **all** `filenames` inputs are slide file names.

Value

Generally, a `data.frame` of identifier mappings

UUIDhistory: A `data.frame` containting a list of associated UUIDs for the given input along with `file_change` status, `data_release` versions, etc.

Author(s)

Sean Davis, M. Ramos

Examples

```
## Translate UUIDs >> TCGA Barcode

uuids <- c("b4bce3ff-7fdc-4849-880b-56f2b348ceac",
"5ca9fa79-53bc-4e91-82cd-5715038ee23e",
"b7c3e5ad-4ffc-4fc4-acbf-1dfcbd2e5382")

UUIDtoBarcode(uuids, from_type = "file_id")

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

UUIDtoBarcode("d85d8a17-8aea-49d3-8a03-8f13141c163b", "aliquot_ids")

## Translate file UUIDs >> case UUIDs

uuids <- c("b4bce3ff-7fdc-4849-880b-56f2b348ceac",
"5ca9fa79-53bc-4e91-82cd-5715038ee23e",
"b7c3e5ad-4ffc-4fc4-acbf-1dfcbd2e5382")

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)

### Query CNV data and get file names

cnv <- files() |>
    filter(
        ~ cases.project.project_id == "TCGA-COAD" &
        data_category == "Copy Number Variation" &
        data_type == "Copy Number Segment"
    ) |>
    results(size = 6)

filenameToBarcode(cnv$file_name)

### Query slides data and get file names

slides <- files() |>
    filter(
        ~ cases.project.project_id == "TCGA-BRCA" &
        cases.samples.sample_type == "Primary Tumor" &
        data_type == "Slide Image" &
        experimental_strategy == "Diagnostic Slide"
    ) |>
    results(size = 3)

filenameToBarcode(slides$file_name)

## Get the version history of a BAM file in TCGA-KIRC
UUIDhistory("0001801b-54b0-4551-8d7a-d66fb59429bf")
```

`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

A MultiAssayExperiment with imputed assays values

Examples

```
example(getSubtypeMap)

## convert data to matrix and add as experiment
gbm <-
  c(gbm, RPPA_matrix = data.matrix(assay(gbm[["GBM_RPPAArray-20160128"]])))

imputeAssay(gbm, i = "RPPA_matrix")
```

---

makeGRangesListFromCopyNumber

Make a GRangesList from TCGA Copy Number data

Description

makeGRangesListFromCopyNumber allows the user to convert objects of class data.frame or S4Vectors::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 GenomicRanges::makeGRangesListFromDataFrame

Value

A GRangesList class object

Examples

```
library(GenomicDataCommons)

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

fname <- gdcdata(manif$id)

UUIDtoBarcode(names(fname), from_type = "file_id")

cndata <- read.delim(fname[[1L]])

makeGRangesListFromCopyNumber(
    df = cndata,
    split.field = "GDC_Aliquot",
    keep.extra.columns = TRUE
)
```

makeGRangesListFromExonFiles
Read exon-level expression 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) file paths and represents such data as a GRangesList. The data files can be downloaded via the Genomic Data Commons (GDC) Legacy Archive.

Usage

```
makeGRangesListFromExonFiles(
  filepaths,
  sampleNames = NULL,
  fileNames = basename(filepaths),
  getBarcodes = TRUE,
  rangesColumn = "exon",
  nrows = Inf
)
```

Arguments

filepaths	character() vector of file paths containing TCGA exon data usually obtained from the GDC
sampleNames	character() vector of TCGA barcodes to be used as names for the GRangesList output (default NULL)
fileNames	character() vector of file names as downloaded from the Genomic Data Commons Legacy archive (default basename(filepaths))
getBarcodes	logical(1). Whether to query the GDC API with the filenameToBarcode and obtain the TCGA barcodes from the file names (default TRUE); see details.
rangesColumn	character(1). The name of the column in the data containing the ranges information (default "exon"); see details.
nrows	numeric(1). The number of rows to return from each of the files read in (all rows by default; default Inf)

Details

The rangesColumn name in the GDC data files is usually "exon" but can be changed with the rangesColumn argument, if different. To avoid programmatically obtaining TCGA barcodes from the GDC API, set the getBarcodes to FALSE. When getBarcodes is set to FALSE, the file names are used to name the elements of the GRangesList output.

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")
```

---

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())
```

oncoPrintTCGA *OncoPrint for TCGA Mutation Assays*

Description

OncoPrint for TCGA Mutation Assays

Usage

```
oncoPrintTCGA(
  multiassayexperiment,
  matchassay = "*_Mutation-*",
  variantCol = "Variant_Classification",
  brewerPal = "Set3",
  ntop = 25,
  incl.thresh = 0.01,
  rowcol = "Hugo_Symbol"
)
```

Arguments

multiassayexperiment	A `MultiAssayExperiment`, usually from `curatedTCGAData`
matchassay	character(1) The name of the assay containing mutation data, this can be a pattern (e.g., "*_Mutation-*", the default)
variantCol	character(1) The name of the metadata column containing the mutation categories, usually "Variant_Classification" in TCGA
brewerPal	character(1) The name of the `RColorBrewer::brewer.pal` palette, (default: "Set3")
ntop	integer(1) The number of the top N genes for displaying based on per-sample mutation frequency
incl.thresh	double(1) The inclusion threshold for empirical mutations, mutations less frequent than this value will not be included
rowcol	character(1) The name of the column in the metadata to annotate the rows with either "Hugo_Symbol" (default) or

Value

An oncoPrint plot of mutations

Examples

```
library(curatedTCGAData)

acc <- curatedTCGAData("ACC", "Mutation", version = "1.1.38", FALSE)
```

```
oncoPrintTCGA(acc)
```

`sampleTypes` *Barcode Sample Type Table*

Description

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

Usage

```
data("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` *Functions to convert rows annotations to ranges and RaggedExperiment to RangedSummarizedExperiment*

Description

This group of functions will convert row annotations as either gene symbols or miRNA symbols to row ranges based on database resources 'TxDB' and 'org.Hs' packages. It will also simplify the representation of RaggedExperiment objects to RangedSummarizedExperiment.

Usage

```
simplifyTCGA(obj, keep.assay = FALSE, unmapped = TRUE)

symbolsToRanges(obj, keep.assay = FALSE, unmapped = TRUE)

mirToRanges(obj, keep.assay = FALSE, unmapped = TRUE)

CpGtoRanges(obj, keep.assay = FALSE, unmapped = TRUE)

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

Arguments

obj	A MultiAssayExperiment object obtained from curatedTCGAData
keep.assay	logical (default FALSE) Whether to keep the SummarizedExperiment assays that have been converted to RangedSummarizedExperiment
unmapped	logical (default TRUE) Include an assay of data that was not able to be mapped in reference database
suffix	character (default "_simplified") A character string to append to the newly modified assay for qreduceTCGA.

Details

The original SummarizedExperiment containing either gene symbol or miR annotations is replaced or supplemented by a RangedSummarizedExperiment for those that could be mapped to GRanges, and optionally another SummarizedExperiment for annotations that could not be mapped to GRanges.

Value

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

qreduceTCGA

Using TxDb.Hsapiens.UCSC.hg19.knownGene as the reference, qreduceTCGA reduces the data by applying either the weightedmean or nonsilent function (see below) to non-mutation or mutation data, respectively. Internally, it uses RaggedExperiment::qreduceAssay() to reduce the ranges to the gene-level.

qreduceTCGA will update genome(x) based on the NCBI reference annotation which includes the patch number, e.g., GRCh37.p14, as provided by the seqlevelsStyle setter, seqlevelsStyle(gn) <- "NCBI". qreduceTCGA uses the NCBI genome annotation as the default reference.

```
nonsilent <- function(scores, ranges, qranges)
    any(scores != "Silent")
```

RaggedExperiment mutation objects become a genes by patients RangedSummarizedExperiment object containing '1' if there is a non-silent mutation somewhere in the gene, and '0' otherwise as obtained from the Variant_Classification column in the data.

```
weightedmean <- function(scores, ranges, qranges) {
    isects <- GenomicRanges::pintersect(ranges, qranges)
    sum(scores * BiocGenerics::width(isects)) /
        sum(BiocGenerics::width(isects))
}
```

"CNA" and "CNV" segmented copy number are reduced using a weighted mean in the rare cases of overlapping (non-disjoint) copy number regions.

These functions rely on TxDb.Hsapiens.UCSC.hg19.knownGene and org.Hs.eg.db to map to the 'hg19' NCBI build. Use the liftOver procedure for datasets that are provided against a different reference genome (usually 'hg18'). See an example in the vignette.

Author(s)

L. Waldron, M. Ramos

Examples

```
library(curatedTCGAData)
library(GenomeInfoDb)

accmae <- curatedTCGAData(
    diseaseCode = "ACC",
    assays = c("CNASNP", "Mutation", "miRNASeqGene", "GISTICT"),
    version = "1.1.38",
    dry.run = FALSE
)

## update genome annotation
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)
```

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	An optional numeric vector indicating barcode positions when split by the delimiter (i.e., hyphen '-'). For example, an index of `c(1, 2)` corresponds to 'TCGA-ZZ' in `TCGA-ZZ-A1A1`.

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)
```

TCGAprimaryTumors *Select primary tumors from TCGA datasets*

Description

Tumor selection is decided using the sampleTypes data. For 'LAML' datasets, the primary tumor code used is "03" otherwise, "01" is used.

Usage

```
TCGAprimaryTumors(multiassayexperiment)
```

Arguments

multiassayexperiment	A MultiAssayExperiment with TCGA data as obtained from curatedTCGAData::curatedTCGAData()

Value

A `MultiAssayExperiment` containing only primary tumor samples

Examples

```
example(getSubtypeMap)

TCGAprimaryTumors(gbm)
```

`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")`. Barcode inputs can be a character vector or a CharacterList object.

Usage

```
TCGAsampleSelect(barcodes, sampleCodes)
```

Arguments

`barcodes`	Either a TCGA barcode vector or CharacterList containing patient identifiers, sample, portion, plate, and center codes.
`sampleCodes`	Either a character or numeric vector of TCGA sample codes. See the `sampleType` dataset.

Value

A logical vector or LogicalList of the same length as 'barcodes' indicating sample type matches

Examples

```
example("TCGAbarcode")
TCGAsampleSelect(barcodes, c(11, 01))
```

`trimColData` *Minimize the number of variables in colData*

Description

This function removes variables that have a high number of missing data and contain keywords.

Usage

```
trimColData(
  multiassayexperiment,
  maxNAfrac = 0.2,
  keystring = c("portion", "analyte")
)
```

Arguments

`multiassayexperiment`	A `MultiAssayExperiment` object with `colData`
`maxNAfrac`	(numeric default 0.2) A decimal between 0 and 1 to indicate the amount of NA values allowed per column
`keystring`	(character) A vector of keywords to match and remove variables

Value

A `MultiAssayExperiment` object

Examples

```
example(getSubtypeMap)

(gbm_trimmed <- trimColData(gbm))

head(colData(gbm_trimmed))[1:5]
```

Index

∗ **datasets**
clinicalNames, 5
diseaseCodes, 7
sampleTypes, 19

barcodeToUUID *(*ID-translation*)*, 11
builds, 3

CharacterList, *5*, *24*
clinicalNames, 5
correctBuild *(*builds*)*, 3
CpGtoRanges *(*simplifyTCGA*)*, 19
curatedTCGAData-helpers, 5
curatedTCGAData::curatedTCGAData(), *23*

diseaseCodes, 7

extractBuild *(*builds*)*, 3

filenameToBarcode *(*ID-translation*)*, 11
findGRangesCols, 8

generateMap, 9
GenomicRanges::makeGRangesListFromDataFrame, *15*
getClinicalNames *(*curatedTCGAData-helpers*)*, 5
getFileName, 10
getFirehoseData, *5*, *6*, *10*
getSubtypeMap *(*curatedTCGAData-helpers*)*, 5
GRanges, *20*
GRangesList, *2*, *14–16*

ID-translation, 11
impute::impute.knn, *14*
imputeAssay, 13
isCorrect *(*builds*)*, 3

LogicalList, *24*

makeGRangesListFromCopyNumber, 14
makeGRangesListFromExonFiles, 16
mergeColData, 17
mirToRanges *(*simplifyTCGA*)*, 19
MultiAssayExperiment, *2*, *6*, *17*, *20*, *23*, *25*

oncoPrintTCGA, 18

qreduceTCGA *(*simplifyTCGA*)*, 19

RaggedExperiment, *19*
RaggedExperiment::qreduceAssay(), *20*
RangedSummarizedExperiment, *19*, *20*

S4Vectors::DataFrame, *14*
sampleTables *(*curatedTCGAData-helpers*)*, 5
sampleTypes, 19
simplifyTCGA, 19
SummarizedExperiment, *2*, *20*
symbolsToRanges *(*simplifyTCGA*)*, 19

TCGAbarcode, 21, *24*
TCGAbiospec, 23
TCGAprimaryTumors, 23
TCGAsampleSelect, 24
TCGAsplitAssays *(*curatedTCGAData-helpers*)*, 5
TCGAutils *(*TCGAutils-package*)*, 2
TCGAutils-package, 2
translateBuild *(*builds*)*, 3
trimColData, 25

uniformBuilds *(*builds*)*, 3
UUIDhistory *(*ID-translation*)*, 11
UUIDtoBarcode *(*ID-translation*)*, 11
UUIDtoUUID *(*ID-translation*)*, 11