

# Package ‘BiocOncoTK’

March 24, 2019

**Title** Bioconductor components for general cancer genomics

**Description** Provide a central interface to various tools for genome-scale analysis of cancer studies.

**Version** 1.3.6

**Author** Vince Carey

**Suggests** knitr, dbplyr, DBI, org.Hs.eg.db, MultiAssayExperiment, BiocStyle, ontoProc, ontologyPlot, pogos, GenomeInfoDb, restfulSE (>= 1.3.7), BiocFileCache, TxDb.Hsapiens.UCSC.hg19.knownGene, Biobase, TxDb.Hsapiens.UCSC.hg18.knownGene, reshape2, testthat, AnnotationDbi, FDb.InfiniumMethylation.hg19, EnsDb.Hsapiens.v75

**Imports** ComplexHeatmap, S4Vectors, bigrquery, shiny, stats, httr, rjson, dplyr, magrittr, grid, utils, DT, GenomicRanges, IRanges, ggplot2, SummarizedExperiment, DBI, GenomicFeatures

**Depends** R (>= 3.5.0), methods

**Maintainer** VJ Carey <stvjc@channing.harvard.edu>

**License** Artistic-2.0

**LazyLoad** yes

**LazyData** yes

**biocViews** CopyNumberVariation, CpGIsland, DNAMethylation, GeneExpression, GeneticVariability, SNP, Transcription, ImmunoOncology

**VignetteBuilder** knitr

**RoxygenNote** 6.1.1

**Encoding** UTF-8

**git\_url** <https://git.bioconductor.org/packages/BiocOncoTK>

**git\_branch** master

**git\_last\_commit** 1d6a75a

**git\_last\_commit\_date** 2019-03-11

**Date/Publication** 2019-03-24

**R topics documented:**

annotTabs . . . . .	2
bindMSI . . . . .	3
brcaMAE . . . . .	4
buildPancanSE . . . . .	5
CCLC_DRUG_BROAD . . . . .	6
cell_70138 . . . . .	6
clueDemos . . . . .	7
clueServiceNames . . . . .	7
darmGBMcls . . . . .	8
dingMSI . . . . .	8
featIDMapper . . . . .	9
fireMSI . . . . .	9
ggFeatDens . . . . .	10
ggFeatureSegs . . . . .	10
ggMutDens . . . . .	11
icd10_c . . . . .	12
loadPatel . . . . .	12
load_ccleNRAS . . . . .	13
mc3toGR . . . . .	14
oncoPrintISB . . . . .	14
pancan.clin.varnames . . . . .	15
pancan_app . . . . .	15
pancan_BQ . . . . .	16
pancan_clinicalTabVarnames . . . . .	17
pancan_longname . . . . .	17
pancan_sampTypeMap . . . . .	18
pancan_tabulate . . . . .	18
pertClasses . . . . .	19
pert_70138 . . . . .	19
query_clue . . . . .	20
replaceRownames . . . . .	20
TcgaMutCounts . . . . .	21
TcgaNIndWithAnyMut . . . . .	21
tumNorSet . . . . .	22
<b>Index</b>	<b>23</b>

---

annotTabs	<i>table names in Annotated pancancer data release</i>
-----------	--

---

**Description**

table names in Annotated pancancer data release

**Usage**

```
annotTabs
```

**Format**

character vector

**Source**

pancancer-atlas in BigQuery

**Examples**

BiocOncoTK::annotTabs

---

bindMSI	<i>bind MSI data to a SummarizedExperiment</i>
---------	--

---

**Description**

bind MSI data to a SummarizedExperiment

**Usage**

```
bindMSI(se, useDing = TRUE, onlyHL = TRUE)
```

**Arguments**

se	SummarizedExperiment instance
useDing	logical(1) if TRUE, use MSI sensor outputs from Ding et al. Cell 2018, otherwise use firehose labelings msi-h,msi-l
onlyHL	logical(1) if TRUE, retain only msi-h, msi-l records; ignored if useDing is TRUE

**Value**

SummarizedExperiment instance with expanded colData, samples limited to those with microsatellite instability values. The additional variable is called 'msiTest' and is numerical if useDing is TRUE and is character (msi-h,l,s) otherwise.

**Note**

This function adds the column msiTest to colData(se). The contents of the column are given by [fireMSI](#). Samples in se that do not correspond to a row of [fireMSI](#) are dropped. If there is already a column named msiTest in colData(se), it is replaced and samples are filtered as described, and a message is given. If none of the samples in se have rows in [fireMSI](#), an error is thrown.

**Examples**

```
bindMSI
```

brcaMAE

*a virtual MultiAssayExperiment for pancancer-atlas BRCA data***Description**

a virtual MultiAssayExperiment for pancancer-atlas BRCA data

**Usage**

```
brcaMAE
```

**Format**

MultiAssayExperiment instance with DelayedArray (BQ3\_Array) assay data

**Note**

Constructed as

```
library(BiocOncoTK)
pcbq = pancan_BQ()
library(restfulSE)
BRCA_mir = pancan_SE(pcbq)
BRCA_mrna = pancan_SE(pcbq,
  assayDataTableName = pancan_longname("rnaseq"),
  assayFeatureName = "Entrez",
  assayValueFieldName = "normalized_count")
BRCA_rppa = pancan_SE(pcbq,
  assayDataTableName = pancan_longname("RPPA"),
  assayFeatureName = "Protein",
  assayValueFieldName = "Value")
BRCA_meth = pancan_SE(pcbq,
  assayDataTableName = pancan_longname("27k")[2],
  assayFeatureName = "ID",
  assayValueFieldName = "Beta")
library(MultiAssayExperiment)
library(dplyr)
library(magrittr)
clinBRCA = pcbq %>% tbl(pancan_longname("clinical")) %>%
  filter(acronym=="BRCA") %>% as.data.frame()
rownames(clinBRCA) = clinBRCA[,2]
clinDF = DataFrame(clinBRCA)
library(MultiAssayExperiment)
brcaMAE = MultiAssayExperiment(
  ExperimentList(rnaseq=BRCA_mrna, meth=BRCA_meth, rppa=BRCA_rppa,
    mirna=BRCA_mir),colData=clinDF)
upsetSamples(brcaMAE) # to view display
```

**Source**

ISB BigQuery pancan-atlas project

**Examples**

```
if (requireNamespace("MultiAssayExperiment"))
  BiocOncoTK::brcaMAE
```

---

buildPancanSE	<i>helper for SummarizedExperiment construction from pancan</i>
---------------	---

---

**Description**

helper for SummarizedExperiment construction from pancan

**Usage**

```
buildPancanSE(bq, acronym = "BLCA", assay = "meth450k",
  sampType = "TP", subjectIDName = "ParticipantBarcode",
  seTransform = force, bindMethRowranges = TRUE,
  featIDMap = featIDMapper())
```

**Arguments**

bq	instance of BigQueryConnection for pancancer-atlas.Annotated Dataset
acronym	character(1) 'cohort' label, e.g., 'BLCA'
assay	character(1) element from names(BiocOncoTK::annotTabs), e.g., 'meth450k'
sampType	character(1) element from BiocOncoTK::pancan_sampTypeMap\$"SampleTypeLetterCode", e.g., 'TP' for Primary solid Tumor samples, or 'TB' for peripheral blood sample from primary blood derived cancer
subjectIDName	character(1) field name for subject identifier
seTransform	a function that accepts a SummarizedExperiment and returns a SummarizedExperiment; useful for feature name remapping, defaults to force (does nothing)
bindMethRowranges	logical(1) if true and assay is meth27k
featIDMap	a named character() vector defining, for each assay type, what field should be used to label features in rownames. or meth450k, annotation from FDb.InfiniumMethylation.hg19 and EnsDb.Hsapiens.v75 is obtained for available features and bound into the rowRanges component of returned object

**Value**

SummarizedExperiment, with metadata on acronym, assay, and sampleType propagated; if the assay is a methylation assay and bindMethRowranges is TRUE, a RangedSummarizedExperiment is returned.

**Note**

Note that pancancer-atlas is distinguished from TCGA by the presence of more sample types. The default type is 'TP' for primary solid tumor. Codes and their interpretations are available in BiocOncoTK::pancan\_sampTypeMap.

**Examples**

```

if (interactive() && Biobase::testBioCConnection()) {
  billco = Sys.getenv("CGC_BILLING")
  if (nchar(billco)>0) {
    bq = pancan_BQ()
    methSE_BLCA = try(buildPancanSE(bq))
    methSE_BLCA
  }
}

```

---

CCLE_DRUG_BROAD	<i>CCLE_DRUG_BROAD: serialization of legacy CCLE 'Drug data' from Broad Institute</i>
-----------------	---

---

**Description**

CCLE\_DRUG\_BROAD: serialization of legacy CCLE 'Drug data' from Broad Institute

**Usage**

```
CCLE_DRUG_BROAD
```

**Format**

S4Vectors DataFrame instance

**Source**

["https://data.broadinstitute.org/ccle\\_legacy\\_data/pharmacological\\_profiling/CCLE\\_NP24.2009\\_Drug\\_data\\_2015.02.24.csv"](https://data.broadinstitute.org/ccle_legacy_data/pharmacological_profiling/CCLE_NP24.2009_Drug_data_2015.02.24.csv)

**Examples**

```

data(CCLE_DRUG_BROAD)
requireNamespace("S4Vectors")
S4Vectors::metadata(CCLE_DRUG_BROAD) # imported using read.csv, stringsAsFactors=FALSE, coerced to DataFrame
head(CCLE_DRUG_BROAD)

```

---

cell_70138	<i>cell_70138: a table with cell-line information from LINCS</i>
------------	--

---

**Description**

cell\_70138: a table with cell-line information from LINCS

**Usage**

```
cell_70138
```

**Format**

data.frame

**Source**

GEO GSE70138 GSE70138\_Broad\_LINCS\_cell\_info\_2017-04-28.txt.gz

**Examples**

```
data(cell_70138)
```

---

clueDemos

*generate lists to generate clue API queries*

---

**Description**

generate lists to generate clue API queries

**Usage**

```
clueDemos()
```

**Value**

a list of lists of strings with 'where' and substructure as appropriate

**Note**

These are converted to JSON (

**Examples**

```
clueDemos()
```

---

clueServiceNames

*Provide names of some clue.io services for which examples are available in this package.*

---

**Description**

Provide names of some clue.io services for which examples are available in this package.

**Usage**

```
clueServiceNames()
```

**Value**

a character vector of service names

**Note**

See <https://clue.io/api>.

**Examples**

```
clueServiceNames()
```

---

darmGBMcls	<i>Data in count_1stpm format from Darmanis 2017 (PMC 5810554) single cell RNA-seq in GBM</i>
------------	---

---

**Description**

Data in count\_1stpm format from Darmanis 2017 (PMC 5810554) single cell RNA-seq in GBM

**Usage**

```
darmGBMcls
```

**Format**

SummarizedExperiment with HDF Object store back end

**Note**

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5810554/> is the main source article.

**Source**

[http://imlspenticton.uzh.ch/robinson\\_lab/conquer/data-mae/GSE84465.rds](http://imlspenticton.uzh.ch/robinson_lab/conquer/data-mae/GSE84465.rds)

**Examples**

```
BiocOncoTK::darmGBMcls
```

---

dingMSI	<i>microsatellite instability data in TCGA, collected from Ding et al. Cell 173(2) 2018.</i>
---------	--

---

**Description**

microsatellite instability data in TCGA, collected from Ding et al. Cell 173(2) 2018.

**Usage**

```
dingMSI
```

**Format**

DataFrame



**Source**

<https://www.cell.com/cms/10.1016/j.cell.2018.03.033/attachment/0ac495ba-3578-41cf-8fb1-94487f55mmc5.xlsx> retrieved 9/17/2018.

**Examples**

```
str(BiocOncoTK::dingMSI)
```

---

featIDMapper	<i>define assay-specific feature names in a character vector</i>
--------------	--

---

**Description**

define assay-specific feature names in a character vector

**Usage**

```
featIDMapper()
```

**Note**

We may want to use Symbol instead of Entrez when retrieving expression data. The value of this function is supplied as a default for `buildPancanSE`'s `featIDMap` parameter, and alternatives can be selected by passing similarly named vectors in `featIDMap`.

**Examples**

```
featIDMapper()
```

---

fireMSI	<i>microsatellite instability data in TCGA, collected from curatedTCGAData</i>
---------	--

---

**Description**

microsatellite instability data in TCGA, collected from `curatedTCGAData`

**Usage**

```
fireMSI
```

**Format**

```
DataFrame
```

**Source**

firehose via `curatedTCGAData`; see `metadata(BiocOncoTK::fireMSI)`

**Examples**

```
str(S4Vectors::metadata(BiocOncoTK::fireMSI))
```

---

ggFeatDens	<i>create ggplot for density of starts of a GRanges in an interval</i>
------------	--

---

**Description**

create ggplot for density of starts of a GRanges in an interval

**Usage**

```
ggFeatDens(gr, mcolvbl, chrname = "chr15", start = 20450000,
  end = 20730000, binwidth.in = 5000, basicfilt = function(data)
  dplyr::filter(data, Consequence == "non_coding_transcript_exon_variant"),
  ylab.in = "feature\ndensity", slstyle = "UCSC")
```

**Arguments**

gr	GRanges instance of interest
mcolvbl	character(1) mcols(gr) has this variable that will be used to specify different groups for computing/colouring the density traces
chrname	character(1) chromosome/seqname
start	numeric(1) start of interval
end	numeric(1) end of interval
binwidth.in	numeric(1) for geom_freqpoly binwidth setting
basicfilt	a dplyr::filter operation, defaulting to select non-coding variants in mc3 MAF
ylab.in	character(1) label for y axis
slstyle	character(1) for GenomeInfoDb::seqlevelsStyle

**Value**

ggplot instance

**Examples**

```
ggFeatDens
```

---

ggFeatureSegs	<i>generate a ggplot of segments of gene-like regions</i>
---------------	---

---

**Description**

generate a ggplot of segments of gene-like regions

**Usage**

```
ggFeatureSegs(chrname = "chr15", start = 20450000, end = 20730000,
  db = EnsDb.Hsapiens.v75::EnsDb.Hsapiens.v75, slstyle = "UCSC",
  ylab.in = "ensembl\nnoncoding")
```

**Arguments**

chrname	character(1) chromosome tag
start	numeric(1) start of interval
end	numeric(1) end of interval
db	EnsDb instance for example
slstyle	character(1) tag for seqlevelsStyle
ylab.in	character(1) for use as y axis tag

**Value**

ggplot instance

**Note**

Most annotation is turned off with `element_blank()`

**Examples**

```
ggFeatureSegs
```

---

ggMutDens	<i>make a ggplot with density traces of mutations per base pair, for 'most mutated' tumor types in a given interval</i>
-----------	---

---

**Description**

make a ggplot with density traces of mutations per base pair, for 'most mutated' tumor types in a given interval

**Usage**

```
ggMutDens(bq, basicfilt = function(data) dplyr::filter(data, Consequence
  == "non_coding_transcript_exon_variant"), chrname = "15",
  start = 20450000, end = 20730000, project_volume = 5,
  maxnrec = 50000, binwidth = 5000, xlab.in = " ")
```

**Arguments**

bq	bigquery BigQueryConnection instance
basicfilt	a dplyr::filter operation, defaulting to select non-coding variants in mc3 MAF
chrname	character(1) chromosome token in NCBI seqlevels style
start	numeric(1) base coordinate to start
end	numeric(1) base coordinate to end
project_volume	numeric(1) tumor types will have different numbers of contributions; this parameter tells how many tumor types to represent, counting down from the most frequently represented
maxnrec	numeric(1) for as.data.frame
binwidth	numeric(1) passed to geom_freqpoly
xlab.in	character(1) passed to ggplot2::xlab

**Value**

instance of ggplot

**Examples**

```
if (interactive()) {
  if (!requireNamespace("ggplot2")) stop("install ggplot2 to run this function")
  bq = try(pancan_BQ())
  if (!inherits(bq, "try-error")) {
    ggMutDens(bq)
  }
}
```

---

icd10\_c

*helper for interpreting ICD-10 codes*

---

**Description**

helper for interpreting ICD-10 codes

**Usage**

```
icd10_c
```

**Format**

```
data.frame
```

**Source**

```
ICD-10
```

**Examples**

```
BiocOncoTK::icd10_c
```

---

loadPatel

*use BiocFileCache discipline to acquire patelGBMSC SummarizedExperiment*

---

**Description**

use BiocFileCache discipline to acquire patelGBMSC SummarizedExperiment

**Usage**

```
loadPatel(remotePath = "https://s3.us-east-2.amazonaws.com/biocfound-scrna/patelGBMSC.rds",
  cache = BiocFileCache::BiocFileCache())
```

**Arguments**

remotePath      character(1) identifying remote RDS  
 cache            instance of BiocFileCache, defaults to BiocFileCache::BiocFileCache()

**Value**

a SummarizedExperiment instance

**Note**

The RDS for the SummarizedExperiment is in an AWS S3 bucket. This function will check local cache for the data and will download to cache if not found. That download is a one-time operation for any given value of cache.

**Examples**

```
loadPatel
```

---

```
load_ccleNRAS            utilities for mock data (not involving internet access for vignette)
```

---

**Description**

utilities for mock data (not involving internet access for vignette)

**Usage**

```
load_ccleNRAS()  

load_NRAS_AHR()  

load_nrasdf()
```

**Value**

a list of DRProfSet instances  
 a data.frame with fields 'Cell\_line\_primary\_name', 'RMA\_normalized\_expression', 'HGNC\_gene\_symbol'  
 a data.frame

**Note**

These functions are provided only for avoiding reliance on internet connectivity for document production.

**Examples**

```
load_ccleNRAS()  

dim(load_nrasdf())
```

---

mc3toGR	<i>create a GRanges from the MC3 mutation data</i>
---------	--

---

**Description**

create a GRanges from the MC3 mutation data

**Usage**

```
mc3toGR(bq, basicfilt = function(data) dplyr::filter(data, Consequence ==
  "non_coding_transcript_exon_variant"), maxnrec = 1e+05)
```

**Arguments**

bq	bigquery BigQueryConnection instance
basicfilt	a dplyr::filter instance or NULL to convert entire MAF
maxnrec	numeric(1) used with dplyr::as.data.frame en route to GRanges

**Value**

a GRanges instance

**Examples**

```
mc3toGR
```

---

oncoPrintISB	<i>interactive interface to ComplexHeatmap oncoPrint with inputs from ISB Cancer Genomics Cloud BigQuery back end</i>
--------------	---

---

**Description**

interactive interface to ComplexHeatmap oncoPrint with inputs from ISB Cancer Genomics Cloud BigQuery back end

**Usage**

```
oncoPrintISB(bq)
```

**Arguments**

bq	an instance of <a href="#">BigQueryConnection-class</a> authenticated for ISB Cancer Genomics Cloud access
----	--

**Value**

only used for side effect of running shiny app

**Note**

This function will start a shiny app and will generate queries to Google BigQuery tables representing TCGA.

**Examples**

```
if (interactive()) {  
  bcode = Sys.getenv("CGC_BILLING")  
  if (nchar(bcode)>0) {  
    con <- DBI::dbConnect(bigrquery::bigquery(), project = "isb-cgc",  
      dataset = "tcga_201607_beta", billing = bcode)  
    oncoPrintISB(con)  
  }  
}
```

---

pancan.clin.varnames     *pancan.clin.varnames: a data.frame with a list of variable names for clinical patient data*

---

**Description**

pancan.clin.varnames: a data.frame with a list of variable names for clinical patient data

**Usage**

```
pancan.clin.varnames
```

**Format**

data.frame

**Source**

pancancer-atlas in BigQuery

**Examples**

```
BiocOncoTK::pancan.clin.varnames[1:5,]
```

---

pancan\_app     *provide a shiny app to 'glimpse' structure and content of pancan atlas*

---

**Description**

provide a shiny app to 'glimpse' structure and content of pancan atlas

**Usage**

```
pancan_app(dataset = "Annotated", nrecs = 5)
```

**Arguments**

dataset	character(1) name of a BigQuery dataset in the pancan-atlas project
nrecs	numeric(1) number of records to request (limited through the n= parameter to as.data.table)

**Value**

currently only as a side effect of starting app

**Examples**

```
if (interactive()) pancan_app()
```

---

pancan\_BQ

*provide bigquery connection to pancancer Annotated datasets*

---

**Description**

provide bigquery connection to pancancer Annotated datasets

**Usage**

```
pancan_BQ(dataset = "Annotated", billing = Sys.getenv("CGC_BILLING"),
  ...)
```

**Arguments**

dataset	character(1) dataset name
billing	character(1) Google cloud platform billing code; authentication will be attempted when using the resulting connection
...	passed to <a href="#">dbConnect</a> , for example, quiet=TRUE

**Value**

BigQueryConnection instance

**Examples**

```
pancan_BQ
```



---

pancan\_clinicalTabVarnames  
*give an interface to tablenames*

---

**Description**

give an interface to tablenames

**Usage**

```
pancan_clinicalTabVarnames()
```

**Value**

interactive datatable from DT

**Examples**

```
if (interactive()) pancan_clinicalTabVarnames()
```

---

pancan\_longname      *utility to help find long table names*

---

**Description**

utility to help find long table names

**Usage**

```
pancan_longname(guess, ...)
```

**Arguments**

guess	a regexp to match the table of interest
...	passed to <a href="#">agrep</a>

**Value**

character vector of matches

**Note**

Note that ignore.case=TRUE is set in the function.

**Examples**

```
pancan_longname("rnaseq")
```

---

pancan\_sampTypeMap      *helper for interpreting pancan-atlas sample type codes*

---

**Description**

helper for interpreting pancan-atlas sample type codes

**Usage**

```
pancan_sampTypeMap
```

**Format**

```
data.frame
```

**Note**

The sample type codes are not straightforward to interpret. Primary solid tumor is denoted "TP", and metastatic samples are denoted "TM". This data frame pairs code and natural language terms.

**Source**

ISB BigQuery pancan-atlas project

**Examples**

```
BiocOncoTK::pancan_sampTypeMap
```

---

pancan\_tabulate      *tabulate a variable in a table*

---

**Description**

tabulate a variable in a table

**Usage**

```
pancan_tabulate(dataset = "Annotated", tblname, vblname)
```

**Arguments**

dataset	character(1) dataset name
tblname	character(1) table name in dataset
vblname	character(1) field name in table

**Value**

instance of tbl\_dbi, constituting summarise result

**Examples**

```
if (interactive()) pancan_tabulate(tblname=
  "clinical_PANCAN_patient_with_followup", vblname="icd_10")
```

---

pertClasses	<i>enumerate perturbagen classes</i>
-------------	--------------------------------------

---

**Description**

enumerate perturbagen classes

**Usage**

```
pertClasses(key = Sys.getenv("CLUE_KEY"))
```

**Arguments**

key                    character(1) API key provided by clue.io

**Value**

a character vector

**Examples**

```
if (nchar(Sys.getenv("CLUE_KEY"))>0) {  
  pc = pertClasses()  
  head(vapply(pc, "[", character(1), 1))  
}
```

---

pert_70138	<i>pert_70138: a table with perturbagen information from LINCS</i>
------------	--

---

**Description**

pert\_70138: a table with perturbagen information from LINCS

**Usage**

```
pert_70138
```

**Format**

data.frame

**Source**

GEO GSE70138 GSE70138\_Broad\_LINCS\_pert\_info.txt.gz

**Examples**

```
data(pert_70138)
```

---

query_clue	<i>run the api.clue.io API to acquire information on LINCS experiments</i>
------------	--

---

**Description**

run the api.clue.io API to acquire information on LINCS experiments

**Usage**

```
query_clue(service = "profiles", filter = list(where = (list(pert_iname
  = "sirolimus", cell_id = "MCF7", assay = "L1000"))),
  key = Sys.getenv("CLUE_KEY"))
```

**Arguments**

service	a character(1) service name
filter	a list to be converted to JSON for submission as a GET request
key	character(1) API key provided by clue.io

**Value**

API return value processed by fromJSON

**Examples**

```
if (nchar(Sys.getenv("CLUE_KEY"))>0) {
  demos = clueDemos()
  nd = length(demos)
  chk = lapply(seq_len(nd), function(x) query_clue( service=names(demos)[x],
    filter=demos[[x]]) )
  names(chk) = names(demos)
  sapply(chk,length)
}
```

---

replaceRownames	<i>map rownames of an SE to another vocabulary</i>
-----------------	--

---

**Description**

map rownames of an SE to another vocabulary

**Usage**

```
replaceRownames(se, sourceVocab = "ENTREZID", targetVocab = "SYMBOL")
```

**Arguments**

se	SummarizedExperiment instance
sourceVocab	character(1) must be a keytype of org.Hs.eg.db, defaults to 'ENTREZID'
targetVocab	character(1) must be a column of org.Hs.eg.db

---

TcgaMutCounts	<i>obtain data frame with counts of mutation per gene symbol for selected tumor type</i>
---------------	--

---

**Description**

obtain data frame with counts of mutation per gene symbol for selected tumor type

**Usage**

```
TcgaMutCounts(tumor, limit = NULL, db = "isb-cgc:tcga_201607_beta",
  project)
```

**Arguments**

tumor	character(1) defaults to 'BRCA'
limit	numeric(1) defaults to NULL, appended as limit to number of records returned if non-null
db	character(1) BigQuery database name
project	character(1) project code

**Value**

table as returned by `bigquery::query_exec`

**Note**

This function returns overall mutation count, and many individuals have multiple mutations recorded per gene.

**Examples**

```
if (interactive()) {
  requireNamespace("bigquery")
  tt = TcgaMutCounts("BRCA", project="cgc-05-0009") # substitute your project name
  head(tt)
} # need authentication
```

---

TcgaIndWithAnyMut	<i>Give count of individuals with a mutation recorded for selected tumor</i>
-------------------	--

---

**Description**

Give count of individuals with a mutation recorded for selected tumor

**Usage**

```
TcgaIndWithAnyMut(tumor = "BRCA", limit = NULL,
  db = "isb-cgc:tcga_201607_beta", project)
```

**Arguments**

tumor	character(1) defaults to 'BRCA'
limit	numeric(1) defaults to NULL, appended as limit to number of records returned if non-null
db	character(1) BigQuery database name
project	character(1) project code

**Value**

numeric(1)

**Examples**

```
if (interactive()) TcgaNIndWithAnyMut(project="cgc-05-0009")
```

---

tumNorSet	<i>create list with SEs for tumor and normal for a tumor/assay pairing</i>
-----------	--

---

**Description**

create list with SEs for tumor and normal for a tumor/assay pairing

**Usage**

```
tumNorSet(bq, code = "PRAD",
  assayDataTableName = pancan_longname("rnaseq"),
  assayValueFieldName = "normalized_count",
  assayFeatureName = "Entrez")
```

**Arguments**

bq	a BigQuery connection
code	character(1) a TCGA tumor code, defaults to "PRAD" for prostate tumor
assayDataTableName	character(1) name of table in BigQuery
assayValueFieldName	character(1) field from which assay quantifications are retrieved
assayFeatureName	character(1) field from which assay feature names are retrieved

**Examples**

```
if (interactive()) {
  bqcon = try(pancan_BQ())
  if (!inherits(bqcon, "try-error")) {
    tn = tumNorSet(bqcon)
    tn
  }
}
```

# Index

## \*Topic **datasets**

- annotTabs, [2](#)
- brcaMAE, [4](#)
- CCLE\_DRUG\_BROAD, [6](#)
- cell\_70138, [6](#)
- darmGBMcls, [8](#)
- dingMSI, [8](#)
- fireMSI, [9](#)
- icd10\_c, [12](#)
- pancan.clin.varnames, [15](#)
- pancan\_sampTypeMap, [18](#)
- pert\_70138, [19](#)

[agrep](#), [17](#)

[annotTabs](#), [2](#)

[bindMSI](#), [3](#)

[brcaMAE](#), [4](#)

[buildPancanSE](#), [5, 9](#)

[CCLE\\_DRUG\\_BROAD](#), [6](#)

[cell\\_70138](#), [6](#)

[clueDemos](#), [7](#)

[clueServiceNames](#), [7](#)

[darmGBMcls](#), [8](#)

[dbConnect](#), [16](#)

[dingMSI](#), [8](#)

[featIDMapper](#), [9](#)

[fireMSI](#), [3, 9](#)

[ggFeatDens](#), [10](#)

[ggFeatureSegs](#), [10](#)

[ggMutDens](#), [11](#)

[icd10\\_c](#), [12](#)

[load\\_ccleNRAS](#), [13](#)

[load\\_NRAS\\_AHR \(load\\_ccleNRAS\)](#), [13](#)

[load\\_nrasdf \(load\\_ccleNRAS\)](#), [13](#)

[loadPatel](#), [12](#)

[mc3toGR](#), [14](#)

[oncoPrintISB](#), [14](#)

[pancan.clin.varnames](#), [15](#)

[pancan\\_app](#), [15](#)

[pancan\\_BQ](#), [16](#)

[pancan\\_clinicalTabVarnames](#), [17](#)

[pancan\\_longname](#), [17](#)

[pancan\\_sampTypeMap](#), [18](#)

[pancan\\_tabulate](#), [18](#)

[pert\\_70138](#), [19](#)

[pertClasses](#), [19](#)

[query\\_clue](#), [20](#)

[replaceRownames](#), [20](#)

[TcgaMutCounts](#), [21](#)

[TcgaNIndWithAnyMut](#), [21](#)

[tumNorSet](#), [22](#)