

# Package ‘BiocOncoTK’

February 22, 2021

**Title** Bioconductor components for general cancer genomics

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

**Version** 1.10.0

**Author** Vince Carey

**Suggests** knitr, dbplyr, 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, DT, GenomicRanges, IRanges, ggplot2, SummarizedExperiment, DBI, GenomicFeatures, curatedTCGAData, scales, ggpubr, plyr, car, graph, Rgraphviz

**Depends** R (>= 3.6.0), methods, utils

**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** 7.0.2

**Encoding** UTF-8

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

**git\_branch** RELEASE\_3\_12

**git\_last\_commit** 4267875

**git\_last\_commit\_date** 2020-10-27

**Date/Publication** 2021-02-21

**R topics documented:**

add_sym . . . . .	3
annotTabs . . . . .	3
bindMSI . . . . .	4
bipg_tests . . . . .	5
brcaMAE . . . . .	5
buildPancanSE . . . . .	6
CCLC_DRUG_BROAD . . . . .	8
cell_70138 . . . . .	8
clueDemos . . . . .	9
clueServiceNames . . . . .	9
darmGBMcls . . . . .	10
dingMSI . . . . .	10
featIDMapper . . . . .	11
fireMSI . . . . .	11
get_plates . . . . .	12
ggFeatDens . . . . .	12
ggFeatureSegs . . . . .	13
ggMutDens . . . . .	14
icd10_c . . . . .	15
k23sig . . . . .	16
kang_DNArepair . . . . .	16
loadPatel . . . . .	17
load_cclcNRAS . . . . .	17
log10p11 . . . . .	18
map_tcga_ncit . . . . .	18
mc3toGR . . . . .	19
molpo_3utr . . . . .	20
molpo_5utr . . . . .	20
molpo_CDS . . . . .	21
molpo_WGS . . . . .	21
MSIsensor.10k . . . . .	22
multviz . . . . .	22
oncoPrintISB . . . . .	23
pancan.clin.varnames . . . . .	24
pancan_app . . . . .	24
pancan_BQ . . . . .	25
pancan_clinicalTabVarnames . . . . .	25
pancan_longname . . . . .	26
pancan_sampTypeMap . . . . .	26
pancan_tabulate . . . . .	27
patient_to_tumor_code . . . . .	27
pertClasses . . . . .	28
pert_70138 . . . . .	28
query_clue . . . . .	29
replaceRownames . . . . .	30
small_msi . . . . .	30
TcgaMutCounts . . . . .	31
TcgaNIndWithAnyMut . . . . .	31
tumNorSet . . . . .	32
viz_msi_raw . . . . .	33

---

add_sym	<i>add symbols in rowData to a SummarizedExperiment that has Entrez IDs for rownames</i>
---------	--

---

**Description**

add symbols in rowData to a SummarizedExperiment that has Entrez IDs for rownames

**Usage**

```
add_sym(x)
```

**Arguments**

x                      SummarizedExperiment instance

**Note**

Will fail if 'symbol' is a column of rowData(x)

**Examples**

```
if (interactive()) {
  bq = pancan_BQ()
  rnse = try(buildPancanSE(bq, assay="RNASeqv2"))
  if (inherits(rnse, "try-error")) stop("probably need CGC_BILLING set in environment or with pancan_BQ")
  add_sym(rnse)
}
```

---

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 MSIsensor 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. \*OF NOTE:\* The MSIsensor data from Ding's cell paper (see `help(dingMSI)` for URL) provides the participant barcode. The participant barcode is a substring of the sample barcode. Be sure to filter the input SummarizedExperiment to include only tumor samples, using the `substr(colnames(se),14,15)` (values "10"..."14" correspond to normal, non-tumor samples.) Additionally, `bindMSI` will only work if the `colnames` of the (filtered) SummarizedExperiment have been truncated to the participant barcode, that is, the first 12 characters of the sample barcode.

### Examples

```
bindMSI
```

---

bigg_tests	<i>configure a bipartite graph relating tumor type to gene, using graphNEL</i>
------------	--

---

**Description**

configure a bipartite graph relating tumor type to gene, using graphNEL

**Usage**

```
bigg_tests(
  stattab,
  genes_adverse = NA,
  genes_favorable = NA,
  gpar_cex = 0.65,
  gpar_lwd = 0
)
```

**Arguments**

stattab	a data.frame with columns 'tumor', 'gene', and 'tstat'
genes_adverse	a vector of genes whose increased expression is regarded as adverse
genes_favorable	a vector of genes whose increased expression is regarded as favorable
gpar_cex	tune size of graph labels
gpar_lwd	tune appearance of node boundaries

**Value**

a graphNEL instance (graph package)

**Examples**

```
bigg_tests(k23sig)
```

---

brcaMAE	<i>a virtual MultiAssayExperiment for pancancer-atlas BRCA data</i>
---------	---

---

**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

*helper for SummarizedExperiment construction from pancan*

---

**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 <code>BigQueryConnection</code> for <code>pancancer-atlas.Annotated Dataset</code>
acronym	character(1) 'cohort' label, e.g., 'BLCA'
assay	character(1) element from <code>names(BiocOncoTK::annotTabs)</code> , e.g., 'meth450k'. If 'assay == "mc3_MAF"' an error is thrown as the mutation data are inconsistently annotated; the message produced directs the user to 'mc3toGR'.
sampType	character(1) element from <code>BiocOncoTK::pancan_sampTypeMap\$"SampleTypeLetterCode"</code> , 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 <code>SummarizedExperiment</code> and returns a <code>SummarizedExperiment</code> ; useful for feature name remapping, defaults to <code>force</code> (does nothing)
bindMethRowranges	logical(1) if true and assay is <code>meth27k</code>
featIDMap	a named character() vector defining, for each assay type, what field should be used to label features in rownames. or <code>meth450k</code> , annotation from <code>FDb.InfiniumMethylation.hg19</code> and <code>EnsDb.Hsapiens.v75</code> is obtained for available features and bound into the <code>rowRanges</code> 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	<i>generate lists to generate clue API queries</i>
-----------	--

---

**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	<i>Provide names of some clue.io services for which examples are available in this package.</i>
------------------	---

---

**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_lstpm format from Darmanis 2017 (PMC 5810554) single cell RNA-seq in GBM</i>
------------	---

---

**Description**

Data in count\_lstpm 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 curatedTCGA-Data</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))
```

---

get_plates	<i>use curatedTCGAData request to acquire plate codes for samples</i>
------------	---

---

### Description

use curatedTCGAData request to acquire plate codes for samples

### Usage

```
get_plates(
  tumcode = "BLCA",
  assay = "RNASeq2GeneNorm",
  samptypes = c("01", "02", "03", "04", "06", "09", "40")
)
```

### Arguments

tumcode	a TCGA tumor code, usually 3 or for characters
assay	a curatedTCGAData assay code, run curatedTCGAData() to see a message with available options
samptypes	a character vector with codes as defined at <a href="https://gdc.cancer.gov/resources-tcga-users/tcga-code-tables/sample-type-codes">https://gdc.cancer.gov/resources-tcga-users/tcga-code-tables/sample-type-codes</a>

### Value

a data.frame with a row for each TCGA contribution for the selected tumor type and assay type

### Examples

```
if (interactive()) {
  plts_blca_rnaseq = get_plates()
  dim(plts_blca_rnaseq)
  head(plts_blca_rnaseq)
}
```

---

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**

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

**Value**

ggplot instance

**Examples**

```
ggFeatDens
```

---

ggFeatureSegs

*generate a ggplot of segments of gene-like regions*

---

**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
```

---

k23sig	<i>a table of 'significant' MSIsensor-score/expression relationships in TCGA</i>
--------	--

---

**Description**

a table of 'significant' MSIsensor-score/expression relationships in TCGA

**Usage**

```
k23sig
```

**Format**

```
data.frame
```

**Note**

provided to demonstrate bipartite graph construction

**Examples**

```
head(k23sig)
```

---

kang_DNArepair	<i>list of 151 genes annotated as DNA repair pathway members</i>
----------------	--

---

**Description**

list of 151 genes annotated as DNA repair pathway members

**Usage**

```
kang_DNArepair
```

**Format**

```
named list
```

**Note**

The zipped PDF was read using `pdftools::pdf_text` and then manually organized. All gene symbols present in `curatedTCGAData.RNASEq2GeneNorm` rownames. The list elements are ATM, BER, FA.HR, MMR, NER, NHEJ, OTHER, TLS, RECQ, and XLR. These denote, respectively, ataxia-telangiectasia-mutated, base excision repair, Fanconi anemia/homologous recombination, mismatch repair, nucleotide excision repair, non-homologous end joining, other, translesion synthesis, recQ helicase pathway, and cross-link repair.



**Source**

<https://academic.oup.com/jnci/article/104/9/670/872781#supplementary-data>

---

loadPatel	<i>use BiocFileCache discipline to acquire patelGBMSC SummarizedExperiment</i>
-----------	--

---

**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	<i>utilities for mock data (not involving internet access for vignette)</i>
---------------	---

---

**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())
```

---

log10p11

*log10(x+p) transformation for use with scales/ggplot2*

---

**Description**

log10(x+p) transformation for use with scales/ggplot2

**Usage**

```
log10p11(p = 1)
```

**Arguments**

p value of shift before taking log10

**Value**

an instance of custom trans() for scales package

---

map\_tcga\_ncit

*a manually constructed table mapping TCGA acronyms to NCIT thesaurus tags*

---

**Description**

a manually constructed table mapping TCGA acronyms to NCIT thesaurus tags

**Usage**

```
map_tcga_ncit
```

**Format**

data.frame

**Note**

Constructed using `ontoProc::getOncotreeOnto()` result. See the vignette on Mapping TCGA tumor codes to NCIT for elaborating the mapping to aggregate tumors into NCIT organ systems.

---

mc3toGR

*create a GRanges from the MC3 mutation data*


---

**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**

```
if (interactive()) {
  con = try(pancan_BQ()) # need CGC_BILLING set
  if (!inherits(con, "try-error")) {
    aut = as.character(1:22) # some records in BQ have missing Chromosome
    chk = mc3toGR(con, basicfilt=function(data) dplyr::filter(data,
      project_short_name=="TCGA-BRCA",
      SYMBOL=="TP53", Chromosome %in% aut))
    print(chk[,1:5]) # lots of mcol fields
    table(chk$Variant_Classification)
  }
}
```

---

molpo_3utr	<i>representation of 3'UTR MSI events in TCGA from Cortes-Ciriano et al. 2017</i>
------------	---

---

**Description**

representation of 3'UTR MSI events in TCGA from Cortes-Ciriano et al. 2017

**Usage**

molpo\_3utr

**Format**

SummarizedExperiment

**Note**

Supplementary data 6 from publication noted in Source. See metadata() component of this SummarizedExperiment for more details.

**Source**

<https://www.nature.com/articles/ncomms15180#Sec22>

**Examples**

molpo\_3utr

---

molpo_5utr	<i>representation of 5'UTR MSI events in TCGA from Cortes-Ciriano et al. 2017</i>
------------	---

---

**Description**

representation of 5'UTR MSI events in TCGA from Cortes-Ciriano et al. 2017

**Usage**

molpo\_5utr

**Format**

SummarizedExperiment

**Note**

Supplementary data 7 from publication noted in Source. See metadata() component of this SummarizedExperiment for more details.

**Source**

<https://www.nature.com/articles/ncomms15180#Sec22>

**Examples**

molpo\_5utr

---

molpo_CDS	<i>representation of MSI events in coding regions TCGA from Cortes-Ciriano et al. 2017</i>
-----------	--

---

**Description**

representation of MSI events in coding regions TCGA from Cortes-Ciriano et al. 2017

**Usage**

molpo\_CDS

**Format**

SummarizedExperiment

**Note**

Supplementary data 5 from publication noted in Source. See metadata() component of this SummarizedExperiment for more details.

**Source**

<https://www.nature.com/articles/ncomms15180#Sec22>

**Examples**

molpo\_CDS

---

molpo_WGS	<i>representation of events detected in 708 WGS experiments TCGA from Cortes-Ciriano et al. 2017</i>
-----------	--

---

**Description**

representation of events detected in 708 WGS experiments TCGA from Cortes-Ciriano et al. 2017

**Usage**

molpo\_WGS

**Format**

SummarizedExperiment

**Note**

Supplementary data 10 from publication noted in Source. See metadata() component of this SummarizedExperiment for more details.

**Source**

<https://www.nature.com/articles/ncomms15180#Sec22>

**Examples**

```
molpo_WGS
```

---

MSIsensor.10k	<i>MSIsensor microsatellite instability scores for TCGA, collected from Ding et al. Cell 173(2) 2018.</i>
---------------	---

---

**Description**

MSIsensor microsatellite instability scores for TCGA, collected from Ding et al. Cell 173(2) 2018.

**Usage**

```
MSIsensor.10k
```

**Format**

```
DataFrame
```

**Source**

<https://www.cell.com/cms/10.1016/j.cell.2018.03.033/attachment/0ac495ba-3578-41cf-8fb1-94487f554444/mmc5.xlsx> retrieved 9/17/2018.

**Examples**

```
str(BiocOncTK::dingMSI)
```

---

multiviz	<i>visualize aspects of MSIsensor/expression relationships</i>
----------	--

---

**Description**

visualize aspects of MSIsensor/expression relationships

**Usage**

```

multiviz(
  tum = "MESO",
  gene = "TYMS",
  intrans = log10p11(p = 1),
  inmeth = "auto",
  topmsi = Inf,
  indata,
  nvar = 6
)

```

**Arguments**

tum	a TCGA tumor code
gene	a gene symbol used in the indata data.frame
intrans	an instance of the trans() transformation method of scales package
inmeth	a valid setting for method parameter for geom_smooth
topmsi	maximum numeric value for x-axis when plotting against MSI value
indata	a data.frame instance with values for acronym, gene, msival
nvar	numeric() number of variables to show in biplot

---

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")
```

---

```
patient_to_tumor_code data.frame mapping from TCGA patient_barcode to TCGA tumor code
```

---

**Description**

data.frame mapping from TCGA patient\_barcode to TCGA tumor code

**Usage**

```
patient_to_tumor_code
```

**Format**

data.frame

**Note**

Used IDs recorded in MSISensor.10k; one is unmatched at TCGA portal metadata() component of this SummarizedExperiment for more details.

**Source**

<https://portal.gdc.cancer.gov/exploration?uploadCaseTab=matched>

**Examples**

```
head(patient_to_tumor_code)
```

---

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

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

---

## 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

---

small_msi	<i>filtered MSI data for demonstrating exploratory app</i>
-----------	--

---

**Description**

filtered MSI data for demonstrating exploratory app

**Usage**

```
small_msi
```

**Format**

DataFrame

**Source**

MSI values from dingMSI, expression from curatedTCGADData for three genes, two tumors

**Examples**

```
head(BiocOncoTK::small_msi)
```

---

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 bigrquery::query\_exec

**Note**

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

**Examples**

```
if (interactive()) {
  requireNamespace("bigrquery")
  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
  }
}
```



---

viz_msi_raw	<i>small app to survey MSI sensor against expression</i>
-------------	--

---

**Description**

small app to survey MSI sensor against expression

**Usage**

```
viz_msi_raw(df, inmeth = MASS::rlm, nvar = 3)
```

**Arguments**

df	a data.frame instance
inmeth	a method for geom_smooth
nvar	number of variables to show in biplot

**Note**

Use ask=FALSE if running example.

**Examples**

```
if (interactive()) viz_msi_raw(BiocOncoTK::small_msi, nvar=3)
```

# Index

## \* datasets

- annotTabs, 3
  - brcaMAE, 5
  - CCLC\_DRUG\_BROAD, 8
  - cell\_70138, 8
  - darmGBMcls, 10
  - dingMSI, 10
  - fireMSI, 11
  - icd10\_c, 15
  - k23sig, 16
  - kang\_DNArepair, 16
  - map\_tcga\_ncit, 18
  - molpo\_3utr, 20
  - molpo\_5utr, 20
  - molpo\_CDS, 21
  - molpo\_WGS, 21
  - MSIsensor.10k, 22
  - pancan.clin.varnames, 24
  - pancan\_sampTypeMap, 26
  - patient\_to\_tumor\_code, 27
  - pert\_70138, 28
  - small\_msi, 30
- add\_sym, 3
- agrep, 26
- annotTabs, 3
- bindMSI, 4
- big\_tests, 5
- brcaMAE, 5
- buildPancanSE, 6, 11
- CCLC\_DRUG\_BROAD, 8
- cell\_70138, 8
- clueDemos, 9
- clueServiceNames, 9
- darmGBMcls, 10
- dbConnect, 25
- dingMSI, 10
- featIDMapper, 11
- fireMSI, 4, 11
- get\_plates, 12
- ggFeatDens, 12
- ggFeatureSegs, 13
- ggMutDens, 14
- icd10\_c, 15
- k23sig, 16
- kang\_DNArepair, 16
- load\_ccleNRAS, 17
- load\_NRAS\_AHR (load\_ccleNRAS), 17
- load\_nrasdf (load\_ccleNRAS), 17
- loadPatel, 17
- log10p11, 18
- map\_tcga\_ncit, 18
- mc3toGR, 19
- molpo\_3utr, 20
- molpo\_5utr, 20
- molpo\_CDS, 21
- molpo\_WGS, 21
- MSIsensor.10k, 22
- multiviz, 22
- oncoPrintISB, 23
- pancan.clin.varnames, 24
- pancan\_app, 24
- pancan\_BQ, 25
- pancan\_clinicalTabVarnames, 25
- pancan\_longname, 26
- pancan\_sampTypeMap, 26
- pancan\_tabulate, 27
- patient\_to\_tumor\_code, 27
- pert\_70138, 28
- pertClasses, 28
- query\_clue, 29
- replaceRownames, 30
- small\_msi, 30
- TcgaMutCounts, 31
- TcgaNIndWithAnyMut, 31
- tumNorSet, 32
- viz\_msi\_raw, 33