Package 'ZygosityPredictor'

November 7, 2025

Type Package

Title Package for prediction of zygosity for variants/genes in NGS data

Version 1.11.0

Date 2025-07-28

Imports GenomicAlignments, GenomicRanges, Rsamtools, IRanges, VariantAnnotation, DelayedArray, dplyr, stringr, purrr, tibble, methods, knitr, igraph, readr, stats, magrittr, rlang

License GPL-2

Description

The ZygosityPredictor allows to predict how many copies of a gene are affected by small variants. In addition to the basic calculations of the affected copy number of a variant, the ZygosityPredictor can integrate the influence of several variants on a gene and ultimately make a statement if and how many wild-type copies of the gene are left. This information proves to be of particular use in the context of translational medicine. For example, in cancer genomes, the ZygosityPredictor can address whether unmutated copies of tumor-suppressor genes are present. Beyond this, it is possible to make this statement for all genes of an organism. The ZygosityPredictor was primarily developed to handle SNVs and INDELs (later addressed as small-variants) of somatic and germline origin. In order not to overlook severe effects outside of the small-variant context, it has been extended with the assessment of large scale deletions, which cause losses of whole genes or parts of them.

RoxygenNote 7.2.3

Encoding UTF-8

biocViews BiomedicalInformatics, FunctionalPrediction, SomaticMutation, GenePrediction

Depends R (>= 4.3.0)

LazyData false

Suggests rmarkdown, testthat, BiocStyle

VignetteBuilder knitr

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

git_branch devel

git_last_commit 21ba67a

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2025-11-06

Author Marco Rheinnecker [aut, cre] (ORCID:
<https://orcid.org/0009-0009-7181-3977>),
Marc Ruebsam [aut],
Daniel Huebschmann [aut],
Martina Froehlich [aut],
Barbara Hutter [aut]

Maintainer Marco Rheinnecker <marco.rheinnecker@dkfz-heidelberg.de>

Contents

`aff_germ_copies` *calculates how many copies are affected by a germnline small variant*

Description

calculates how many copies are affected by a germnline small variant

Usage

```
aff_germ_copies(chr, af, tcn, purity, sex, c_normal = NULL, af_normal = 0.5)
```

Arguments

`chr`	chromosome of the variant (either format 1,2,..,X,Y or chr1,..,chrX)
`af`	Allele-frequency of the variant (numeric value between 0 and 1)
`tcn`	total-copynumber at position of the variant (numeric value >0)

purity	purity of the sample (numeric value between 0 and 1 indicating the fraction of relevant sample with control/unrelevant tissue)
sex	sex of the sample (character: "male", "female", "m", "f")
c_normal	expected copy number at position of the variant in normal tissue, 1 for gonosomes in male samples, and 2 for male autosomes and all chromosomes in female samples. (The function can also assess the c_normal parameter by itself, but then the following two inputs must be provided: chr and sex)
af_normal	Allele-frequency in normal tissue (numeric value between 0 and 1) 0.5 represents heterozygous variants in diploid genome, 1 would be homozygous. Could be relevant if germline CNVs are present at the position. Then also the c_normal parameter would have to be adjusted.

Value

A numeric value indicating the affecting copies for the variant

Examples

```
library(dplyr)
library(purrr)
library(stringr)
aff_germ_copies(af=0.67, tcn=2, purity=0.9, chr="chrX", sex="female")
```

---

aff_som_copies *calculates how many copies are affected by a somatic small variant*

Description

calculates how many copies are affected by a somatic small variant

Usage

```
aff_som_copies(chr, af, tcn, purity, sex, c_normal = NULL)
```

Arguments

chr	chromosome of the variant (either format 1,2,..,X,Y or chr1,..,chrX)
af	Allele-frequency of the variant (numeric value between 0 and 1)
tcn	total-copynumber at position of the variant (numeric value >0)
purity	purity of the sample (numeric value between 0 and 1 indicating the fraction of relevant sample with control/unrelevant tissue)
sex	sex of the sample (character: "male", "female", "m", "f")
c_normal	expected copy number at the position of the variant in normal tissue, 1 for gonosomes in male samples, and 2 for male autosomes and all chromosomes in female samples. (The function can also assess the c_normal parameter by itself, but then the following two inputs must be provided: chr and sex)

Value

A numeric value indicating the affecting copies for the variant

Examples

```
library(dplyr)
library(purrr)
library(stringr)
aff_som_copies(chr="chrX", af=0.67, tcn=2, purity=0.9, sex="female")
```

---

gene_ov *accesor for gene predictions printing detailed info about how a gene status was assigned*

Description

accesor for gene predictions printing detailed info about how a gene status was assigned

Usage

```
gene_ov(fp, inp_gene, n = 20)
```

Arguments

fp	full prediction (output of predict_zygoisty())
inp_gene	name of gene that should be printed with detailed information
n	max number of rows to print, as some gene status depend on loads of phasing results#'

Value

prints overview about run from function predict_zygoisty() with specific information about provided gene

Examples

```
cnvs  = GenomicRanges::GRanges(
  dplyr::tibble(
    chr = "chr17",
    start = c(170060, 34520990),
    end = c(34520990, 83198614),
    tcn = c(2, 1),
    cna_type = c("neutral", "LOH")
  )
)
somatic_vars = GenomicRanges::GRanges(
  dplyr::tibble(
    chr="chr17",
```

```
    start = 7675088,
    end = 7675088,
    ref = "C",
    alt = "T",
    af = 0.65,
    gene = "TP53"
  )
)
germline_vars = GenomicRanges::GRanges(
  dplyr::tibble(
    chr="chr17",
    start = 41771694,
    end = 41771694,
    ref = "GTGT",
    alt = "G",
    af = 0.95,
    gene = "JUP"
  )
)
reference = GenomicRanges::GRanges(
  dplyr::tibble(
    chr = "chr17",
    start = c(7661778, 41754603),
    end = c(7687538, 41786931),
    gene = c("TP53", "JUP")
  )
)
sex = "female"
purity = 0.9
bamfile <- system.file("extdata", "ZP_example.bam",
  package = "ZygosityPredictor")
fp <- predict_zygosity(purity = purity, sex = sex,
  somCna = cnvs,
  somSmallVars = somatic_vars,
  germSmallVars = germline_vars,
  geneModel = reference,
  bamDna = bamfile
)
gene_ov(fp, TP53)
```

GR_GENE_MODEL *germline small variant object*

Description

germline small variant object

Usage

```
data(GR_GENE_MODEL)
```

Format

'GR_GENE_MODEL' GRanges object

Value

Object containing gene model of hg38

GR_GERM_SMALL_VARS *germline small variant object*

Description

germline small variant object

Usage

```
data(GR_SOM_SMALL_VARS)
```

Format

'GR_SOM_SMALL_VARS' GRanges object

Value

Object containing germline Indels and SNVs of SeqC2 example case

---

GR_HAPLOBLOCKS *haploblocks*

Description

haploblocks

Usage

```
data(GR_HAPLOBLOCKS)
```

Format

'GR_HAPLOBLOCKS' GRanges object

Value

Object containing haploblock annotations

GR_SCNA *copynumber object*

Description

copynumber object

Usage

```
data(GR_SCNA)
```

Format

'GR_SCNA' GRanges object

Value

Object containing somatic copy number abberations (sCNAs) of SeqC2 example case

GR_SOM_SMALL_VARS *somatic small variant object*

Description

somatic small variant object

Usage

```
data(GR_GERM_SMALL_VARS)
```

Format

'GR_GERM_SMALL_VARS' GRanges object

Value

Object containing somatic Indels and SNVs of SeqC2 example case

predict_per_variant *predicts zygosity of a set of variants*

Description

predicts zygosity of a set of variants

Usage

```
predict_per_variant(
  purity,
  sex,
  somCna,
  geneModel = NULL,
  somSmallVars = NULL,
  germSmallVars = NULL,
  ploidy = NULL,
  colnameTcn = NULL,
  colnameCnaType = NULL,
  includeHomoDel = TRUE,
  includeIncompleteDel = TRUE,
  assumeSomCnaGaps = FALSE,
  byTcn = TRUE,
  ZP_env = NULL,
  verbose = FALSE
)
```

Arguments

purity	purity of the sample (numeric value between 0 and 1 indicating the fraction of relevant sample with control/unrelevant tissue)
sex	sex of the sample (character: "male", "female", "m", "f")
somCna	GRanges object containing all genomic regions with annotated total copynumber and cna_type as metadata columns. The total-copynumber column should be named "tcn" but also some other commonly used names. It should contain numeric values or characters that can be converted to numeric values. The cna_type column must contain the information about loss of heterozygosity (LOH). Therefore the term "LOH" must be explicitely mentioned in the column. If a genomic region is not present in the object, it will be taken as heterozygous with neutral TCN of 2.
geneModel	GRanges object containing the gene-annoattion of the used reference genome with metadata column of the gene name (gene)
somSmallVars	GRanges object containing all somatic small variants (SNV and INDEL). Required metadata columns are reference base (ref/REF), alternative base (alt/ALT), annotation of the gene name (gene/GENE) and the allele-frequency (af/AF). If the object is not provided the tool assumes there are no somatic small variants.

germSmallVars	GRanges object containing all germline small variants (SNV and INDEL). Required metadata columns are reference base (ref/REF), alternative base (alt/ALT), annotation of the gene name (gene/GENE) and the allele-frequency (af/AF) If the object is not provided the tool assumes there are no germline small variants.
ploidy	ploidy of the sample (numeric value)
colnameTcn	character indicating the name of the metadata containing the tcn information in the somCna object. If not provided the tool tries to detect the column according to default names
colnameCnaType	character indicating the name of the metadata containing cna type information in the somCna object. If not provided the tool tries to detect the column according to default names
includeHomoDel	default = TRUE; if FALSE homozygous deleteions are excluded
includeIncompleteDel	default = TRUE; if FALSE heterzygous deleteions are excluded
assumeSomCnaGaps	(logical, default=FALSE) Only required if the somCna object lacks copy number information for genomic segments on which small variants are detected. By default, variants in such regions will be excluded from the analysis as required information about the copy number is missing. These variants will be attached to the final output list in a separate tibble. To include them, this flag must be set TRUE and the ground ploidy must be given as an input. This ground ploidy will then be taken as tcn in the missing regions. If no ploidy is given the tool will assume the ground ploidy of 2 when this flag is TRUE.
byTcn	logical, default=TRUE; optional if includeHomoDel or includeIncompleteDelS is TRUE. If FALSE the tool will not use tcn as a criterion to assign large deletions. It will use the cna_type column and check for indicating strings like HOMDEL/HomoDel/DEL. Some commonly used strings are covered. It is recommended to leave this flag TRUE
ZP_env	internal variable... not recommened to be changed by user
verbose	logical, default=FALSE; prints functions that are called

Value

A list containing tibbles with all input variants

Examples

```
cnvs  = GenomicRanges::GRanges(
  dplyr::tibble(
    chr = "chr17",
    start = c(170060, 34520990),
    end = c(34520990, 83198614),
    tcn = c(2, 1),
    cna_type = c("neutral", "LOH")
  )
)
somatic_vars = GenomicRanges::GRanges(
```

```
  dplyr::tibble(
    chr="chr17",
    start = 7675088,
    end = 7675088,
    ref = "C",
    alt = "T",
    af = 0.65,
    gene = "TP53"
  )
)
germline_vars = GenomicRanges::GRanges(
  dplyr::tibble(
    chr="chr17",
    start = 41771694,
    end = 41771694,
    ref = "GTGT",
    alt = "G",
    af = 0.95,
    gene = "JUP"
  )
)
reference = GenomicRanges::GRanges(
  dplyr::tibble(
    chr = "chr17",
    start = c(7661778, 41754603),
    end = c(7687538, 41786931),
    gene = c("TP53", "JUP")
  )
)
sex = "female"
purity = 0.9
predict_per_variant(purity = purity, sex = sex,
  somCna = cnvs,
  somSmallVars = somatic_vars,
  germSmallVars = germline_vars,
  geneModel = reference
)
```

`predict_zygosity` *predicts zygosity of a set of genes of a sample*

Description

predicts zygosity of a set of genes of a sample

Usage

```
predict_zygosity(
  purity,
  sex,
```

```
  somCna,
  geneModel,
  bamDna,
  somSmallVars = NULL,
  germSmallVars = NULL,
  bamRna = NULL,
  ploidy = NULL,
  colnameTcn = NULL,
  colnameCnaType = NULL,
  includeHomoDel = TRUE,
  includeIncompleteDel = TRUE,
  showReadDetail = FALSE,
  printLog = FALSE,
  assumeSomCnaGaps = FALSE,
  byTcn = TRUE,
  vcf = NULL,
  haploBlocks = NULL,
  distCutOff = 5000,
  verbose = FALSE,
  debug = FALSE,
  logDir = NULL,
  snpQualityCutOff = 1,
  phasingMode = "fast",
  AllelicImbalancePhasing = FALSE
)
```

Arguments

purity	purity of the sample (numeric value between 0 and 1 indicating the fraction of relevant sample with control/unrelevant tissue)
sex	sex of the sample (character: "male", "female", "m", "f")
somCna	GRanges object containing all genomic regions with annotated total copynumber and cna_type as metadata columns. The total-copynumber column should be named "tcn" but also some other commonly used names. It should contain numeric values or characters that can be converted to numeric values. The cna_type column must contain the information about loss of heterozygosity (LOH). Therefore the term "LOH" must be explicitely mentioned in the column. If a genomic region is not present in the object, it will be taken as heterozygous with neutral TCN of 2.
geneModel	GRanges object containing the gene-annoattion of the used reference genome with metadata column of the gene name (gene)
bamDna	path to bam-file
somSmallVars	GRanges object containing all somatic small variants (SNV and INDEL). Required metadata columns are reference base (ref/REF), alternative base (alt/ALT), annotation of the gene name (gene/GENE) and the allele-frequency (af/AF). If the object is not provided the tool assumes there are no somatic small variants.

`germSmallVars` GRanges object containing all germline small variants (SNV and INDEL). Required metadata columns are reference base (ref/REF), alternative base (alt/ALT), annotation of the gene name (gene/GENE) and the allele-frequency (af/AF) If the object is not provided the tool assumes there are no germline small variants.

`bamRna` optional; path to rna file (bam format)

`ploidy` ploidy of the sample (numeric value)

`colnameTcn` character indicating the name of the metadata containing the tcn information in the somCna object. If not provided the tool tries to detect the column according to default names

`colnameCnaType` character indicating the name of the metadata containing cna type information in the somCna object. If not provided the tool tries to detect the column according to default names

`includeHomoDel` default = TRUE; if FALSE homozygous deleteions are excluded

`includeIncompleteDel`
default = TRUE; if FALSE heterzygous deleteions are excluded

`showReadDetail` default = FALSE; if TRUE a table is added to the output, containing all used reads/rea-pairs with anntated read classification (mut1, mut2, both, none, skipped, dev_var)

`printLog` default = FALSE; if TRUE the gene which is evaluated is printed in console, containing the query-name of each read which was used to perform haplotype-phasing and the info into which class it was assigned.

`assumeSomCnaGaps`
(logical, default=FALSE) Only required if the somCna object lacks copy number information for genomic segments on which small variants are detected. By default, variants in such regions will be excluded from the analysis as required information about the copy number is missing. These variants will be attached to the final output list in a separate tibble. To include them, this flag must be set TRUE and the ground ploidy must be given as an input. This ground ploidy will then be taken as tcn in the missing regions. If no ploidy is given the tool will assume the ground ploidy of 2 when this flag is TRUE.

`byTcn` logical, default=TRUE; optional if includeHomoDel or includeIncompleteDelS is TRUE. If FALSE the tool will not use tcn as a criterion to assign large deletions. It will use the cna_type column and check for indicating strings like HOMDEL/HomoDel/DEL. Some commonly used strings are covered. It is recommended to leave this flag TRUE

`vcf` character; path to variant call file (.vcf.gz format). Will be used (if provided) for extended SNP phasing if variants on the same gene are too far away from each other for direct haplotype phasing

`haploBlocks` GRanges object containing haploblocks. Haploblocks are defined as genomic regions in which SNPs are phased to a specific allele. For example a haploblock could be chr1:1000-10000. This would mean that every genotype annotation in the format "1|0" or "0|1" of a SNP in this region will be used to phase somatic variants and define their genotype

`distCutOff` numeric, default=5000; if input vcf is provided and SNP phasing is performed, this will limt the distance at which the SNP phasing should not be tried anymore.

As the probability of finding overlapping reads at such a long distance is very low and the runtime will increase exponentially.

`verbose` logical, default=FALSE; prints functions that are called

`debug` logical, default=FALSE; prints output for debugging

`logDir` character; path to directory where logfiles and detailed infos of the run can be stored, if not given, no details will be stored or printed

`snpQualityCutOff`
numeric, default=1; Cutoff to filter for SNPS that can be used for phasing

`phasingMode` character, default="fast"; if set to full. Even if high confidence phasing result could be achieved, following phasing approaches will be carried out

`AllelicImbalancePhasing`
logical, default=FALSE. Enables alleleic imbalance phasing if TRUE

Value

A list of dataframes. Those are the evaluation per variant, the evaluation per gene and, if performed, the info about the haplotype-phasing.

Examples

```
cnvs  = GenomicRanges::GRanges(
  dplyr::tibble(
    chr = "chr17",
    start = c(170060, 34520990),
    end = c(34520990, 83198614),
    tcn = c(2, 1),
    cna_type = c("neutral", "LOH")
  )
)
somatic_vars = GenomicRanges::GRanges(
  dplyr::tibble(
    chr="chr17",
    start = 7675088,
    end = 7675088,
    ref = "C",
    alt = "T",
    af = 0.65,
    gene = "TP53"
  )
)
germline_vars = GenomicRanges::GRanges(
  dplyr::tibble(
    chr="chr17",
    start = 41771694,
    end = 41771694,
    ref = "GTGT",
    alt = "G",
    af = 0.95,
    gene = "JUP"
  )
```

```
)
reference = GenomicRanges::GRanges(
  dplyr::tibble(
    chr = "chr17",
    start = c(7661778, 41754603),
    end = c(7687538, 41786931),
    gene = c("TP53", "JUP")
  )
)
sex = "female"
purity = 0.9
bamfile <- system.file("extdata", "ZP_example.bam",
  package = "ZygosityPredictor")
predict_zygosity(purity = purity, sex = sex,
  somCna = cnvs,
  somSmallVars = somatic_vars,
  germSmallVars = germline_vars,
  geneModel = reference,
  bamDna = bamfile
)
```

ZP_ov *accesor for ZygoistyPredictor runs. Prints an overview about the run*

Description

accesor for ZygoistyPredictor runs. Prints an overview about the run

Usage

```
ZP_ov(fp)
```

Arguments

fp	full prediction (output of predict_zygoisty())

Value

prints overview about run from function predict_zygoisty()

Examples

```
cnvs  = GenomicRanges::GRanges(
  dplyr::tibble(
    chr = "chr17",
    start = c(170060, 34520990),
    end = c(34520990, 83198614),
    tcn = c(2, 1),
    cna_type = c("neutral", "LOH")
  )
```

```
)
somatic_vars = GenomicRanges::GRanges(
  dplyr::tibble(
    chr="chr17",
    start = 7675088,
    end = 7675088,
    ref = "C",
    alt = "T",
    af = 0.65,
    gene = "TP53"
  )
)
germline_vars = GenomicRanges::GRanges(
  dplyr::tibble(
    chr="chr17",
    start = 41771694,
    end = 41771694,
    ref = "GTGT",
    alt = "G",
    af = 0.95,
    gene = "JUP"
  )
)
reference = GenomicRanges::GRanges(
  dplyr::tibble(
    chr = "chr17",
    start = c(7661778, 41754603),
    end = c(7687538, 41786931),
    gene = c("TP53", "JUP")
  )
)
sex = "female"
purity = 0.9
bamfile <- system.file("extdata", "ZP_example.bam",
  package = "ZygosityPredictor")
fp <- predict_zygosity(purity = purity, sex = sex,
  somCna = cnvs,
  somSmallVars = somatic_vars,
  germSmallVars = germline_vars,
  geneModel = reference,
  bamDna = bamfile
)
ZP_ov(fp)
```

Index