Package 'PureCN'

April 12, 2018

Type Package

Title Copy number calling and SNV classification using targeted short
read sequencing

Version 1.8.1

Date 2017-12-09

Description This package estimates tumor purity, copy number, and loss of
heterozygosity (LOH), and classifies single nucleotide variants (SNVs) by
somatic status and clonality. PureCN is designed for targeted short read
sequencing data, integrates well with standard somatic variant detection
and copy number pipelines, and has support for tumor samples without
matching normal samples.

Depends R (>= 3.3), DNAcopy, VariantAnnotation (>= 1.14.1)

Imports GenomicRanges (>= 1.20.3), IRanges (>= 2.2.1), RColorBrewer,
S4Vectors, data.table, grDevices, graphics, stats, utils,
SummarizedExperiment, GenomeInfoDb, GenomicFeatures, Rsamtools,
Biostrings, BiocGenerics, rtracklayer, ggplot2, futile.logger,
VGAM, edgeR, tools, limma

Suggests PSCBS, RUnit, BiocStyle, knitr, optparse,
TxDb.Hsapiens.UCSC.hg19.knownGene, org.Hs.eg.db

VignetteBuilder knitr

License Artistic-2.0

URL https://github.com/lima1/PureCN

biocViews CopyNumberVariation, Software, Sequencing,
VariantAnnotation, VariantDetection, Coverage

NeedsCompilation no

ByteCompile yes

RoxygenNote 6.0.1

Author Markus Riester [aut, cre],
Angad P. Singh [aut]

Maintainer Markus Riester <markus.riester@novartis.com>

1

R **topics documented:**

---

annotateTargets *Annotate targets with gene symbols*

Description

This function can be used to add a 'Gene' meta column containing gene symbols to a `GRanges` object. It applies heuristics to find the protein coding genes that were likely meant to target in the assay design in case transcripts overlap.

Usage

```
annotateTargets(x, txdb, org)
```

Arguments

x	A `GRanges` object with interals to annotate
txdb	A `TxDb` database, e.g. `TxDb.Hsapiens.UCSC.hg19.knownGene`
org	A `OrgDb` object, e.g. `org.Hs.eg.db`.

Value

A `GRanges` object.

Author(s)

Markus Riester

Examples

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
library(org.Hs.eg.db)

normal.coverage.file <- system.file("extdata", "example_normal.txt",
    package="PureCN")
x <- head(readCoverageFile(normal.coverage.file),100)
x <- annotateTargets(x,TxDb.Hsapiens.UCSC.hg19.knownGene, org.Hs.eg.db)
```

---

autoCurateResults *Heuristics to find the best purity/ploidy solution.*

Description

This function is defunct.

Usage

```
autoCurateResults()
```

Author(s)

Markus Riester

bootstrapResults	*Filter unlikely purity/ploidy solutions*

Description

This function bootstraps SNVs, then re-ranks solutions by using the bootstrap estimate of the likelihood score, and then keeps only solutions that were ranked highest in any bootstrap replicate. Large-scale copy number artifacts can cause true purity/ploidy solutions rank low.

Usage

```
bootstrapResults(res, n = 500, top = 2)
```

Arguments

res	Return object of the [runAbsoluteCN](#) function.
n	Number of bootstrap replicates.
top	Include solution if it appears in the top n solutions of any bootstrap replicate.

Value

Returns the [runAbsoluteCN](#) object with low likelihood solutions removed. Also adds a bootstrap value to each solution. This value is the fraction of bootstrap replicates in which the solution ranked first.

Author(s)

Markus Riester

See Also

[runAbsoluteCN](#)

Examples

```
data(purecn.example.output)
ret.boot <- bootstrapResults(purecn.example.output, n=100)
plotAbs(ret.boot, type="overview")
```

calculateBamCoverageByInterval

Function to calculate coverage from BAM file

Description

Takes a BAM file and an interval file as input and returns coverage for each interval. Coverage should be then GC-normalized using the [correctCoverageBias](#) function before determining purity and ploidy with [runAbsoluteCN](#). Uses the scanBam function and applies low quality, duplicate reads as well as secondary alignment filters.

Usage

```
calculateBamCoverageByInterval(bam.file, interval.file, output.file = NULL,
    index.file = bam.file, keep.duplicates = FALSE)
```

Arguments

`bam.file`	Filename of a BAM file.
`interval.file`	File specifying the intervals. Interval is expected in first column in format CHR:START-END. The `gc.gene.file` can be used.
`output.file`	Optionally, write minimal coverage file. Can be read with the [readCoverageFile](#) function.
`index.file`	The bai index. This is expected without the .bai file suffix, see ?scanBam.
`keep.duplicates`	
	Keep or remove duplicated reads.

Value

Returns total and average coverage by intervals.

Author(s)

Markus Riester

See Also

[calculateGCContentByInterval](#) [correctCoverageBias](#) [runAbsoluteCN](#)

Examples

```
bam.file <- system.file("extdata", "ex1.bam", package="PureCN",
    mustWork = TRUE)
interval.file <- system.file("extdata", "ex1_intervals.txt",
    package="PureCN", mustWork = TRUE)

# Calculate raw coverage from BAM file. These need to be corrected for GC-bias
# using the correctCoverageBias function before determining purity and ploidy.
coverage <- calculateBamCoverageByInterval(bam.file=bam.file,
    interval.file=interval.file)
```

calculateGCContentByInterval   *Calculates GC content by interval*

Description

Uses scanFa from the Rsamtools package to retrieve GC content of intervals in a reference FASTA file. Can optimize intervals for copy number calling by tiling long intervals and by including off-target regions. This optimization largely follows CNVkit.

Usage

```
calculateGCContentByInterval(interval.file, reference.file,
  output.file = NULL, off.target = FALSE, average.target.width = 400,
  min.off.target.width = 20000, average.off.target.width = 2e+05,
  off.target.padding = -500, mappability = NULL, min.mappability = c(0.5,
  0.1, 0.7), off.target.seqlevels = c("targeted", "all"))
```

Arguments

interval.file	File specifying the intervals. Interval is expected in first column in format CHR:START-END. Instead of a file, a GRanges object can be provided. This allows the use of BED files for example. Note that GATK interval files are 1-based (first position of the genome is 1). Other formats like BED files are often 0-based. The import function will automatically convert to 1-based GRanges.
reference.file	Reference FASTA file.
output.file	Optionally, write GC content file.
off.target	Include off-target regions.
average.target.width	
	Split large targets to approximately this size.
min.off.target.width	
	Only include off-target regions of that size
average.off.target.width	
	Split off-target regions to that
off.target.padding	
	Pad off-target regions.
mappability	Annotate intervals with mappability score. Assumed on a scale from 0 to 1, with score being 1/(number alignments). Expected as GRanges object with first meta column being the score. Regions outside these ranges are ignored, assuming that mappability covers the whole accessible genome.
min.mappability	
	double(3) specifying the minimum mappability score for on-target, off-target, and chrY regions in that order. The chrY regions are only used for sex determination in 'PureCN' and are therefore treated differently. Requires mappability.
off.target.seqlevels	
	Controls how to deal with chromosomes/contigs found in the reference.file but not in the interval.file.

Value

Returns GC content by interval as GRanges object.

Author(s)

Markus Riester

References

Talevich et al. (2016). CNVkit: Genome-Wide Copy Number Detection and Visualization from Targeted DNA Sequencing. PLoS Comput Biol.

Examples

```
reference.file <- system.file("extdata", "ex2_reference.fa",
    package="PureCN", mustWork = TRUE)
interval.file <- system.file("extdata", "ex2_intervals.txt",
    package="PureCN", mustWork = TRUE)
bed.file <- system.file("extdata", "ex2_intervals.bed",
    package="PureCN", mustWork = TRUE)
calculateGCContentByInterval(interval.file, reference.file,
    output.file="gc_file.txt")

intervals <- import(bed.file)
calculateGCContentByInterval(intervals, reference.file,
    output.file="gc_file.txt")
```

---

calculateLogRatio	*Calculate coverage log-ratio of tumor vs. normal*

Description

This function is automatically called by [runAbsoluteCN](#) when normal and tumor coverage are provided (and not a segmentation file or target-level log-ratios). This function is therefore normally not called by the user.

Usage

```
calculateLogRatio(normal, tumor)
```

Arguments

normal	Normal coverage read in by the [readCoverageFile](#) function.
tumor	Tumor coverage read in by the [readCoverageFile](#) function.

Value

numeric(length(tumor)), tumor vs. normal copy number log-ratios for all targets.

Author(s)

Markus Riester

Examples

```
normal.coverage.file <- system.file("extdata", "example_normal.txt",
    package="PureCN")
tumor.coverage.file <- system.file("extdata", "example_tumor.txt",
    package="PureCN")
normal <- readCoverageFile(normal.coverage.file)
tumor <- readCoverageFile(tumor.coverage.file)
log.ratio <- calculateLogRatio(normal, tumor)
```

---

calculatePowerDetectSomatic

Power calculation for detecting somatic mutations

Description

This function calculates the probability of correctly rejecting the null hypothesis that an alt allele is a sequencing error rather than a true (mono-)clonal mutation.

Usage

```
calculatePowerDetectSomatic(coverage, f = NULL, purity = NULL,
  ploidy = NULL, cell.fraction = 1, error = 0.001, fpr = 5e-07,
  verbose = TRUE)
```

Arguments

coverage	Mean sequencing coverage.
f	Mean expected allelic fraction. If NULL, requires purity and ploidy and then calculates the expected fraction.
purity	Purity of sample. Only required when f is NULL.
ploidy	Ploidy of sample. Only required when f is NULL.
cell.fraction	Fraction of cells harboring mutation. Ignored if f is not NULL.
error	Estimated sequencing error rate.
fpr	Required false positive rate for mutation vs. sequencing error.
verbose	Verbose output.

Value

A list with elements

power	Power to detect somatic mutations.
k	Minimum number of supporting reads.
f	Expected allelic fraction.

Author(s)

Markus Riester

References

Carter et al. (2012), Absolute quantification of somatic DNA alterations in human cancer. Nature Biotechnology.

Examples

```
purity <- c(0.1,0.15,0.2,0.25,0.4,0.6,1)
coverage <- seq(5,35,1)
power <- lapply(purity, function(p) sapply(coverage, function(cv)
    calculatePowerDetectSomatic(coverage=cv, purity=p, ploidy=2,
    verbose=FALSE)$power))

# Figure S7b in Carter et al.
plot(coverage, power[[1]], col=1, xlab="Sequence coverage",
    ylab="Detection power", ylim=c(0,1), type="l")

for (i in 2:length(power)) lines(coverage, power[[i]], col=i)
abline(h=0.8, lty=2, col="grey")
legend("bottomright", legend=paste("Purity", purity), fill=seq_along(purity))

# Figure S7c in Carter et al.
coverage <- seq(5,350,1)
power <- lapply(purity, function(p) sapply(coverage, function(cv)
    calculatePowerDetectSomatic(coverage=cv, purity=p, ploidy=2,
        cell.fraction=0.2, verbose=FALSE)$power))
plot(coverage, power[[1]], col=1, xlab="Sequence coverage",
    ylab="Detection power", ylim=c(0,1), type="l")

for (i in 2:length(power)) lines(coverage, power[[i]], col=i)
abline(h=0.8, lty=2, col="grey")
legend("bottomright", legend=paste("Purity", purity), fill=seq_along(purity))
```

---

callAlterations   *Calling of amplifications and deletions*

Description

Function to extract major copy number alterations from a runAbsoluteCN return object.

Usage

```
callAlterations(res, id = 1, cutoffs = c(0.5, 6, 7),
  log.ratio.cutoffs = c(-0.9, 0.9), failed = NULL, all.genes = FALSE)
```

Arguments

res	Return object of the [runAbsoluteCN](#) function.
id	Candidate solutions to be used. id=1 will use the maximum likelihood (or curated) solution.
cutoffs	Copy numbers cutoffs to call losses, focal amplifications and broad amplifications.
log.ratio.cutoffs	
	Copy numbers log-ratio cutoffs to call losses and amplifications in failed samples.
failed	Indicates whether sample was failed. If NULL, use available annotation, which can be set in the curation file.
all.genes	If FALSE, then only return amplifications and deletions passing the thresholds.

Value

A data.frame with gene-level amplification and deletion calls.

Author(s)

Markus Riester

See Also

[runAbsoluteCN](#)

Examples

```
data(purecn.example.output)
callAlterations(purecn.example.output)
callAlterations(purecn.example.output, all.genes=TRUE)["ESR2",]
```

---

callAlterationsFromSegmentation
Calling of amplifications and deletions from segmentations

Description

This function can be used to obtain gene-level copy number calls from segmentations. This is useful for comparing PureCN's segmentations with segmentations obtained by different tools on the gene-level. Segmentation file can contain multiple samples.

Usage

```
callAlterationsFromSegmentation(sampleid, chr, start, end, num.mark = NA,
  seg.mean, C, gc.gene.file, fun.focal = findFocal, args.focal = list(),
  ...)
```

Arguments

sampleid	The sampleid column in the segmentation file.
chr	The chromosome column.
start	The start positions of the segments.
end	The end positions of the segments.
num.mark	Optionally, the number of probes or markers in each segment.
seg.mean	The segment mean.
C	The segment integer copy number.
gc.gene.file	A mapping file that assigns GC content and gene symbols to each exon in the coverage files. Used for generating gene-level calls. First column in format CHR:START-END. Second column GC content (0 to 1). Third column gene symbol. This file can be generated with the 'GATK GCContentByInterval' tool or with the calculateGCContentByInterval function.
fun.focal	Function for identifying focal amplifications. Defaults to findFocal.
args.focal	Arguments for focal amplification function.
...	Arguments passed to callAlterations.

Value

A list of callAlterations data.frame objects, one for each sample.

Author(s)

Markus Riester

Examples

```
data(purecn.example.output)
seg <- purecn.example.output$results[[1]]$seg
gc.gene.file <- system.file("extdata", "example_gc.gene.file.txt",
        package = "PureCN")

calls <- callAlterationsFromSegmentation(sampleid=seg$ID, chr=seg$chrom,
    start=seg$loc.start, end=seg$loc.end, num.mark=seg$num.mark,
    seg.mean=seg$seg.mean, C=seg$C, gc.gene.file=gc.gene.file)
```

---

callLOH	*Get regions of LOH*

Description

This function provides detailed LOH information by region.

Usage

```
callLOH(res, id = 1, arm.cutoff = 0.9)
```

Arguments

res	Return object of the [runAbsoluteCN](#) function.
id	Candidate solution to extract LOH from. id=1 will use the maximum likelihood solution.
arm.cutoff	Min fraction LOH on a chromosome arm to call whole arm events.

Value

Returns data.frame with LOH regions.

Author(s)

Markus Riester

See Also

[runAbsoluteCN](#)

Examples

```
data(purecn.example.output)
head(callLOH(purecn.example.output))
```

---

callMutationBurden *Call mutation burden*

Description

This function provides detailed mutation burden information.

Usage

```
callMutationBurden(res, id = 1, remove.flagged = TRUE,
  min.prior.somatic = 0.1, min.cellfraction = 0,
  fun.countMutation = function(vcf) width(vcf) == 1, callable = NULL,
  exclude = NULL)
```

Arguments

res	Return object of the [runAbsoluteCN](#) function.
id	Candidate solution to extract mutation burden from. id=1 will use the maximum likelihood solution.
remove.flagged	Remove variants flagged by [predictSomatic](#).
min.prior.somatic	
	Exclude variants with somatic prior probability lower than this cutoff.
min.cellfraction	
	Exclude variants with cellular fraction lower than this cutoff. These are subclonal mutations or artifacts with very low allelic fraction.

fun.countMutation

        Function that can be used to filter the input VCF further for filtering, for example to only keep missense mutations. Expects a `logical` vector indicating whether variant should be counted (`TRUE`) or not (`FALSE`). Default is to keep only single nucleotide variants.

callable   GRanges object with callable genomic regions, for example obtained by 'GATK CallableLoci' BED file, imported with `rtracklayer`.

exclude   GRanges object with genomic regions that should be excluded from the `callable` regions, for example intronic regions. Requires `callable`.

Value

Returns `data.frame` with mutation counts and sizes of callable regions.

Author(s)

Markus Riester

See Also

[runAbsoluteCN](#) [predictSomatic](#)

Examples

```
data(purecn.example.output)
callMutationBurden(purecn.example.output)

# To calculate exact mutations per megabase, we can provide a BED
# file containing all callable regions
callableBed <- import(system.file("extdata", "example_callable.bed.gz",
    package = "PureCN"))

# We can exclude some regions for mutation burden calculation,
# for example intronic regions.
exclude <- GRanges(seqnames="chr1", IRanges(start=1,
    end=max(end(callableBed))))

# We can also exclude specific mutations by filtering the input VCF
myVcfFilter <- function(vcf) seqnames(vcf)!="chr2"

callsCallable <- callMutationBurden(purecn.example.output,
    callable=callableBed, exclude=exclude, fun.countMutation=myVcfFilter)
```

---

centromeres   *A list of data.frames containing centromere positions.*

Description

A list of data.frames containing centromere positions for hg18, hg19 and hg38. Downloaded from the UCSC genome browser.

Usage

```
data(centromeres)
```

Value

A list with three data frames, "hg18", "hg19", and "hg38". Each containes three columns

chrom   a factor with levels `chr1 chr10 chr11 chr12 chr13 chr14 chr15 chr16 chr17 chr18 chr19` `chr2 chr20 chr21 chr22 chr3 chr4 chr5 chr6 chr7 chr8 chr9 chrX chrY`

chromStart   a numeric vector

chromEnd   a numeric vector

References

The script downloadCentromeres.R in the extdata directory was used to generate the data.frames.

Examples

```
data(centromeres)
```

---

correctCoverageBias   *Correct for GC bias*

Description

Takes as input coverage data and a mapping file for GC content. Will then normalize coverage data for GC-bias. Optionally plots the pre and post normalization GC profiles.

Usage

```
correctCoverageBias(coverage.file, gc.gene.file, output.file = NULL,
  method = c("LOESS", "POLYNOMIAL"), plot.gc.bias = FALSE,
  plot.max.density = 50000)
```

Arguments

coverage.file   Coverage file or coverage data parsed with the [readCoverageFile](#) function.

gc.gene.file   File providing GC content for each exon in the coverage files. First column in format CHR:START-END. Second column GC content (0 to 1). Third column provides gene symbols, which are optional, but used in [runAbsoluteCN](#) to generate gene level calls. This file can be generated with GATK GCContentBy-Interval tool or with the [calculateGCContentByInterval](#) function.

output.file   Optionally, write file with GC corrected coverage. Can be read with the [readCoverageFile](#) function.

method   Two options for normalization are available: The default "LOESS" largely follows the GC correction of the TitanCNA package. The "POLYNOMIAL" method models the coverage data as a polynomial of degree three and normalizes using the EM approach. The "POLYNOMIAL" is expected to be more robust for very small targeted panels. "POLYNOMIAL" does not support off-target reads.

plot.gc.bias   Optionally, plot GC profiles of the pre-normalized and post-normalized cover-
               age. Provides a quick visual check of coverage bias.

plot.max.density

               By default, if the number of intervals in the probe-set is > 50000, uses a kernel
               density estimate to plot the coverage distribution. This uses the stat_density
               function from the ggplot2 package. Using this parameter, change the threshold
               at which density estimation is applied. If the plot.gc.bias parameter is set as
               FALSE, this will be ignored.

Author(s)

Angad Singh, Markus Riester

See Also

[calculateGCContentByInterval](#)

Examples

```
normal.coverage.file <- system.file("extdata", "example_normal.txt",
    package="PureCN")
gc.gene.file <- system.file("extdata", "example_gc.gene.file.txt",
    package="PureCN")
# normalize using default LOESS method
coverage <- correctCoverageBias(normal.coverage.file, gc.gene.file)
# normalize with POLYNOMIAL method for small panels
coverage <- correctCoverageBias(normal.coverage.file, gc.gene.file,
    method="POLYNOMIAL", plot.gc.bias=TRUE)
```

---

createCurationFile   *Create file to curate PureCN results*

Description

Function to create a CSV file that can be used to mark the correct solution in the output of a
[runAbsoluteCN](#) run.

Usage

```
createCurationFile(file.rds, overwrite.uncurated = TRUE,
  overwrite.curated = FALSE)
```

Arguments

file.rds   Output of the [runAbsoluteCN](#) function, serialized with saveRDS.

overwrite.uncurated

               Overwrite existing files unless flagged as 'Curated'.

overwrite.curated

               Overwrite existing files even if flagged as 'Curated'.

Value

A data.frame with the tumor purity and ploidy of the maximum likelihood solution.

Author(s)

Markus Riester

See Also

runAbsoluteCN

Examples

```
data(purecn.example.output)
file.rds <- "Sample1_PureCN.rds"
saveRDS(purecn.example.output, file=file.rds)
createCurationFile(file.rds)
```

---

createNormalDatabase   *Create database of normal samples*

Description

Function to create a database of normal samples, used to find a good match for tumor copy number normalization. Internally, this function determines the sex of the samples and trains a PCA that is later used for clustering a tumor file with all normal samples in the database.

Usage

```
createNormalDatabase(normal.coverage.files, sex = NULL,
  max.mean.coverage = NULL, ...)
```

Arguments

normal.coverage.files
> Vector with file names pointing to coverage files of normal samples.

sex
> character(length(normal.coverage.files)) with sex for all files. F for female, M for male. If all chromosomes are diploid, specify diploid. If NULL, determine from coverage.

max.mean.coverage
> Assume that coverages above this value do not necessarily improve copy number normalization. Internally, samples with coverage higher than this value will be normalized to have mean coverage equal to this value. If NULL, use the 80 percentile as cutoff. If NA, does not use a maximum value.

...
> Arguments passed to the prcomp function.

Value

A normal database that can be used in the findBestNormal function to retrieve good matching normal samples for a given tumor sample.

Author(s)

Markus Riester

See Also

[findBestNormal](#)

Examples

```
normal.coverage.file <- system.file("extdata", "example_normal.txt",
    package="PureCN")
normal2.coverage.file <- system.file("extdata", "example_normal2.txt",
    package="PureCN")
normal.coverage.files <- c(normal.coverage.file, normal2.coverage.file)
normalDB <- createNormalDatabase(normal.coverage.files)
```

---

createSNPBlacklist   *Create SNP black list*

Description

This function is defunct If a pool of normal VCF is available, please use the normal.panel.vcf.file argument of the [setMappingBiasVcf](#) function.

Usage

```
createSNPBlacklist()
```

Author(s)

Markus Riester

---

createTargetWeights   *Calculate target weights*

Description

Creates a target weight file useful for segmentation. Requires a set of coverage files from normal samples. A small number of tumor (or other normal) samples is then tested against all normals. Target weights will be set proportional to the inverse of coverage standard deviation across all normals. Targets with high variance in coverage in the pool of normals are thus down-weighted.

Usage

```
createTargetWeights(tumor.coverage.files, normal.coverage.files,
  target.weight.file, plot = FALSE)
```

Arguments

tumor.coverage.files

> A small number (1-3) of tumor or normal coverage samples.

normal.coverage.files

> A large number of normal coverage samples (>20) to estimate target log-ratio standard deviations. Should not overlap with files in tumor.coverage.files.

target.weight.file

> Output filename.

plot   Diagnostics plot, useful to tune parameters.

Value

A data.frame with target weights.

Author(s)

Markus Riester

Examples

```
target.weight.file <- "target_weights.txt"
normal.coverage.file <- system.file("extdata", "example_normal.txt",
    package="PureCN")
normal2.coverage.file <- system.file("extdata", "example_normal2.txt",
    package="PureCN")
normal.coverage.files <- c(normal.coverage.file, normal2.coverage.file)
tumor.coverage.file <- system.file("extdata", "example_tumor.txt",
    package="PureCN")

createTargetWeights(tumor.coverage.file, normal.coverage.files, target.weight.file)
```

---

filterTargets   *Remove low quality targets*

Description

This function determines which intervals in the coverage files should be included or excluded in the segmentation. It is called via the fun.filterTargets argument of runAbsoluteCN. The arguments are passed via args.filterTargets.

Usage

```
filterTargets(normal, tumor, log.ratio, seg.file, filter.lowhigh.gc = 0.001,
  min.coverage = 15, min.targeted.base = 5, normalDB = NULL,
  normalDB.min.coverage = 0.25, normalDB.max.missing = 0.03)
```

Arguments

normal	Coverage data for normal sample.
tumor	Coverage data for tumor sample.
log.ratio	Copy number log-ratios, one for each target or interval in coverage file.
seg.file	If not NULL, then do not filter targets, because data is already segmented via the provided segmentation file.
filter.lowhigh.gc	
	Quantile q (defines lower q and upper 1-q) for removing targets with outlier GC profile. Assuming that GC correction might not have been worked on those. Requires `gc.gene.file`.
min.coverage	Minimum coverage in both normal and tumor. Targets with lower coverage are ignored. If a `normalDB` is provided, then this database already provides information about low quality targets and the `min.coverage` is set to `min.coverage/10000`.
min.targeted.base	
	Exclude intervals with targeted base (size in bp) smaller than this cutoff. This is useful when the same interval file was used to calculate GC content. For such small targets, the GC content is likely very different from the true GC content of the probes.
normalDB	Normal database, created with [createNormalDatabase]().
normalDB.min.coverage	
	Exclude targets with coverage lower than 20 percent of the chromosome median in the pool of normals.
normalDB.max.missing	
	Exclude targets with zero coverage in the specified fraction of normal samples.

Value

`logical(length(log.ratio))` specifying which targets should be used in segmentation.

Author(s)

Markus Riester

Examples

```
normal.coverage.file <- system.file("extdata", "example_normal.txt",
    package="PureCN")
normal2.coverage.file <- system.file("extdata", "example_normal2.txt",
    package="PureCN")
normal.coverage.files <- c(normal.coverage.file, normal2.coverage.file)
normalDB <- createNormalDatabase(normal.coverage.files)

tumor.coverage.file <- system.file("extdata", "example_tumor.txt",
    package="PureCN")
vcf.file <- system.file("extdata", "example_vcf.vcf",
    package="PureCN")
gc.gene.file <- system.file("extdata", "example_gc.gene.file.txt",
    package="PureCN")

# The max.candidate.solutions, max.ploidy and test.purity parameters are set to
# non-default values to speed-up this example.  This is not a good idea for real
```

```
# samples.
ret <-runAbsoluteCN(normal.coverage.file=normal.coverage.file,
    tumor.coverage.file=tumor.coverage.file, genome="hg19", vcf.file=vcf.file,
    sampleid="Sample1", gc.gene.file=gc.gene.file, normalDB=normalDB,
    args.filterTargets=list(min.targeted.base=10), max.ploidy=4,
    test.purity=seq(0.3,0.7,by=0.05), max.candidate.solutions=1)
```

---

filterVcfBasic   *Basic VCF filter function*

Description

Function to remove artifacts and low confidence/quality variant calls.

Usage

```
filterVcfBasic(vcf, tumor.id.in.vcf = NULL, use.somatic.status = TRUE,
  snp.blacklist = NULL, af.range = c(0.03, 0.97),
  contamination.range = c(0.01, 0.075), min.coverage = 15,
  min.base.quality = 25, min.supporting.reads = NULL, error = 0.001,
  target.granges = NULL, remove.off.target.snvs = TRUE,
  model.homozygous = FALSE, interval.padding = 50)
```

Arguments

vcf	CollapsedVCF object, read in with the `readVcf` function from the VariantAnnotation package.
tumor.id.in.vcf	
	The tumor id in the `CollapsedVCF` (optional).
use.somatic.status	
	If somatic status and germline data is available, then use this information to remove non-heterozygous germline SNPs or germline SNPS with biased allelic fractions.
snp.blacklist	A file with blacklisted genomic regions. Must be parsable by `import` from rtracklayer, for a example a BED file with file extension '.bed'.
af.range	Exclude variants with allelic fraction smaller or greater than the two values, respectively. The higher value removes homozygous SNPs, which potentially have allelic fractions smaller than 1 due to artifacts or contamination. If a matched normal is available, this value is ignored, because homozygosity can be confirmed in the normal.
contamination.range	
	Count variants in dbSNP with allelic fraction in the specified range. If the number of these putative contamination variants exceeds an expected value and if they are found on almost all chromosomes, the sample is flagged as potentially contaminated and extra contamination estimation steps will be performed later on.
min.coverage	Minimum coverage in tumor. Variants with lower coverage are ignored.
min.base.quality	
	Minimim base quality in tumor. Requires a `BQ` genotype field in the VCF.

min.supporting.reads

> Minimum number of reads supporting the alt allele. If NULL, calculate based on coverage and assuming sequencing error of 10^-3.

error Estimated sequencing error rate. Used to calculate minimum number of supporting reads using [calculatePowerDetectSomatic](#).

target.granges GenomicRanges object specifiying the target postions. Used to remove off-target reads. If NULL, do not check whether variants are on or off-target.

remove.off.target.snvs

> If set to a true value, will remove all SNVs outside the covered regions.

model.homozygous

> If set to TRUE, does not remove homozygous variants. Ignored in case a matched normal is provided in the VCF.

interval.padding

> Include variants in the interval flanking regions of the specified size in bp. Requires target.granges.

Value

A list with elements

vcf The filtered CollapsedVCF object.

flag A flag (logical(1)) if problems were identified.

flag_comment A comment describing the flagging.

Author(s)

Markus Riester

See Also

[calculatePowerDetectSomatic](#)

Examples

```
# This function is typically only called by runAbsolute via
# fun.filterVcf and args.filterVcf.
vcf.file <- system.file("extdata", "example_vcf.vcf", package="PureCN")
vcf <- readVcf(vcf.file, "hg19")
vcf.filtered <- filterVcfBasic(vcf)
```

---

filterVcfMuTect *Filter VCF MuTect*

Description

Function to remove artifacts and low confidence/quality calls from a MuTect generated VCF file. Also applies filters defined in filterVcfBasic. This function will only keep variants listed in the stats file and those not matching the specified failure reasons.

Usage

```
filterVcfMuTect(vcf, tumor.id.in.vcf = NULL, stats.file = NULL,
  ignore = c("clustered_read_position", "fstar_tumor_lod",
  "nearby_gap_events", "poor_mapping_region_alternate_allele_mapq",
  "poor_mapping_region_mapq0", "possible_contamination", "strand_artifact",
  "seen_in_panel_of_normals"), ...)
```

Arguments

vcf	CollapsedVCF object, read in with the readVcf function from the VariantAnnotation package.
tumor.id.in.vcf	
	The tumor id in the VCF file, optional.
stats.file	MuTect stats file. If NULL, will check if VCF was generated by MuTect2 and if yes will call filterVcfMuTect2 instead.
ignore	MuTect flags that mark variants for exclusion.
...	Additional arguments passed to filterVcfBasic.

Value

A list with elements vcf, flag and flag_comment. vcf contains the filtered CollapsedVCF, flag a logical(1) flag if problems were identified, further described in flag_comment.

Author(s)

Markus Riester

See Also

filterVcfBasic

Examples

```
### This function is typically only called by runAbsolute via the
### fun.filterVcf and args.filterVcf comments.
library(VariantAnnotation)
vcf.file <- system.file("extdata", "example_vcf.vcf", package="PureCN")
vcf <- readVcf(vcf.file, "hg19")
vcf.filtered <- filterVcfMuTect(vcf)
```

---

filterVcfMuTect2 *Filter VCF MuTect2*

Description

Function to remove artifacts and low confidence/quality calls from a GATK4/MuTect2 generated VCF file. Also applies filters defined in filterVcfBasic.

Usage

```
filterVcfMuTect2(vcf, tumor.id.in.vcf = NULL, ignore = c("clustered_events",
  "t_lod", "str_contraction", "read_position", "fragment_length",
  "multiallelic", "clipping", "strand_artifact"), ...)
```

Arguments

vcf	CollapsedVCF object, read in with the readVcf function from the VariantAnnotation package.
tumor.id.in.vcf	
	The tumor id in the VCF file, optional.
ignore	MuTect2 flags that mark variants for exclusion.
...	Additional arguments passed to filterVcfBasic.

Value

A list with elements vcf, flag and flag_comment. vcf contains the filtered CollapsedVCF, flag a logical(1) flag if problems were identified, further described in flag_comment.

Author(s)

Markus Riester

See Also

filterVcfBasic

Examples

```
### This function is typically only called by runAbsolute via the
### fun.filterVcf and args.filterVcf comments.
library(VariantAnnotation)
vcf.file <- system.file("extdata", "example_vcf.vcf", package="PureCN")
vcf <- readVcf(vcf.file, "hg19")
vcf.filtered <- filterVcfMuTect(vcf)
```

---

findBestNormal   *Find best normal sample in database*

Description

Function to find the best matching normal for a provided tumor sample.

Usage

```
findBestNormal(tumor.coverage.file, normalDB, pcs = 1:3, num.normals = 1,
  ignore.sex = FALSE, sex = NULL, normal.coverage.files = NULL,
  pool = FALSE, pool.weights = c("voom", "equal"), plot.pool = FALSE, ...)
```

Arguments

tumor.coverage.file

        Coverage file or data of a tumor sample.

normalDB	Database of normal samples, created with [createNormalDatabase](#).
pcs	Principal components to use for distance calculation.
num.normals	Return the num.normals best normals.
ignore.sex	If FALSE, detects sex of sample and returns best normals with matching sex.
sex	Sex of sample. If NULL, determine with [getSexFromCoverage](#) and default parameters. Valid values are F for female, M for male. If all chromosomes are diploid, specify diploid.

normal.coverage.files

        Only consider these normal samples. If NULL, use all in the database. Must match normalDB$normal.coverage.files.

pool	If TRUE, use [poolCoverage](#) to pool best normals.
pool.weights	Either find good pooling weights by optimization or weight all best normals equally.
plot.pool	Allows the pooling function to create plots.
...	Additional arguments passed to [poolCoverage](#).

Value

Filename of the best matching normal.

Author(s)

Markus Riester

See Also

[createNormalDatabase](#) [getSexFromCoverage](#)

Examples

```
normal.coverage.file <- system.file("extdata", "example_normal.txt",
    package="PureCN")
normal2.coverage.file <- system.file("extdata", "example_normal2.txt",
    package="PureCN")
normal.coverage.files <- c(normal.coverage.file, normal2.coverage.file)
normalDB <- createNormalDatabase(normal.coverage.files)

tumor.coverage.file <- system.file("extdata", "example_tumor.txt",
    package="PureCN")
best.normal.coverage.file <- findBestNormal(tumor.coverage.file, normalDB)

pool <- findBestNormal(tumor.coverage.file, normalDB, num.normals=2,
    pool=TRUE)
```

---

findFocal   *Find focal amplifications*

Description

Function to find focal amplifications in segmented data. This is automatically called in [runAbsoluteCN](#).

Usage

```
findFocal(seg, max.size = 3e+06, cn.diff = 2, min.amp.cn = 5)
```

Arguments

seg	Segmentation data.
max.size	Cutoff for focal in base pairs.
cn.diff	Minimum copy number delta between neighboring segments.
min.amp.cn	Minimum amplification integer copy number. Segments with lower copy number are not tested.

Value

logical(n), indicating for all n segments whether they are focally amplified or not.

Author(s)

Markus Riester

See Also

[runAbsoluteCN](#)

Examples

```
normal.coverage.file <- system.file("extdata", "example_normal.txt",
    package="PureCN")
tumor.coverage.file <- system.file("extdata", "example_tumor.txt",
    package="PureCN")
vcf.file <- system.file("extdata", "example_vcf.vcf",
    package="PureCN")
gc.gene.file <- system.file("extdata", "example_gc.gene.file.txt",
    package="PureCN")

# The max.candidate.solutions, max.ploidy and test.purity parameters are set to
# non-default values to speed-up this example.  This is not a good idea for real
# samples.
ret <-runAbsoluteCN(normal.coverage.file=normal.coverage.file,
    tumor.coverage.file=tumor.coverage.file, vcf.file=vcf.file, genome="hg19",
    sampleid="Sample1", gc.gene.file=gc.gene.file,
    max.candidate.solutions=1, max.ploidy=4, test.purity=seq(0.3,0.7,by=0.05),
    args.focal=list(max.size = 2e+06), fun.focal=findFocal)
```

---

getDiploid   *Function to extract diploid solutions.*

Description

This function is defunct.

Usage

```
getDiploid()
```

Author(s)

Markus Riester

---

getSexFromCoverage   *Get sample sex from coverage*

Description

This function determines the sex of a sample by the coverage ratio of chrX and chrY. Loss of
chromosome Y (LOY) can result in a wrong female call. For small targeted panels, this will only
work when sufficient sex marker genes such as AMELY are covered. For optimal results, parameters
might need to be tuned for the assay.

Usage

```
getSexFromCoverage(coverage.file, min.ratio = 25, min.ratio.na = 20,
  remove.outliers = TRUE)
```

Arguments

coverage.file   Coverage file or data read with [readCoverageFile](#).

min.ratio   Min chrX/chrY coverage ratio to call sample as female.

min.ratio.na   Min chrX/chrY coverage ratio to call sample as NA. This ratio defines a grey zone
                    from min.ratio.na to min.ratio in which samples are not called. The default
                    is set to a copy number ratio that would be rare in male samples, but lower
                    than expected in female samples. Contamination can be a source of ambiguous
                    calls. Mappability issues on chromosome Y resulting in low coverage need to
                    be considered when setting cutoffs.

remove.outliers
                    Removes coverage outliers before calculating mean chromosome coverages.

Value

Returns a character(1) with M for male, F for female, or NA if unknown.

Author(s)

Markus Riester

See Also

[getSexFromVcf](#)

Examples

```
tumor.coverage.file <- system.file("extdata", "example_tumor.txt",
    package="PureCN")
sex <- getSexFromCoverage(tumor.coverage.file)
```

---

getSexFromVcf   *Get sample sex from a VCF file*

Description

This function detects non-random distribution of homozygous variants on chromosome X compared to all other chromosomes. A non-significant Fisher's exact p-value indicates more than one chromosome X copy. This function is called in runAbsoluteCN as sanity check when a VCF is provided. It is also useful for determining sex when no sex marker genes on chrY (e.g. AMELY) are available.

Usage

```
getSexFromVcf(vcf, tumor.id.in.vcf = NULL, min.or = 4, min.or.na = 2.5,
  max.pv = 0.001, homozygous.cutoff = 0.95, af.cutoff = 0.2,
  min.coverage = 15, use.somatic.status = TRUE)
```

Arguments

vcf	CollapsedVCF object, read in with the readVcf function from the VariantAnnotation package.
tumor.id.in.vcf	
	The tumor id in the CollapsedVCF (optional).
min.or	Minimum odds-ratio to call sample as male. If p-value is not significant due to a small number of SNPs on chromosome X, sample will be called as NA even when odds-ratio exceeds this cutoff.
min.or.na	Minimum odds-ratio to not call a sample. Odds-ratios in the range min.or.na to min.or define a grey area in which samples are not called. Contamination can be a source of ambiguous calls.
max.pv	Maximum Fisher's exact p-value to call sample as male.
homozygous.cutoff	
	Minimum allelic fraction to call position homozygous.
af.cutoff	Remove all SNVs with allelic fraction lower than the specified value.
min.coverage	Minimum coverage in tumor. Variants with lower coverage are ignored.
use.somatic.status	
	If somatic status and germline data is available, then exclude somatic variants.

Value

Returns a character(1) with M for male, F for female, or NA if unknown.

Author(s)

Markus Riester

See Also

getSexFromCoverage

Examples

```
vcf.file <- system.file("extdata", "example_vcf.vcf", package="PureCN")
vcf <- readVcf(vcf.file, "hg19")
# This example vcf is already filtered and contains no homozygous calls,
# which are necessary for determining sex from chromosome X.
getSexFromVcf(vcf)
```

---

plotAbs   *Plots for analyzing PureCN solutions*

Description

This function provides various plots for finding correct purity and ploidy combinations in the results
of a runAbsoluteCN call.

Usage

```
plotAbs(res, ids = NULL, type = c("hist", "overview", "overview2", "BAF",
  "AF", "volcano", "all", "contamination"), chr = NULL,
  germline.only = TRUE, show.contour = FALSE, purity = NULL,
  ploidy = NULL, alpha = TRUE, show.segment.means = c("SNV", "segments",
  "both"), max.mapping.bias = 0.8, palette.name = "Paired", ...)
```

Arguments

res	Return object of the runAbsoluteCN function.
ids	Candidate solutions to be plotted. ids=1 will draw the plot for the maximum likelihood solution.
type	Different types of plots. hist will plot a histogram, assigning log-ratio peaks to integer values. overview will plot all local optima, sorted by likelihood. overview2 adds additional barplots showing likelihood and goodness-of-fit scores. BAF plots something like a B-allele frequency plot known from SNP arrays: it plots allele frequencies of germline variants (or most likely germline when status is not available) against copy number. AF plots observed allelic fractions against expected (purity), maximum likelihood (optimal multiplicity) allelic fractions. volcano plots coverage p-values against log-ratios on the gene-level. all plots all, and is useful for generate a PDF for a sample for manual inspection. contamination plots expected contamination rate per chromosome.
chr	If NULL, show all chromosomes, otherwise only the ones specified (type="BAF" only).

germline.only	If TRUE, show only variants most likely being germline in BAF plot. Useful to set to FALSE (in combination with chr) to study potential artifacts.
show.contour	For type="overview", display contour plot.
purity	Display expected integer copy numbers for purity, defaults to purity of the solution (type="hist" only).
ploidy	Display expected integer copy numbers for ploidy, defaults to ploidy of the solution (type="hist" only).
alpha	Add transparency to the plot if VCF contains many variants (>2000, type="AF" and type="BAF" only).
show.segment.means	
	Show segment means in germline allele frequency plot? If both, show SNVs and segment means. If SNV show all SNVs. Only for type="AF".
max.mapping.bias	
	Exclude variants with high mapping bias from plotting. Note that bias is reported on an inverse scale; a variant with mapping bias of 1 has no bias. (type="AF" and type="BAF" only).
palette.name	The default RColorBrewer palette.
...	Additonal parameters passed to the plot function.

Value

Returns NULL.

Author(s)

Markus Riester

See Also

[runAbsoluteCN](#)

Examples

```
data(purecn.example.output)
plotAbs(purecn.example.output, type="overview")
# plot details for the maximum likelihood solution (rank 1)
plotAbs(purecn.example.output, 1, type="hist")
plotAbs(purecn.example.output, 1, type="BAF")
plotAbs(purecn.example.output, 1, type = "BAF", chr="chr2")
```

---

plotBestNormal   *Plot the PCA of tumor and its best normal(s)*

Description

This function can be used to understand how a best normal is chosen by the [findBestNormal](#) function. It can be also used to tune the best normal selection by finding good parameter values for num.normals and pcs.

Usage

```
plotBestNormal(normal.coverage.files, tumor.coverage.file, normalDB, x = 1,
  y = 2, col.tumor = "red", col.best.normal = "blue",
  col.other.normals = "black", ...)
```

Arguments

normal.coverage.files
                    Coverage file names of normal samples, typically identified via findBestNormal.

tumor.coverage.file
                    Coverage file or data of a tumor sample.

normalDB   Database of normal samples, created with createNormalDatabase.

x   Principal component (PC) to be plotted on x-axis.

y   PC to be plotted on y-axis.

col.tumor   Color of tumor in plot.

col.best.normal
                    Color of best normals in plot.

col.other.normals
                    Color of other normals in plot.

...   Arguments passed to the plot function.

Value

Returns NULL.

Author(s)

Markus Riester

See Also

createNormalDatabase findBestNormal

Examples

```
normal.coverage.file <- system.file("extdata", "example_normal.txt",
    package="PureCN")
normal2.coverage.file <- system.file("extdata", "example_normal2.txt",
     package="PureCN")
normal.coverage.files <- c(normal.coverage.file, normal2.coverage.file)
normalDB <- createNormalDatabase(normal.coverage.files)

tumor.coverage.file <- system.file("extdata", "example_tumor.txt",
    package="PureCN")
best.normal.coverage.file <- findBestNormal(tumor.coverage.file, normalDB)
plotBestNormal(best.normal.coverage.file, tumor.coverage.file, normalDB)

# Display sample sex. The first point in the plot is always tumor.
plotBestNormal(best.normal.coverage.file, tumor.coverage.file, normalDB,
    pch=c(1,ifelse(normalDB$sex=="F", 1, 2)))
```

poolCoverage	*Pool coverage from multiple samples*

Description

Averages the coverage of a list of samples.

Usage

```
poolCoverage(all.data, remove.chrs = c(), w = NULL)
```

Arguments

`all.data`	List of normals, read with [readCoverageFile](#).
`remove.chrs`	Remove these chromosomes from the pool.
`w`	`numeric(length(all.data))` vector of weights. If NULL, weight all samples equally.

Value

A `data.frame` with the averaged coverage over all normals.

Author(s)

Markus Riester

See Also

[readCoverageFile](#)

Examples

```
normal.coverage.file <- system.file("extdata", "example_normal.txt",
    package="PureCN")
normal2.coverage.file <- system.file("extdata", "example_normal2.txt",
    package="PureCN")
normal.coverage.files <- c(normal.coverage.file, normal2.coverage.file)
tumor.coverage.file <- system.file("extdata", "example_tumor.txt",
    package="PureCN")

normalDB <- createNormalDatabase(normal.coverage.files)

# get the best 2 normals and average them
best.normal.coverage.files <- findBestNormal(tumor.coverage.file, normalDB,
    num.normals=2)
pool <- poolCoverage(lapply(best.normal.coverage.files, readCoverageFile),
     remove.chrs=c("chrX", "chrY"))
```

predictSomatic   *Predict germline vs. somatic status*

Description

This function takes as input the output of a [runAbsoluteCN](#) run and provides SNV posterior probabilities for all possible states.

Usage

```
predictSomatic(res, id = 1, return.vcf = FALSE, vcf.field.prefix = "")
```

Arguments

res	Return object of the [runAbsoluteCN](#) function.
id	Candidate solutions to be analyzed. id=1 will analyze the maximum likelihood solution.
return.vcf	Returns an annotated CollapsedVCF object. Note that this VCF will only contain variants not filtered out by the filterVcf functions. Variants outside segments or intervals might be included or not depending on [runAbsoluteCN](#) arguments.
vcf.field.prefix	
	Prefix all VCF info field names with this string.

Value

A data.frame or CollapsedVCF with SNV state posterior probabilities.

Author(s)

Markus Riester

See Also

[runAbsoluteCN](#)

Examples

```
data(purecn.example.output)
# the output data was created using a matched normal sample, but in case
# no matched normal is available, this will help predicting somatic vs.
# germline status
purecnSnvs <- predictSomatic(purecn.example.output)

# Prefer GRanges?
purecnSnvs <- GRanges(predictSomatic(purecn.example.output))

# write a VCF file
purecnVcf <- predictSomatic(purecn.example.output, return.vcf=TRUE)
writeVcf(purecnVcf, file="Sample1_PureCN.vcf")
```

PureCN-defunct *Defunct functions in package 'PureCN'*

Description

These functions are defunct and no longer available.

Details

The following functions are defunct; use the replacement indicated below:

- autoCurateResults: no replacement
- createExonWeightFile: `createTargetWeights`
- createSNPBlacklist: `setMappingBiasVcf`
- getDiploid: no replacement
- readCoverageGatk: `readCoverageFile`

PureCN-deprecated *Deprecated functions in package 'PureCN'*

Description

These functions are provided for compatibility with older versions of 'PureCN' only, and will be defunct at the next release.

Details

The following functions are deprecated and will be made defunct; use the replacement indicated below:

- none

---

purecn.example.output *Example output*

Description

This provides the output of the `runAbsoluteCN` call used in the vignette and examples.

Usage

```
data(purecn.example.output)
```

Value

Output of the `runAbsoluteCN` call used in the vignette.

---

readCoverageFile *Read coverage file*

Description

Read coverage file produced by external tools like The Genome Analysis Toolkit or by calculateBamCoverageByInterv

Usage

```
readCoverageFile(file, format, zero = NULL)
```

Arguments

file	Target coverage file.
format	File format. If missing, derived from the file extension. Currently only GATK DepthofCoverage and CNVkit formats supported.
zero	Start position is 0-based. Default is FALSE for GATK, TRUE for BED file based intervals.

Value

A data.frame with the parsed coverage information.

Author(s)

Markus Riester

See Also

calculateBamCoverageByInterval

Examples

```
tumor.coverage.file <- system.file("extdata", "example_tumor.txt",
    package="PureCN")
coverage <- readCoverageFile(tumor.coverage.file)
```

---

readCoverageGatk *Read GATK coverage files*

Description

This function is deprecated. Please use readCoverageFile instead.

Usage

```
readCoverageGatk()
```

Author(s)

Markus Riester

See Also

[calculateBamCoverageByInterval](#) [readCoverageFile](#)

---

readCurationFile *Read curation file*

Description

Function that can be used to read the curated output of the [runAbsoluteCN](#) function.

Usage

```
readCurationFile(file.rds, file.curation = gsub(".rds$", ".csv", file.rds),
  remove.failed = FALSE, report.best.only = FALSE, min.ploidy = NULL,
  max.ploidy = NULL)
```

Arguments

file.rds	Output of the [runAbsoluteCN](#) function, serialized with saveRDS.
file.curation	Filename of a curation file that points to the correct tumor purity and ploidy solution.
remove.failed	Do not return solutions that failed.
report.best.only	
	Only return correct/best solution (useful on low memory machines when lots of samples are loaded).
min.ploidy	Minimum ploidy to be considered. If NULL, all. Can be used to automatically ignore unlikely solutions.
max.ploidy	Maximum ploidy to be considered. If NULL, all. Can be used to automatically ignore unlikely solutions.

Value

The return value of the corresponding [runAbsoluteCN](#) call, but with the results array manipulated according the curation CSV file and arguments of this function.

Author(s)

Markus Riester

See Also

[runAbsoluteCN](#) [createCurationFile](#)

Examples

```
data(purecn.example.output)
file.rds <- "Sample1_PureCN.rds"
createCurationFile(file.rds)
# User can change the maximum likelihood solution manually in the generated
# CSV file. The correct solution is then loaded with readCurationFile.
purecn.curated.example.output <-readCurationFile(file.rds)
```

---

runAbsoluteCN   *Run PureCN implementation of ABSOLUTE*

Description

This function takes as input tumor and normal control coverage data and a VCF containing allelic
fractions of germline variants and somatic mutations. Normal control does not need to be from the
same patient. In case VCF does not contain somatic status, it should contain dbSNP and optionally
COSMIC annotation. Returns purity and ploidy combinations, sorted by likelihood score. Provides
copy number and LOH data, by both gene and genomic region.

Usage

```
runAbsoluteCN(normal.coverage.file = NULL, tumor.coverage.file = NULL,
  log.ratio = NULL, seg.file = NULL, seg.file.sdev = 0.4,
  vcf.file = NULL, normalDB = NULL, genome, centromeres = NULL,
  sex = c("?", "F", "M", "diploid"), fun.filterVcf = filterVcfMuTect,
  args.filterVcf = list(), fun.setPriorVcf = setPriorVcf,
  args.setPriorVcf = list(), fun.setMappingBiasVcf = setMappingBiasVcf,
  args.setMappingBiasVcf = list(), fun.filterTargets = filterTargets,
  args.filterTargets = list(), fun.segmentation = segmentationCBS,
  args.segmentation = list(), fun.focal = findFocal, args.focal = list(),
  sampleid = NULL, min.ploidy = 1, max.ploidy = 6, test.num.copy = 0:7,
  test.purity = seq(0.15, 0.95, by = 0.01), prior.purity = NULL,
  prior.K = 0.999, prior.contamination = 0.01,
  max.candidate.solutions = 20, candidates = NULL, min.coverage = 15,
  max.coverage.vcf = 300, max.non.clonal = 0.2,
  max.homozygous.loss = c(0.05, 1e+07), non.clonal.M = 1/3,
  max.mapping.bias = 0.8, max.pon = 3, iterations = 30,
  min.variants.segment = 5, log.ratio.calibration = 0.1,
  smooth.log.ratio = TRUE, remove.off.target.snvs = NULL,
  model.homozygous = FALSE, error = 0.001, gc.gene.file = NULL,
  max.dropout = c(0.95, 1.1), min.logr.sdev = 0.15, max.logr.sdev = 0.75,
  max.segments = 300, min.gof = 0.8, plot.cnv = TRUE,
  cosmic.vcf.file = NULL, model = c("beta", "betabin"),
  post.optimize = FALSE, log.file = NULL, verbose = TRUE)
```

Arguments

normal.coverage.file

> Coverage file of normal control (optional if log.ratio is provided - then it will
> be only used to filter low coverage exons). Should be already GC-normalized

with correctCoverageBias. Needs to be either a file name or data read with the readCoverageFile function.

tumor.coverage.file

Coverage file of tumor. If NULL, requires seg.file and an interval file via gc.gene.file. Should be already GC-normalized with correctCoverageBias. Needs to be either a file name or data read with the readCoverageFile function.

log.ratio
Copy number log-ratios for all exons in the coverage files. If NULL, calculated based on coverage files.

seg.file
Segmented data. Optional, to support matched SNP6 data. If NULL, use coverage files or log.ratio to segment the data.

seg.file.sdev
If seg.file provided, the log-ratio standard deviation, used to model likelihood of sub-clonal copy number events.

vcf.file
VCF file, tested with 'MuTect 1' output files. Optional, but typically needed to select between local optima of similar likelihood. Can also be a CollapsedVCF, read with the readVcf function. Requires a DB info flag for dbSNP membership. The default fun.setPriorVcf function will also look for a Cosmic.CNT slot (see cosmic.vcf.file), containing the hits in the COSMIC database. Again, do not expect very useful results without a VCF file.

normalDB
Normal database, created with createNormalDatabase. If provided, used to calculate gene-level p-values (requires Gene column in gc.gene.file) and to filter targets with low coverage in the pool of normal samples.

genome
Genome version, for example hg19.

centromeres
A GRanges object with centromere positions. If NULL, use pre-stored positions for genome versions hg18, hg19 and hg38.

sex
Sex of sample. If ?, detect using getSexFromCoverage function and default parameters. Default parameters might not work well with every assay and might need to be tuned. If set to diploid, then PureCN will assume all chromosomes are diploid and will not try to detect sex.

fun.filterVcf
Function for filtering variants. Expected output is a list with elements vcf (CollapsedVCF), flag (logical(1)) and flag_comment (character(1)). The flags will be added to the output data and can be used to warn users, for example when samples look too noisy. Default filter will remove variants flagged by MuTect, but will keep germline variants. If ran in matched normal mode, it will by default use somatic status of variants and filter non-somatic calls with allelic fraction significantly different from 0.5 in normal. Defaults to filterVcfMuTect, which in turn also calls filterVcfBasic.

args.filterVcf
Arguments for variant filtering function. Arguments vcf, tumor.id.in.vcf, min.coverage, model.homozygous and error are required in the filter function and are automatically set.

fun.setPriorVcf

Function to set prior for somatic status for each variant in the VCF. Defaults to setPriorVcf.

args.setPriorVcf

Arguments for somatic prior function.

fun.setMappingBiasVcf

Function to set mapping bias for each variant in the VCF. Defaults to setMappingBiasVcf.

args.setMappingBiasVcf

Arguments for mapping bias function.

fun.filterTargets

> Function for filtering low-quality targets in the coverage files. Needs to return a `logical` vector whether an interval should be used for segmentation. Defaults to `filterTargets`.

args.filterTargets

> Arguments for target filtering function. Arguments `normal`, `tumor`, `log.ratio`, `min.coverage` `seg.file` and `normalDB` are required and automatically set.

fun.segmentation

> Function for segmenting the copy number log-ratios. Expected return value is a `data.frame` representation of the segmentation. Defaults to `segmentationCBS`.

args.segmentation

> Arguments for segmentation function. Arguments `normal`, `tumor`, `log.ratio`, `plot.cnv`, `sampleid`, `vcf`, `tumor.id.in.vcf`, `centromeres` are required in the segmentation function and automatically set.

fun.focal   Function for identifying focal amplifications. Defaults to `findFocal`.

args.focal   Arguments for focal amplification function.

sampleid   Sample id, provided in output files etc.

min.ploidy   Minimum ploidy to be considered.

max.ploidy   Maximum ploidy to be considered.

test.num.copy   Copy numbers tested in the grid search. Note that focal amplifications can have much higher copy numbers, but they will be labeled as subclonal (because they do not fit the integer copy numbers).

test.purity   Considered tumor purity values.

prior.purity   `numeric(length(test.purity))` with priors for tested purity values. If `NULL`, use flat priors.

prior.K

> This defines the prior probability that the multiplicity of a SNV corresponds to either the maternal or the paternal copy number (for somatic variants additionally to a multiplicity of 1). For perfect segmentations, this value would be 1; values smaller than 1 thus may provide some robustness against segmentation errors.

prior.contamination

> The prior probability that a known SNP is from a different individual.

max.candidate.solutions

> Number of local optima considered in optimization and variant fitting steps. If there are too many local optima, it will use specified number of top candidate solutions, but will also include all optima close to diploid, because silent genomes have often lots of local optima.

candidates   Candidates to optimize from a previous run (`return.object$candidates`). If `NULL`, do 2D grid search and find local optima.

min.coverage   Minimum coverage in both normal and tumor. Targets and variants with lower coverage are ignored. This value is provided to the `args.filterTargets` and `args.filterVcf` lists, but can be overwritten in these lists if different cutoffs for the coverage and variant filters are wanted. To increase the sensitivity of homozygous deletions in high purity samples, the coverage cutoff in tumor is automatically lowered by 50 percent if the normal coverage is high.

max.coverage.vcf

> This will set the maximum number of reads in the SNV fitting. This is to avoid that small non-reference biases that come apparent only at high coverages have a dramatic influence on likelihood scores.

max.non.clonal Maximum genomic fraction assigned to a subclonal copy number state.

max.homozygous.loss

> double(2) with maximum genomic fraction assigned to homozygous loss and maximum size of a homozygous loss segment. These are set to a fairly high default value to not exclude correct solutions, especially in noisy segmentations.

non.clonal.M Average expected cellular fraction of sub-clonal somatic mutations. This is to calculate expected allelic fractions of a single sub-clonal bin for variants. For all somatic variants, more accurate cellular fractions are calculated.

max.mapping.bias

> Exclude variants with high mapping bias from the likelihood score calculation. Note that bias is reported on an inverse scale; a variant with mapping bias of 1 has no bias.

max.pon Exclude variants found more than max.pon times in pool of normals and not in dbSNP. Requires normal.panel.vcf.file in [setMappingBiasVcf](). Should be set to a value high enough to be much more likely an artifact and not a true germline variant not present in dbSNP.

iterations Maximum number of iterations in the Simulated Annealing copy number fit optimization. Note that this an integer optimization problem that should converge quickly. Allowed range is 10 to 250.

min.variants.segment

> Flag segments with fewer variants. The minor copy number estimation is not reliable with insufficient variants.

log.ratio.calibration

> Re-calibrate log-ratios in the window sd(log.ratio)*log.ratio.calibration.

smooth.log.ratio

> Smooth log.ratio using the DNAcopy package.

remove.off.target.snvs

> Deprecated. Use the corresponding argument in args.filterVcf.

model.homozygous

> Homozygous germline SNPs are uninformative and by default removed. In 100 percent pure samples such as cell lines, however, heterozygous germline SNPs appear homozygous in case of LOH. Setting this parameter to TRUE will keep homozygous SNPs and include a homozygous SNP state in the likelihood model. Not necessary when matched normal samples are available.

error Estimated sequencing error rate. Used to calculate minimum number of supporting reads for variants using [calculatePowerDetectSomatic](). Also used to calculate the probability of homozygous SNP allelic fractions (assuming reference reads are sequencing errors).

gc.gene.file A mapping file that assigns GC content and gene symbols to each exon in the coverage files. Used for generating gene-level calls. First column in format CHR:START-END. Second column GC content (0 to 1). Third column gene symbol. This file can be generated with the 'GATK GCContentByInterval' tool or with the [calculateGCContentByInterval]() function.

max.dropout Measures GC bias as ratio of coverage in AT-rich (GC < 0.5) versus GC-rich on-target regions (GC >= 0.5). High drop-out might indicate that data was not GC-normalized or that the sample quality might be insufficient. Requires gc.gene.file.

min.logr.sdev Minimum log-ratio standard deviation used in the model. Useful to make fitting more robust to outliers in very clean data.

max.logr.sdev	Flag noisy samples with segment log-ratio standard deviation larger than this. Assay specific and needs to be calibrated.
max.segments	Flag noisy samples with a large number of segments. Assay specific and needs to be calibrated.
min.gof	Flag purity/ploidy solutions with poor fit.
plot.cnv	Generate segmentation plots.
cosmic.vcf.file	
	Add a `Cosmic.CNT` info field to the provided `vcf.file` using a VCF file containing the COSMIC database. The default `fun.setPriorVcf` function will give variants found in the COSMIC database a higher prior probability of being somatic. Not used in likelihood model when matched normal is available in `vcf.file`. Should be compressed and indexed with bgzip and tabix, respectively.
model	Use either a beta or a beta-binomial distribution for fitting observed to expected allelic fractions of alterations in `vcf.file`. The latter can be useful to account for significant overdispersion, for example due to mapping biases when no pool of normals is available or due to other unmodeled biases, e.g. amplification biases. The amount of expected overdispersion can be controlled via the `max.coverage.vcf` argument (the higher, the less expected bias).
post.optimize	Optimize purity using final SCNA-fit and variants. This might take a long time when lots of variants need to be fitted, but will typically result in a slightly more accurate purity, especially for rather silent genomes or very low purities. Otherwise, it will just use the purity determined via the SCNA-fit.
log.file	If not `NULL`, store verbose output to file.
verbose	Verbose output.

Value

A list with elements

candidates	Results of the grid search.
results	All local optima, sorted by final rank.
input	The input data.

Author(s)

Markus Riester

References

Riester et al. (2016). PureCN: Copy number calling and SNV classification using targeted short read sequencing. Source Code for Biology and Medicine, 11, pp. 13.

Carter et al. (2012), Absolute quantification of somatic DNA alterations in human cancer. Nature Biotechnology.

See Also

[correctCoverageBias](#) [segmentationCBS](#) [calculatePowerDetectSomatic](#)

Examples

```
normal.coverage.file <- system.file('extdata', 'example_normal.txt',
    package='PureCN')
tumor.coverage.file <- system.file('extdata', 'example_tumor.txt',
    package='PureCN')
vcf.file <- system.file('extdata', 'example_vcf.vcf',
    package='PureCN')
gc.gene.file <- system.file('extdata', 'example_gc.gene.file.txt',
    package='PureCN')

# The max.candidate.solutions, max.ploidy and test.purity parameters are set to
# non-default values to speed-up this example.  This is not a good idea for real
# samples.
ret <-runAbsoluteCN(normal.coverage.file=normal.coverage.file,
    tumor.coverage.file=tumor.coverage.file, genome='hg19', vcf.file=vcf.file,
    sampleid='Sample1', gc.gene.file=gc.gene.file,
    max.ploidy=4, test.purity=seq(0.3,0.7,by=0.05), max.candidate.solutions=1)


# If a high-quality segmentation was obtained with third-party tools:
seg.file <- system.file('extdata', 'example_seg.txt',
    package = 'PureCN')

# By default, PureCN will re-segment the data, for example to identify
# regions of copy number neutral LOH. If this is not wanted, we can provide
# a minimal segmentation function which just returns the provided one:
funSeg <- function(seg, ...) return(seg)

res <- runAbsoluteCN(seg.file=seg.file, fun.segmentation=funSeg, max.ploidy = 4,
    test.purity = seq(0.3, 0.7, by = 0.05), max.candidate.solutions=1,
    genome='hg19', gc.gene.file=gc.gene.file)
```

---

segmentationCBS   *CBS segmentation*

Description

The default segmentation function. This function is called via the `fun.segmentation` argument of
[runAbsoluteCN](#). The arguments are passed via `args.segmentation`.

Usage

```
segmentationCBS(normal, tumor, log.ratio, seg, plot.cnv, sampleid,
  target.weight.file = NULL, alpha = 0.005, undo.SD = NULL, vcf = NULL,
  tumor.id.in.vcf = 1, normal.id.in.vcf = NULL, max.segments = NULL,
  prune.hclust.h = NULL, prune.hclust.method = "ward.D", chr.hash = NULL,
  centromeres = NULL)
```

Arguments

normal   Coverage data for normal sample.

tumor	Coverage data for tumor sample.
log.ratio	Copy number log-ratios, one for each target in the coverage files.
seg	If segmentation was provided by the user, this data structure will contain this segmentation. Useful for minimal segmentation functions. Otherwise PureCN will re-segment the data. This segmentation function ignores this user provided segmentation.
plot.cnv	Segmentation plots.
sampleid	Sample id, used in output files.
target.weight.file	
	Can be used to assign weights to targets.
alpha	Alpha value for CBS, see documentation for the segment function.
undo.SD	undo.SD for CBS, see documentation of the segment function. If NULL, try to find a sensible default.
vcf	Optional CollapsedVCF object with germline allelic ratios.
tumor.id.in.vcf	
	Id of tumor in case multiple samples are stored in VCF.
normal.id.in.vcf	
	Id of normal in in VCF. Currently not used.
max.segments	If not NULL, try a higher undo.SD parameter if number of segments exceeds the threshold.
prune.hclust.h	Height in the hclust pruning step. Increasing this value will merge segments more aggressively. If NULL, try to find a sensible default.
prune.hclust.method	
	Cluster method used in the hclust pruning step. See documentation for the hclust function.
chr.hash	Mapping of non-numerical chromsome names to numerical names (e.g. chr1 to 1, chr2 to 2, etc.). If NULL, assume chromsomes are properly ordered.
centromeres	A GRanges object with centromere positions. Currently not supported in this function.

Value

data.frame containing the segmentation.

Author(s)

Markus Riester

References

Olshen, A. B., Venkatraman, E. S., Lucito, R., Wigler, M. (2004). Circular binary segmentation for the analysis of array-based DNA copy number data. Biostatistics 5: 557-572.

Venkatraman, E. S., Olshen, A. B. (2007). A faster circular binary segmentation algorithm for the analysis of array CGH data. Bioinformatics 23: 657-63.

See Also

[runAbsoluteCN](#)

Examples

```
normal.coverage.file <- system.file("extdata", "example_normal.txt",
    package="PureCN")
tumor.coverage.file <- system.file("extdata", "example_tumor.txt",
    package="PureCN")
vcf.file <- system.file("extdata", "example_vcf.vcf",
    package="PureCN")
gc.gene.file <- system.file("extdata", "example_gc.gene.file.txt",
    package="PureCN")

# The max.candidate.solutions, max.ploidy and test.purity parameters are set to
# non-default values to speed-up this example.  This is not a good idea for real
# samples.
ret <-runAbsoluteCN(normal.coverage.file=normal.coverage.file,
    tumor.coverage.file=tumor.coverage.file, vcf.file=vcf.file, genome="hg19",
    sampleid="Sample1", gc.gene.file=gc.gene.file,
    max.candidate.solutions=1, max.ploidy=4, test.purity=seq(0.3,0.7,by=0.05),
    fun.segmentation=segmentationCBS, args.segmentation=list(alpha=0.001))
```

---

segmentationPSCBS   *PSCBS segmentation*

Description

Alternative segmentation function using the PSCBS package. This function is called via the fun.segmentation argument of [runAbsoluteCN](). The arguments are passed via args.segmentation.

Usage

```
segmentationPSCBS(normal, tumor, log.ratio, seg, plot.cnv, sampleid,
  target.weight.file = NULL, alpha = 0.005, undo.SD = NULL,
  flavor = "tcn&dh", tauA = 0.03, vcf = NULL, tumor.id.in.vcf = 1,
  normal.id.in.vcf = NULL, max.segments = NULL, prune.hclust.h = NULL,
  prune.hclust.method = "ward.D", chr.hash = NULL, centromeres = NULL,
  ...)
```

Arguments

normal	Coverage data for normal sample.
tumor	Coverage data for tumor sample.
log.ratio	Copy number log-ratios, one for each exon in coverage file.
seg	If segmentation was provided by the user, this data structure will contain this segmentation. Useful for minimal segmentation functions. Otherwise PureCN will re-segment the data. This segmentation function ignores this user provided segmentation.
plot.cnv	Segmentation plots.
sampleid	Sample id, used in output files.

target.weight.file	
	Can be used to assign weights to targets. NOT SUPPORTED YET in segmentation. Will remove targets with weight below 1/3.
alpha	Alpha value for CBS, see documentation for the `segment` function.
undo.SD	`undo.SD` for CBS, see documentation of the `segment` function. If `NULL`, try to find a sensible default.
flavor	Flavor value for PSCBS. See `segmentByNonPairedPSCBS`.
tauA	tauA argument for PSCBS. See `segmentByNonPairedPSCBS`.
vcf	Optional VCF object with germline allelic ratios.
tumor.id.in.vcf	
	Id of tumor in case multiple samples are stored in VCF.
normal.id.in.vcf	
	Id of normal in in VCF. If `NULL`, use unpaired PSCBS.
max.segments	If not `NULL`, try a higher `undo.SD` parameter if number of segments exceeds the threshold.
prune.hclust.h	Height in the `hclust` pruning step. Increasing this value will merge segments more aggressively. If `NULL`, try to find a sensible default.
prune.hclust.method	
	Cluster method used in the `hclust` pruning step. See documentation for the `hclust` function.
chr.hash	Mapping of non-numerical chromsome names to numerical names (e.g. chr1 to 1, chr2 to 2, etc.). If `NULL`, assume chromsomes are properly ordered.
centromeres	A GRanges with centromere positions. If not `NULL`, add breakpoints at centromeres.
...	Additional parameters passed to the `segmentByNonPairedPSCBS` function.

Value

`data.frame` containing the segmentation.

Author(s)

Markus Riester

References

Olshen, A. B., Venkatraman, E. S., Lucito, R., Wigler, M. (2004). Circular binary segmentation for the analysis of array-based DNA copy number data. Biostatistics 5: 557-572.

Venkatraman, E. S., Olshen, A. B. (2007). A faster circular binary segmentation algorithm for the analysis of array CGH data. Bioinformatics 23: 657-63.

Olshen et al. (2011). Parent-specific copy number in paired tumor-normal studies using circular binary segmentation. Bioinformatics.

See Also

[runAbsoluteCN](#)

Examples

```
normal.coverage.file <- system.file("extdata", "example_normal.txt",
    package="PureCN")
tumor.coverage.file <- system.file("extdata", "example_tumor.txt",
    package="PureCN")
vcf.file <- system.file("extdata", "example_vcf.vcf",
    package="PureCN")
gc.gene.file <- system.file("extdata", "example_gc.gene.file.txt",
    package="PureCN")

# The max.candidate.solutions, max.ploidy and test.purity parameters are set to
# non-default values to speed-up this example.  This is not a good idea for real
# samples.
 ret <-runAbsoluteCN(normal.coverage.file=normal.coverage.file,
    tumor.coverage.file=tumor.coverage.file, vcf.file=vcf.file, genome="hg19",
    sampleid="Sample1", gc.gene.file=gc.gene.file,
    fun.segmentation=segmentationPSCBS, max.ploidy=4,
    test.purity=seq(0.3,0.7,by=0.05), max.candidate.solutions=1)
```

---

setMappingBiasVcf *Set Mapping Bias VCF*

Description

Function to set mapping bias for each variant in the provided CollapsedVCF object. By default, it returns the same value for all variants, but a pool of normal samples can be provided for position-specific mapping bias calculation.

Usage

```
setMappingBiasVcf(vcf, tumor.id.in.vcf = NULL, normal.panel.vcf.file = NULL,
  min.normals = 2, smooth = TRUE, smooth.n = 5)
```

Arguments

vcf	CollapsedVCF object, read in with the readVcf function from the VariantAnnotation package.
tumor.id.in.vcf	
	Id of tumor in case multiple samples are stored in VCF.
normal.panel.vcf.file	
	Combined VCF file of a panel of normals, expects allelic fractions as FA genotype field. Should be compressed and indexed with bgzip and tabix, respectively.
min.normals	Minimum number of normals with heterozygous SNP for calculating position-specific mapping bias. Requires normal.panel.vcf.file.
smooth	Impute mapping bias of variants not found in the panel by smoothing of neighboring SNPs. Requires normal.panel.vcf.file.
smooth.n	Number of neighboring variants used for smoothing.

Value

A list with elements

bias   A `numeric(nrow(vcf))` vector with the mapping bias of for each variant in
                the `CollapsedVCF`. Mapping bias is expected as scaling factor. Adjusted allelic
                fraction is (observed allelic fraction)/(mapping bias). Maximum scaling factor
                is 1 and means no bias.

pon.count   A `numeric(nrow(vcf))` vector with the number of hits in the `normal.panel.vcf.file`.

Author(s)

Markus Riester

Examples

```
# This function is typically only called by runAbsoluteCN via
# fun.setMappingBiasVcf and args.setMappingBiasVcf.
vcf.file <- system.file("extdata", "example_vcf.vcf", package="PureCN")
vcf <- readVcf(vcf.file, "hg19")
vcf.bias <- setMappingBiasVcf(vcf)
```

---

setPriorVcf   *Set Somatic Prior VCF*

Description

Function to set prior for somatic mutation status for each variant in the provided `CollapsedVCF`
object.

Usage

```
setPriorVcf(vcf, prior.somatic = c(0.5, 5e-04, 0.999, 1e-04, 0.995, 0.5),
  tumor.id.in.vcf = NULL, min.cosmic.cnt = 4)
```

Arguments

vcf   `CollapsedVCF` object, read in with the `readVcf` function from the VariantAn-
                notation package.

prior.somatic   Prior probabilities for somatic mutations. First value is for the case when no
                matched normals are available and the variant is not in dbSNP (second value).
                Third value is for variants with MuTect somatic call. Different from 1, because
                somatic mutations in segments of copy number 0 have 0 probability and artifacts
                can thus have dramatic influence on likelihood score. Forth value is for variants
                not labeled as somatic by MuTect. Last two values are optional, if vcf contains
                a flag Cosmic.CNT, it will set the prior probability for variants with CNT > 2 to
                the first of those values in case of no matched normal available (0.995 default).
                Final value is for the case that variant is in both dbSNP and COSMIC > 2.

tumor.id.in.vcf
                Id of tumor in case multiple samples are stored in VCF.

min.cosmic.cnt Minimum number of hits in the COSMIC database to call variant as likely so-
                matic.

Value

A `numeric(nrow(vcf))` vector with the prior probability of somatic status for each variant in the `CollapsedVCF`.

Author(s)

Markus Riester

Examples

```
# This function is typically only called by runAbsoluteCN via the
# fun.setPriorVcf and args.setPriorVcf comments.
vcf.file <- system.file("extdata", "example_vcf.vcf", package="PureCN")
vcf <- readVcf(vcf.file, "hg19")
vcf.priorsomatic <- setPriorVcf(vcf)
```

Index