Package: gcap (via r-universe)

July 19, 2024

Type Package

Title Gene-level Circular Amplicon Prediction

Version 1.2.0

Description Provides data processing pipeline feeding paired bam files (or allele-specific copy number data) and XGBOOST model for predicting tumor circular amplicons (also known as ecDNA) in gene level.

License Non-Commercial Academic License + file LICENSE

URL https://github.com/ShixiangWang/gcap, https://shixiangwang.github.io/gcap/

BugReports https://github.com/ShixiangWang/gcap/issues

Depends ASCAT (>= 3.0.0), R (>= 3.5), sigminer (>= 2.1.1)

Imports cli (>= 3.1.0), data.table, GetoptLong, glue, lgr, magrittr, mltools, purrr, quadprog, R6, rappdirs, Rcpp, stats, uuid, xgboost

Suggests BiocManager, copynumber, facets, IDConverter (>= 0.3.0), PRROC, sequenza, testthat (>= 3.0.0), utils

LinkingTo Rcpp

Remotes git::https://bitbucket.org/sequenzatools/sequenza.git@master, github::VanLoo-lab/ascat/ASCAT, github::ShixiangWang/copynumber, github::ShixiangWang/facets, github::ShixiangWang/IDConverter

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE, roclets = c(``collate", ``namespace", ``rd", ``roxytest::testthat_roclet"))

RoxygenNote 7.3.1

Repository https://shixiangwang.r-universe.dev

RemoteUrl https://github.com/ShixiangWang/gcap

RemoteRef HEAD

RemoteSha 895bfd656f5d56c3a615e9ad04fb1df63cf37d94

Contents

`ascn` *Example allele specific copy number (ASCN) data*

Description

Example allele specific copy number (ASCN) data

Format

A `data.frame`

Source

Generate from `data-raw/`, raw source from our study by calling ASCAT v3.0 alpha on corresponding WES sequencing data.

Examples

```
data("ascn")
```

deploy — *Deploy Command Line Interface to System Local Path*

Description

Only should be used in Unix-like system. For details of the arguments passing to CLI, please check `gcap.workflow()` and `gcap.ASCNworkflow()`.

Usage

```
deploy(target = "/usr/local/bin")
```

Arguments

`target`	the target path to deploy the CLI.

Value

Nothing.

ec — *Example ecDNA training data*

Description

Example ecDNA training data

Format

A `data.table`

Source

Generate from `data-raw/`

Examples

```
data("ec")
```

fCNA — *R6 class representing focal copy number amplification list predicted from a cohort*

Description

Contains fields storing data and methods to get, process and visualize fCNA information. Examples please see `gcap.ASCNworkflow()`.

Public fields

`data` a `data.table` storing fCNA list, which typically contains following columns:

- `sample` sample or case ID.
- `band` chromosome cytoband.
- `gene_id` gene ID, typically Ensembl ID. You can convert the ID with R package `IDConverter`.
- `total_cn` total copy number value.
- `minor_cn` copy number value for minor allele.
- `prob` the probability the gene located in circular DNA.
- `gene_class` gene level amplicon classification.

`sample_summary` a `data.table` storing sample summary data, which typically contains at least the following columns:

- `sample` sample or case ID. **Should only include cases have been called with GCAP workflow, otherwise the extra cases would be automatically classified as 'nofocal' (i.e. `NA` in `sample_summary` field) class**.
- `purity`, `ploidy` for tumor purity or ploidy.
- `AScore` aneuploidy score.
- `pLOH` genome percentage harboring LOH events.
- `CN1 ... CN19` activity of copy number signatures.
- `class` the sample class based on amplicon type.
- `ec_genes` number of genes predicted as located on circular DNA.
- `ec_possibly_genes` same with `ec_genes` but with less confidence.
- `ec_cytobands` number of cytobands predicted as located on circular DNA. (the regions of `ec_possibly_genes` are not included in computation)

Active bindings

`min_prob` check `$new()` method for details. If you updated this value, a function will be called to update the sample summary.

Methods

Public methods:

- `fCNA$new()`
- `fCNA$subset()`

- fCNA$getSampleSummary()
- fCNA$getGeneSummary()
- fCNA$getCytobandSummary()
- fCNA$saveToFiles()
- fCNA$convertGeneID()
- fCNA$print()

Method `new()`**:** Create a `fCNA` object. Typically, you can obtain this object from `gcap.workflow()` or `gcap.ASCNworkflow()`.

Usage:

```
fCNA$new(
  fcna,
  pdata = fcna[, "sample", drop = FALSE],
  min_prob = 0.6,
  only_oncogenes = FALSE,
  genome_build = c("hg38", "hg19", "mm10")
)
```

Arguments:

`fcna` a `data.frame` storing focal copy number amplicon list.

`pdata` a `data.frame` storing phenotype or sample-level related data. (Optional)

`min_prob` the minimal aggregated (in cytoband level) probability to determine a circular amplicon.

`only_oncogenes` only_oncogenes if `TRUE`, only known oncogenes are kept for circular prediction.

`genome_build` genome version

Method `subset()`**:** Return a subset `fCNA` object

Usage:

```
fCNA$subset(..., on = c("data", "sample_summary"))
```

Arguments:

`...` subset expressions on `fCNA$data` or `fCNA$sample_summary`.

`on` if it is "data", subset operations are on data field of `fCNA` object, same for "sample_summary".

Returns: a `fCNA`

Method `getSampleSummary()`**:** Get sample summary of fCNA

Usage:

```
fCNA$getSampleSummary(
  only_oncogenes = FALSE,
  genome_build = c("hg38", "hg19", "mm10")
)
```

Arguments:

`only_oncogenes` only_oncogenes if `TRUE`, only known oncogenes are kept for circular prediction.

`genome_build` genome version.

Returns: a `data.table`

Method `getGeneSummary()`**:** Get gene level summary of fCNA type

Usage:

```
fCNA$getGeneSummary(return_mat = FALSE)
```

Arguments:

`return_mat` if `TRUE`, return a cytoband by sample matrix instead of a summary.

Returns: a `data.table` or a `matrix`.

Method `getCytobandSummary()`**:** Get cytoband level summary of fCNA type

Usage:

```
fCNA$getCytobandSummary(unique = FALSE, return_mat = FALSE)
```

Arguments:

`unique` if `TRUE`, count sample frequency instead of gene frequency.

`return_mat` if `TRUE`, return a cytoband by sample matrix instead of a summary.

Returns: a `data.table`

Method `saveToFiles()`**:** Save the key data to local files

Usage:

```
fCNA$saveToFiles(dirpath, fileprefix = "fCNA")
```

Arguments:

`dirpath` directory path storing output files.

`fileprefix` file prefix. Two result files shall be generated.

Method `convertGeneID()`**:** Convert Gene IDs between Ensembl and Hugo Symbol System

Usage:

```
fCNA$convertGeneID(
  type = c("ensembl", "symbol"),
  genome_build = c("hg38", "hg19", "mm10")
)
```

Arguments:

`type` type of input IDs, could be 'ensembl' or 'symbol'.

`genome_build` reference genome build.

Method `print()`**:** print the fCNA object

Usage:

```
fCNA$print(...)
```

Arguments:

`...` unused.

gcap.ASCNworkflow *GCAP workflow for gene-level amplicon prediction from ASCN input*

Description

Unlike gcap.workflow, this function directly uses the allele-specific copy number data along with some extra sample information to infer ecDNA genes.

Usage

```
gcap.ASCNworkflow(
  data,
  genome_build = c("hg38", "hg19"),
  model = "XGB11",
  tightness = 1L,
  gap_cn = 3L,
  overlap = 1,
  only_oncogenes = FALSE,
  outdir = getwd(),
  result_file_prefix = paste0("gcap_", uuid::UUIDgenerate(TRUE))
)
```

Arguments

`data` a `data.frame` with following columns. The key columns can be obtained from common allele specific CNV calling software, e.g., ASCAT, Sequenza, FACETS.

- chromosome: chromosome names starts with ’chr’.
- start: start position of the segment.
- end: end position of the segment.
- total_cn: total integer copy number of the segment.
- minor_cn: minor allele integer copy number of the segment. Set it to `NA` if you don’t have this data.
- sample: sample identifier.
- purity: tumor purity of the sample. Set to `1` if you don’t know.
- ploidy (optinal): ploidy value of the sample tumor genome.
- age (optional): age of the case, use along with `gender`.
- gender (optional): gender of the case, use along with `age`.
- type (optional): cancer type of the case, use along with `age` and `gender`. Please refer to gcap.collapse2Genes to see the supported cancer types. This info is only used in ’XGB56’ model. If you don’t use this model, you don’t need to set it.

`genome_build` "hg38" or "hg19".

`model` model name ("XGB11", "XGB32", "XGB56") or a custom model from input. ’toy’ can be used for test.

`tightness` a coefficient to times to TCGA somatic CN to set a more strict threshold as a circular amplicon. If the value is larger, it is more likely a fCNA assigned to `noncircular` instead of `circular`. **When it is `NA`, we don't use TCGA somatic CN data as reference**.

`gap_cn` a gap copy number value. A gene with copy number above background (`ploidy + gap_cn` in general) would be treated as focal amplicon. Smaller, more amplicons.

`overlap` the overlap percentage on gene.

`only_oncogenes` if `TRUE`, only known oncogenes are kept for circular prediction.

`outdir` result output path.

`result_file_prefix` file name prefix (without directory path) for storing final model prediction file in CSV format. Default a unique file name is generated by UUID approach.

Value

a list of invisible `data.table` and corresponding files saved to local machine.

Examples

```
data("ascn")
data <- ascn
rv <- gcap.ASCNworkflow(data, outdir = tempdir(), model = "XGB11")
data$purity <- 1
rv2 <- gcap.ASCNworkflow(data, outdir = tempdir(), model = "XGB11")
data$age <- 60
data$gender <- "XY"
rv3 <- gcap.ASCNworkflow(data, outdir = tempdir(), model = "XGB32")
# If you want to use 'XGB56', you should include 'type' column
data$type <- "LUAD"
rv4 <- gcap.ASCNworkflow(data, outdir = tempdir(), model = "XGB56")
# If you only have total integer copy number
data$minor_cn <- NA
rv5 <- gcap.ASCNworkflow(data, outdir = tempdir(), model = "XGB11")

# R6 class fCNA --------------------------------
print(rv)
print(rv$data)
print(rv$sample_summary)
print(rv$gene_summary)
print(rv$cytoband_summary)

# Create a subset fCNA
rv_subset <- rv$subset(total_cn > 10)
nrow(rv$data)
nrow(rv_subset$data)

rv_subset2 <- rv$subset(sample == "TCGA-02-2485-01")
nrow(rv_subset2$data)
unique(rv_subset2$data$sample)
```

```
sum_gene <- rv$getGeneSummary()
sum_gene
mat_gene <- rv$getGeneSummary(return_mat = TRUE)
mat_gene

sum_cytoband <- rv$getCytobandSummary()
sum_cytoband
mat_cytoband <- rv$getCytobandSummary(return_mat = TRUE)
mat_cytoband
```

gcap.collapse2Genes *Generate unified gene-level feature data*

Description

Generate unified gene-level feature data

Usage

```
gcap.collapse2Genes(
  fts,
  extra_info = NULL,
  include_type = FALSE,
  fix_type = TRUE,
  genome_build = c("hg38", "hg19", "mm10"),
  overlap = 1
)
```

Arguments

fts	(modified) result from gcap.extractFeatures()
extra_info	(optional) a data.frame with 3 columns 'sample', 'age' and 'gender', for including cancer type, check parameter include_type. For gender, should be 'XX' or 'XY', also could be 0 for 'XX' and 1 for 'XY'.
include_type	if TRUE, a fourth column named 'type' should be included in extra_info, the supported cancer type should be described with TCGA cancer type abbr..
fix_type	default is TRUE, only cancer types used in pre-trained models are used, others will be convert to NA. If FALSE, only generating one-hot encoding for cancer types in input data.
genome_build	genome build version, should be one of 'hg38', 'hg19'.
overlap	the overlap percentage on gene.

Value

a data.table.

gcap.extractFeatures *Extract sample and region level features*

Description

Extract sample and region level features

Usage

```
gcap.extractFeatures(
  ascat_files,
  genome_build = c("hg38", "hg19", "mm10"),
  ascn_data = NULL
)
```

Arguments

`ascat_files`	a list of file path. Typically the result of `gcap.runASCAT()`
`genome_build`	genome build version, should be one of 'hg38', 'hg19'.
`ascn_data`	if `ascat_files` is missing, an alternative `data.frame` can be provided for ASCN data along with purity and ploidy (optional).

Value

a `list`.

gcap.runASCAT *Run ASCAT on tumor-normal pair WES data files*

Description

A wrapper calling ASCAT on WES data on one or more tumor(-normal paired) bam data. Note, for multiple tumor-normal pairs, the first 5 arguments should be a vector with same length.

Usage

```
gcap.runASCAT(
  tumourseqfile,
  normalseqfile = NA_character_,
  tumourname,
  normalname = NA_character_,
  jobname = tumourname,
  outdir = getwd(),
  allelecounter_exe = "~/miniconda3/envs/cancerit/bin/alleleCounter",
  g1000allelesprefix = file.path("~/data/snp/1000G_loci_hg38",
    "1kg.phase3.v5a_GRCh38nounref_allele_index_chr"),
  g1000lociprefix = file.path("~/data/snp/1000G_loci_hg38",
    "1kg.phase3.v5a_GRCh38nounref_loci_chrstring_chr"),
  GCcontentfile = "~/data/snp/GC_correction_hg38.txt",
  replictimingfile = "~/data/snp/RT_correction_hg38.txt",
  nthreads = 22,
  minCounts = 10,
  BED_file = NA,
  probloci_file = NA,
  chrom_names = 1:22,
  gender = "XX",
  min_base_qual = 20,
  min_map_qual = 35,
  penalty = 70,
  genome_build = "hg38",
  skip_finished_ASCAT = FALSE
)
```

Arguments

`tumourseqfile` Full path to the tumour BAM file.

`normalseqfile` Full path to the normal BAM file.

`tumourname` Identifier to be used for tumour output files.

`normalname` Identifier to be used for normal output files.

`jobname` job name, typically an unique name for a tumor-normal pair.

`outdir` result output path.

`allelecounter_exe` Path to the allele counter executable.

`g1000allelesprefix` Prefix path to the allele data (e.g. "G1000_alleles_chr").

`g1000lociprefix` Prefix path to the loci data (e.g. "G1000_loci_chr").

`GCcontentfile` File containing the GC content around every SNP for increasing window sizes.

`replictimingfile` File containing replication timing at every SNP for various cell lines.

`nthreads` The number of parallel processes for getting allele counts (optional, default=1).

`minCounts` Minimum depth required in the normal for a SNP to be considered (optional, default=10).

`BED_file` A BED file for only looking at SNPs within specific intervals (optional, default=NA).

`probloci_file` A file (chromosome <tab> position; no header) containing specific loci to ignore (optional, default=NA).

`chrom_names` A vector containing the names of chromosomes to be considered (optional, default=1:22).

`gender` a vector of gender for each cases ("XX" or "XY"). Default = all female ("XX"). Ignore this if you don't include sex chromosomes.

`min_base_qual` Minimum base quality required for a read to be counted (optional, default=20).

`min_map_qual` Minimum mapping quality required for a read to be counted (optional, default=35).

`penalty` penalty of introducing an additional ASPCF breakpoint (expert parameter, don't adapt unless you know what you're doing)

`genome_build` "hg38" or "hg19".

`skip_finished_ASCAT`
if `TRUE`, skipped finished ASCAT calls to save time.

Value

Nothing. Check the `outdir` for results.

`gcap.runASCNBuildflow` *Build data for prediction from absolute copy number data*

Description

This is is a wrapper of `gcap.extractFeatures()` and `gcap.collapse2Genes()` to combine the feature extraction and predict input generate procedure. If you want to modify the result of `gcap.extractFeatures()`, you should always use the two functions instead of this wrapper.

Usage

```
gcap.runASCNBuildflow(data, genome_build = c("hg38", "hg19"), overlap = 1)
```

Arguments

`data` a `data.frame` with following columns. The key columns can be obtained from common allele specific CNV calling software, e.g., ASCAT, Sequenza, FACETS.

- chromosome: chromosome names starts with 'chr'.
- start: start position of the segment.
- end: end position of the segment.
- total_cn: total integer copy number of the segment.
- minor_cn: minor allele integer copy number of the segment. Set it to `NA` if you don't have this data.
- sample: sample identifier.
- purity: tumor purity of the sample. Set to `1` if you don't know.
- ploidy (optinal): ploidy value of the sample tumor genome.
- age (optional): age of the case, use along with `gender`.
- gender (optional): gender of the case, use along with `age`.

- type (optional): cancer type of the case, use along with age and gender. Please refer to gcap.collapse2Genes to see the supported cancer types. This info is only used in ’XGB56’ model. If you don’t use this model, you don’t need to set it.

`genome_build` "hg38" or "hg19".

`overlap` the overlap percentage on gene.

Value

a `data.table`.

See Also

gcap.runBuildflow

`gcap.runBuildflow` *Build data for prediction from ASCAT result files*

Description

This is is a wrapper of `gcap.extractFeatures()` and `gcap.collapse2Genes()` to combine the feature extraction and predict input generate procedure. If you want to modify the result of `gcap.extractFeatures()`, you should always use the two functions instead of this wrapper.

Usage

```
gcap.runBuildflow(
  ascat_files,
  extra_info,
  include_type = FALSE,
  genome_build = c("hg38", "hg19", "mm10"),
  overlap = 1
)
```

Arguments

`ascat_files` a list of file path. Typically the result of `gcap.runASCAT()`

`extra_info` (optional) a `data.frame` with 3 columns ’sample’, ’age’ and ’gender’, for including cancer type, check parameter `include_type`. For gender, should be ’XX’ or ’XY’, also could be `0` for ’XX’ and 1 for ’XY’.

`include_type` if `TRUE`, a fourth column named ’type’ should be included in `extra_info`, the supported cancer type should be described with TCGA cancer type abbr..

`genome_build` genome build version, should be one of ’hg38’, ’hg19’.

`overlap` the overlap percentage on gene.

Value

a `data.table`.

gcap.runPrediction *Run gene-level circular prediction*

Description

Run gene-level circular prediction

Usage

```
gcap.runPrediction(data, model = "XGB11")
```

Arguments

data	data to predict (data.frame/matrix format), from gcap.collapse2Genes() in general.
model	model name ("XGB11", "XGB32", "XGB56") or a custom model from input. 'toy' can be used for test.

Value

a numeric vector representing prob.

Examples

```
data("ec")
# Use toy model for illustration
y_pred <- gcap.runPrediction(ec, "toy")
y_pred
```

gcap.runScoring *Summarize prediction result into gene/sample-level*

Description

Summarize prediction result into gene/sample-level

Usage

```
gcap.runScoring(
  data,
  genome_build = "hg38",
  min_prob = 0.6,
  tightness = 1L,
  gap_cn = 3L,
  only_oncogenes = FALSE
)
```

Arguments

`data` a `data.table` containing result from gcap.runPrediction.

`genome_build` genome build version, should be one of 'hg38', 'hg19'.

`min_prob` the minimal aggregated (in cytoband level) probability to determine a circular amplicon. The default value is for the balance of recall and precision. **We highly recomment set it to 0.95 or larger if you want to detect solid positive cases (for experimental validation etc.) instead of subtyping cases**.

`tightness` a coefficient to times to TCGA somatic CN to set a more strict threshold as a circular amplicon. If the value is larger, it is more likely a fCNA assigned to `noncircular` instead of `circular`. **When it is `NA`, we don't use TCGA somatic CN data as reference**.

`gap_cn` a gap copy number value. A gene with copy number above background (`ploidy` + `gap_cn` in general) would be treated as focal amplicon. Smaller, more amplicons.

`only_oncogenes` if `TRUE`, only known oncogenes are kept for circular prediction.

Value

a list of `data.table`.

Examples

```
data("ec")
ec2 <- ec
ec2$prob <- gcap.runPrediction(ec)
score <- gcap.runScoring(ec2)
score
```

`gcap.workflow` *GCAP workflow for gene-level amplicon prediction*

Description

GCAP workflow for gene-level amplicon prediction

Usage

```
gcap.workflow(
  tumourseqfile,
  normalseqfile,
  tumourname,
  normalname,
  jobname = tumourname,
  extra_info = NULL,
  include_type = FALSE,
  genome_build = c("hg38", "hg19"),
  model = "XGB11",
  tightness = 1L,
  gap_cn = 3L,
  overlap = 1,
  only_oncogenes = FALSE,
  outdir = getwd(),
  result_file_prefix = paste0("gcap_", uuid::UUIDgenerate(TRUE)),
  allelecounter_exe = "~/miniconda3/envs/cancerit/bin/alleleCounter",
  g1000allelesprefix = file.path("~/data/snp/1000G_loci_hg38",
    "1kg.phase3.v5a_GRCh38nounref_allele_index_chr"),
  g1000lociprefix = file.path("~/data/snp/1000G_loci_hg38",
    "1kg.phase3.v5a_GRCh38nounref_loci_chrstring_chr"),
  GCcontentfile = "~/data/snp/GC_correction_hg38.txt",
  replictimingfile = "~/data/snp/RT_correction_hg38.txt",
  nthreads = 22,
  minCounts = 10,
  BED_file = NA,
  probloci_file = NA,
  chrom_names = 1:22,
  min_base_qual = 20,
  min_map_qual = 35,
  penalty = 70,
  skip_finished_ASCAT = TRUE,
  skip_ascat_call = FALSE
)
```

Arguments

`tumourseqfile` Full path to the tumour BAM file.

`normalseqfile` Full path to the normal BAM file.

`tumourname` Identifier to be used for tumour output files.

`normalname` Identifier to be used for normal output files.

`jobname` job name, typically an unique name for a tumor-normal pair.

`extra_info` (optional) a (file containing) `data.frame` with 3 columns ’sample’ (must identical to the setting of parameter `jobname`), ’age’ and ’gender’. For gender, should be ’XX’ or ’XY’, also could be `0` for ’XX’ and `1` for ’XY’.

`include_type` if `TRUE`, a fourth column named ’type’ should be included in `extra_info`, the supported cancer type should be described with TCGA cancer type abbr..

`genome_build` "hg38" or "hg19".

`model` model name ("XGB11", "XGB32", "XGB56") or a custom model from input. ’toy’ can be used for test.

`tightness` a coefficient to times to TCGA somatic CN to set a more strict threshold as a circular amplicon. If the value is larger, it is more likely a fCNA assigned to `noncircular` instead of `circular`. **When it is `NA`, we don’t use TCGA somatic CN data as reference**.

`gap_cn` a gap copy number value. A gene with copy number above background (`ploidy + gap_cn` in general) would be treated as focal amplicon. Smaller, more amplicons.

`overlap` the overlap percentage on gene.

`only_oncogenes` if `TRUE`, only known oncogenes are kept for circular prediction.

`outdir` result output path.

`result_file_prefix`
file name prefix (without directory path) for storing final model prediction file in CSV format. Default a unique file name is generated by UUID approach.

`allelecounter_exe`
Path to the allele counter executable.

`g1000allelesprefix`
Prefix path to the allele data (e.g. "G1000_alleles_chr").

`g1000lociprefix`
Prefix path to the loci data (e.g. "G1000_loci_chr").

`GCcontentfile` File containing the GC content around every SNP for increasing window sizes.

`replictimingfile`
File containing replication timing at every SNP for various cell lines.

`nthreads` The number of parallel processes for getting allele counts (optional, default=1).

`minCounts` Minimum depth required in the normal for a SNP to be considered (optional, default=10).

`BED_file` A BED file for only looking at SNPs within specific intervals (optional, default=NA).

`probloci_file` A file (chromosome <tab> position; no header) containing specific loci to ignore (optional, default=NA).

`chrom_names` A vector containing the names of chromosomes to be considered (optional, default=1:22).

`min_base_qual` Minimum base quality required for a read to be counted (optional, default=20).

`min_map_qual` Minimum mapping quality required for a read to be counted (optional, default=35).

`penalty` penalty of introducing an additional ASPCF breakpoint (expert parameter, don't adapt unless you know what you're doing)

`skip_finished_ASCAT`
if `TRUE`, skipped finished ASCAT calls to save time.

`skip_ascat_call`
if `TRUE`, skip calling ASCAT. This is useful when you have done this step and just want to run next steps.

Value

a list of invisible `data.table` and corresponding files saved to local machine.

`gcap.workflow.facets` *GCAP FACETS workflow for gene-level amplicon prediction*

Description

GCAP FACETS workflow for gene-level amplicon prediction

Usage

```
gcap.workflow.facets(
  tumourseqfile,
  normalseqfile,
  jobname,
  extra_info = NULL,
  include_type = FALSE,
  genome_build = c("mm10", "hg38", "hg19"),
  model = "XGB11",
  tightness = 1L,
  gap_cn = 3L,
  overlap = 1,
  pro_cval = 100,
  only_oncogenes = FALSE,
  snp_file = "path/to/genome_build_responding.vcf.gz",
  outdir = getwd(),
  result_file_prefix = paste0("gcap_", uuid::UUIDgenerate(TRUE)),
  util_exe = system.file("extcode", "snp-pileup", package = "facets"),
  nthreads = 1,
  skip_finished_facets = TRUE,
  skip_facets_call = FALSE
)
```

Arguments

`tumourseqfile`	Full path to the tumour BAM file.
`normalseqfile`	Full path to the normal BAM file.
`jobname`	job name, typically an unique name for a tumor-normal pair.
`extra_info`	(optional) a (file containing) `data.frame` with 3 columns 'sample' (must identical to the setting of parameter `jobname`), 'age' and 'gender'. For gender, should be 'XX' or 'XY', also could be `0` for 'XX' and 1 for 'XY'.
`include_type`	if `TRUE`, a fourth column named 'type' should be included in `extra_info`, the supported cancer type should be described with TCGA cancer type abbr..
`genome_build`	genome build version, should be one of 'hg38', 'hg19' and 'mm10'.
`model`	model name ("XGB11", "XGB32", "XGB56") or a custom model from input. 'toy' can be used for test.

tightness a coefficient to times to TCGA somatic CN to set a more strict threshold as a circular amplicon. If the value is larger, it is more likely a fCNA assigned to noncircular instead of circular. **When it is NA, we don't use TCGA somatic CN data as reference**.

gap_cn a gap copy number value. A gene with copy number above background (ploidy + gap_cn in general) would be treated as focal amplicon. Smaller, more amplicons.

overlap the overlap percentage on gene.

pro_cval critical value for segmentation used in facets::procSample().

only_oncogenes if TRUE, only known oncogenes are kept for circular prediction.

snp_file a file path to SNP file of genome, should be consistent with genome_build option.

outdir result output path.

result_file_prefix
file name prefix (without directory path) for storing final model prediction file in CSV format. Default a unique file name is generated by UUID approach.

util_exe the path to snp-pileup.

nthreads The number of parallel processes for getting allele counts (optional, default=1).

skip_finished_facets
if TRUE, skip finished FACETS runs.

skip_facets_call
if TRUE, skip calling FACETS. This is useful when you have done this step and just want to run next steps.

Details

For generating the snp-pileup program, reference commands given here. You need modify corresponding path to fit your own machine.

```
cd /data3/wsx/R/x86_64-pc-linux-gnu-library/4.2/facets/extcode/
g++ -std=c++11 -I/data3/wsx/miniconda3/envs/circlemap/include snp-pileup.cpp -L/data3/wsx/miniconda3/
```

Value

a list of invisible data.table and corresponding files saved to local machine.

---

gcap.workflow.seqz *GCAP sequenza workflow for gene-level amplicon prediction*

Description

GCAP sequenza workflow for gene-level amplicon prediction

Usage

```
gcap.workflow.seqz(
  tumourseqfile,
  normalseqfile,
  jobname,
  extra_info = NULL,
  include_type = FALSE,
  genome_build = c("mm10", "hg38", "hg19"),
  model = "XGB11",
  tightness = 1L,
  gap_cn = 3L,
  overlap = 1,
  only_oncogenes = FALSE,
  ref_file = "path/to/reference.fa",
  data_tmp_dir = "~/gcap_data",
  outdir = getwd(),
  result_file_prefix = paste0("gcap_", uuid::UUIDgenerate(TRUE)),
  util_exe = "~/miniconda3/bin/sequenza-utils",
  samtools_exe = "~/miniconda3/bin/samtools",
  tabix_exe = "~/miniconda3/bin/tabix",
  nthreads = 1,
  skip_finished_sequenza = TRUE,
  skip_sequenza_call = FALSE
)
```

Arguments

`tumourseqfile` Full path to the tumour BAM file.

`normalseqfile` Full path to the normal BAM file.

`jobname` job name, typically an unique name for a tumor-normal pair.

`extra_info` (optional) a (file containing) `data.frame` with 3 columns 'sample' (must identical to the setting of parameter `jobname`), 'age' and 'gender'. For gender, should be 'XX' or 'XY', also could be `0` for 'XX' and `1` for 'XY'.

`include_type` if `TRUE`, a fourth column named 'type' should be included in `extra_info`, the supported cancer type should be described with TCGA cancer type abbr..

`genome_build` genome build version, should be one of 'hg38', 'hg19' and 'mm10'.

`model` model name ("XGB11", "XGB32", "XGB56") or a custom model from input. 'toy' can be used for test.

`tightness` a coefficient to times to TCGA somatic CN to set a more strict threshold as a circular amplicon. If the value is larger, it is more likely a fCNA assigned to `noncircular` instead of `circular`. **When it is `NA`, we don't use TCGA somatic CN data as reference**.

`gap_cn` a gap copy number value. A gene with copy number above background (`ploidy + gap_cn` in general) would be treated as focal amplicon. Smaller, more amplicons.

`overlap` the overlap percentage on gene.

`only_oncogenes` if `TRUE`, only known oncogenes are kept for circular prediction.

`ref_file` a reference genome file, should be consistent with `genome_build` option.

`data_tmp_dir` a directory path for storing temp data for reuse in handling multiple samples.

`outdir` result output path.

`result_file_prefix`
file name prefix (without directory path) for storing final model prediction file in CSV format. Default a unique file name is generated by UUID approach.

`util_exe` the path to `sequenza-utils`.

`samtools_exe` the path to `samtools_exe`.

`tabix_exe` the path to `tabix`.

`nthreads` The number of parallel processes for getting allele counts (optional, default=1).

`skip_finished_sequenza`
if `TRUE`, skip finished sequenza runs.

`skip_sequenza_call`
if `TRUE`, skip calling sequenza. This is useful when you have done this step and just want to run next steps.

Value

a list of invisible `data.table` and corresponding files saved to local machine.

`get_auc` *Get AUC value*

Description

Get AUC value

Usage

```
get_auc(y_pred, y, type = c("pr", "roc"), curve = FALSE)
```

Arguments

`y_pred` y prediction vector.

`y` y true label vector.

`type` AUC type, either 'pr' or 'roc'.

`curve` if `TRUE`, generate plot data, the result can be plotted by `plot()`.

Value

A object.

Examples

```
if (require("PRROC")) {
  set.seed(2021)
  auc <- get_auc(sample(1:10, 10), c(rep(0, 5), rep(1, 5)))
  auc
}
```

mergeDTs *Merge a list of data.table*

Description

Merge a list of data.table

Usage

```
mergeDTs(dt_list, by = NULL, sort = FALSE)
```

Arguments

`dt_list`	a list of `data.tables`.
`by`	which column used for merging.
`sort`	should sort the result?

Value

a `data.table`

oncogenes *Oncogene list*

Description

Oncogene list

Format

A `data.frame`

Source

Generate from `data-raw/`, raw source from `http://ongene.bioinfo-minzhao.org/`

Examples

```
data("oncogenes")
```

`overlaps` *Get overlaps of two genomic regions*

Description

Get overlaps of two genomic regions

Usage

```
overlaps(x, y)
```

Arguments

`x, y`	a genemic region with data.frame format, the first 3 columns should representing chromosome, start and end position.

Value

a `data.table`

Index