

Package ‘PureCN’

April 15, 2017

Type Package

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

Version 1.2.3

Date 2016-10-27

Author Markus Riester

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

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

Depends R (>= 3.2), DNACopy, VariantAnnotation (>= 1.14.1)

Imports GenomicRanges (>= 1.20.3), IRanges (>= 2.2.1), RColorBrewer, S4Vectors, data.table, grDevices, graphics, stats, utils, SummarizedExperiment, GenomeInfoDb, Rsamtools, Biostrings

Suggests PSCBS, RUnit, BiocStyle, BiocGenerics, knitr, getopt, rtracklayer

VignetteBuilder knitr

License Artistic-2.0

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

NeedsCompilation no

R topics documented:

autoCurateResults	2
bootstrapResults	3
calculateBamCoverageByInterval	4
calculateGCContentByInterval	5
calculateLogRatio	6
calculatePowerDetectSomatic	6
callAlterations	8
callAlterationsFromSegmentation	9
callLOH	10

centromeres	11
correctCoverageBias	11
createCurationFile	12
createExonWeightFile	13
createNormalDatabase	14
createSNPBlacklist	15
createTargetWeights	16
filterTargets	17
filterVcfBasic	18
filterVcfMuTect	19
findBestNormal	20
findFocal	21
getDiploid	22
getSexFromCoverage	23
getSexFromVcf	24
plotAbs	25
plotBestNormal	27
poolCoverage	28
predictSomatic	29
PureCN-deprecated	30
purecn.example.output	30
readCoverageGatk	31
readCurationFile	31
runAbsoluteCN	32
segmentationCBS	37
segmentationPSCBS	39
setMappingBiasVcf	40
setPriorVcf	41

Index 43

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

Description

This implements a workflow with various heuristics, with the goal of identifying correct purity/ploidy solutions in difficult samples. This is mainly for automated copy number calling. This function may evolve over time and might produce different rankings after PureCN updates.

Usage

```
autoCurateResults(res, bootstrap = TRUE, bootstrap.n = 500,
  verbose = TRUE)
```

Arguments

res	Return object of the runAbsoluteCN function.
bootstrap	Try to reduce the number of local optima by using the bootstrapResults function.
bootstrap.n	Number of bootstrap replicates.
verbose	Verbose output.

Value

The provided [runAbsoluteCN](#) return object with unlikely purity and ploidy solutions filtered out.

Author(s)

Markus Riester

See Also

[runAbsoluteCN](#)

Examples

```
data(purecn.example.output)
# no diploid solutions in the example
example.output.curated <- autoCurateResults(purecn.example.output,
  bootstrap.n=100)
```

bootstrapResults	<i>Filter unlikely purity/ploidy solutions</i>
------------------	--

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](#).

Usage

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

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 readCoverageGatk function.
index.file	The bai index. This is expected without the .bai file suffix, see ?scanBam.

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)
```

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, verbose = TRUE)
```

Arguments

normal	Normal coverage read in by the <code>readCoverageGatk</code> function.
tumor	Tumor coverage read in by the <code>readCoverageGatk</code> function.
verbose	Verbose output.

Value

numeric(nrow(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 <- readCoverageGatk(normal.coverage.file)
tumor <- readCoverageGatk(tumor.coverage.file)
log.ratio <- calculateLogRatio(normal, tumor, verbose=FALSE)
```

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., Absolute quantification of somatic DNA alterations in human cancer. Nature Biotechnology 2012.

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

<code>sampleid</code>	The sampleid column in the segmentation file.
<code>chr</code>	The chromosome column.
<code>start</code>	The start positions of the segments.
<code>end</code>	The end positions of the segments.
<code>num.mark</code>	Optionally, the number of probes or markers in each segment.
<code>seg.mean</code>	The segment mean.
<code>C</code>	The segment integer copy number.
<code>gc.gene.file</code>	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.
<code>fun.focal</code>	Function for identifying focal amplifications. Defaults to findFocal .
<code>args.focal</code>	Arguments for focal amplification function.
<code>...</code>	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))
```

centromeres	<i>A list of data.frames containing centromere positions.</i>
-------------	---

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 contains 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	<i>Correct for GC bias</i>
---------------------	----------------------------

Description

Takes as input coverage data in GATK format (or data read by [readCoverageGatk](#)) and a mapping file for GC content, and then uses a loess normalization for bias correction. Largely follows the GC correction of the TitanCNA package.

Usage

```
correctCoverageBias(coverage.file, gc.gene.file, output.file = NULL)
```

Arguments

- `coverage.file` Exon coverage file as produced by GATK. Either a file name or data parsed with the [readCoverageGatk](#) 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 [readCoverageGatk](#) function.

Value

GC normalized coverage.

Author(s)

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")
coverage <- correctCoverageBias(normal.coverage.file, gc.gene.file)
```

`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)
```

Arguments

- `file.rds` Output of the [runAbsoluteCN](#) function, serialized with saveRDS.
- `overwrite.uncurated`
 Overwrite existing files unless 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)
```

createExonWeightFile *Calculate exon weights*

Description

This function has been renamed to [createTargetWeights](#) and will be made defunct in the next Bioconductor release.

Usage

```
createExonWeightFile(tumor.coverage.files, normal.coverage.files,
                     exon.weight.file)
```

Arguments

tumor.coverage.files

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

normal.coverage.files

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

exon.weight.file

Output filename.

Author(s)

Markus Riester

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 = 100, ...)
```

Arguments

`normal.coverage.files` Vector with file names pointing to GATK 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.

`...` 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

Function to create a black list of germline SNPs with expected allelic fraction (AF) smaller than 0.5 in diploid genomes.

Usage

```
createSNPBlacklist(vcf.files, n = min(10, length(vcf.files)),  
  low.af = 0.025, high.af = 0.1, chr.hash = NULL,  
  genome = "hg19")
```

Arguments

vcf.files	List of VCF files. When a VCF file contains multiple samples, it will ignore all samples except the first.
n	Required number of VCF files showing low allelic fraction to blacklist a SNP id.
low.af	Defines a low AF p-value.
high.af	Defines a high AF p-value. For every sample with high AF p-value, there must be one more sample with low AF to reach the cutoff.
chr.hash	Mapping of non-numerical chromosome names to numerical names (e.g. chr1 to 1, chr2 to 2, etc.). If NULL, assume chromosomes are properly ordered.
genome	Version of the reference genome, required for the readVcf function.

Value

A list with elements

snp.blacklist	A data.frame with blacklisted SNPs.
segmented	A data.frame with blacklisted regions.

Author(s)

Markus Riester

Examples

```
# Assume VCF files of normals (for example obtained by a MuTect artifact  
# detection run) are in directory poolofnormals:  
mutect.normal.files <- dir("poolofnormals", pattern="vcf$", full.names=TRUE)  
  
# These files do not exist in our example, so we do not run the function here.  
#snp.blacklist <- createSNPBlacklist(mutect.normal.files)
```

createTargetWeights *Calculate target weights*

Description

Creates a target weight file useful for segmentation. Requires a set of GATK 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, verbose = TRUE)
```

Arguments

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

normal.coverage.files A large number of GATK 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.

verbose Verbose output.

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	<i>Remove low quality targets</i>
---------------	-----------------------------------

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(log.ratio, tumor, gc.data, seg.file,
             filter.lowhigh.gc = 0.001, min.targeted.base = 4,
             normalDB = NULL, normalDB.min.coverage = 0.2, verbose)
```

Arguments

log.ratio	Copy number log-ratios, one for each target or interval in coverage file.
tumor	GATK coverage data for tumor sample.
gc.data	data.frame with GC bias for each interval.
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.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 exons, 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.
verbose	Verbose output.

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,
  args.filterTargets=list(normalDB=normalDB), max.ploidy=4,
  test.purity=seq(0.3,0.7,by=0.05), max.candidate.solutions=1)

```

filterVcfBasic	<i>Basic VCF filter function</i>
----------------	----------------------------------

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.cutoff = c(0.05, 0.075), min.coverage = 15,
  min.supporting.reads = NULL, error = 0.001, verbose = 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).
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	CSV file with SNP ids with expected allelic fraction significantly different from 0.5 in diploid genomes. Can be an array of lists. The function createSNPBlacklist can provide appropriate black lists. Can also be a BED file (either tab or comma separated) of blacklisted genomic regions (columns 1-3: chromosome, start, end).

af.range	Exclude SNPs 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.cutoff	Count SNPs in dbSNP with allelic fraction smaller than the first value, if found on most chromosomes, remove all with AF smaller than the second value.
min.coverage	Minimum coverage in tumor. Variants with lower coverage are ignored.
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 .
verbose	Verbose output.

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 the
# fun.filterVcf and args.filterVcf comments.
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"), verbose = TRUE, ...)
```

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
ignore	
verbose	Verbose output.
...	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, verbose = TRUE)
```

Arguments

tumor.coverage.file	GATK coverage file 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.
verbose	Verbose output.

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)
```

findFocal

Find focal amplifications

Description

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

Usage

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

Arguments

seg	Segmentation data.
size.cutoff	Deprecated, use max.size instead.
max.size	Cutoff for focal in base pairs.
cn.diff	Minimum copy number delta between neighboring segments.
amp.cutoff	Deprecated, use min.amp.cn instead.
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 can be used to extract purity and ploidy solutions that are diploid with only few CNVs. Since high ploidy solutions typically have a very different copy number profile, one of the identified diploid solutions is likely correct if there are any. This function can be used for automated curation workflows; very silent genomes have by definition only a small number of CNVs, making it difficult for the algorithm to correctly identify purity and ploidy. If the maximum likelihood solution is diploid, it is always returned; all other solutions must pass the more stringent criteria as defined in the function arguments.

Usage

```
getDiploid(res, min.diploid = 0.5, min.single.gain.loss = 0.05,  
           max.non.single.gain.loss = 0.1, max.loh = 0.5,  
           min.log.likelihood = NULL)
```

Arguments

res	Return object of the runAbsoluteCN function.
min.diploid	Minimum fraction of genome with normal copy number 2.
min.single.gain.loss	Minimum fraction of genome with copy number 1 or 3. This makes sure that low purity samples are not confused with quiet samples.
max.non.single.gain.loss	Maximum fraction of genome with copy number smaller 1 or more than 3.
max.loh	Maximum fraction of genome in LOH.
min.log.likelihood	Minimum copy number log-likelihood to consider sample. If NULL, not tested.

Value

A list with elements

ids	The ids of diploid solutions (res\$results[ids]).
fraction.non.single	The fraction of the genome with copy number <1 or >3.

Author(s)

Markus Riester

See Also

[runAbsoluteCN](#)

Examples

```
data(purecn.example.output)  
# no diploid solutions in the example  
idx <- getDiploid(purecn.example.output)
```

getSexFromCoverage	<i>Get sample sex from coverage</i>
--------------------	-------------------------------------

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, verbose = TRUE)
```

Arguments

`coverage.file` GATK coverage file or data read with [readCoverageGatk](#).

`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.

`verbose` Verbose output.

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 = 0.03, verbose = 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.
verbose	Verbose output.

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",
  "BAF", "AF", "all"), chr = NULL, germline.only = TRUE,
  show.contour = FALSE, purity = NULL, ploidy = NULL,
  alpha = TRUE, show.segment.means = c("SNV", "segments",
  "both"), ...)
```

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. 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. all plots all, and is useful for generate a PDF for a sample for manual inspection.
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".
...	Additional 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	<i>Plot the PCA of tumor and its best normal(s)</i>
----------------	---

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

<code>normal.coverage.files</code>	GATK coverage file of normal files, typically identified via findBestNormal .
<code>tumor.coverage.file</code>	GATK coverage file of a tumor sample.
<code>normalDB</code>	Database of normal samples, created with createNormalDatabase .
<code>x</code>	Principal component (PC) to be plotted on x-axis.
<code>y</code>	PC to be plotted on y-axis.
<code>col.tumor</code>	Color of tumor in plot.
<code>col.best.normal</code>	Color of best normals in plot.
<code>col.other.normals</code>	Color of other normals in plot.
<code>...</code>	Arguments passed to the <code>plot</code> 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 readCoverageGatk .
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

[readCoverageGatk](#)

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, readCoverageGatk),
  remove.chrs=c('chrX', 'chrY'))

```

predictSomatic	<i>Predict germline vs. somatic status</i>
----------------	--

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, cutoff = 0.1, return.vcf = FALSE)
```

Arguments

res	Return object of the runAbsoluteCN function.
id	Candidate solutions to be analyzed. id=1 will analyze the maximum likelihood solution.
cutoff	Exclude maternal/paternal chromosome number in segment if posterior probability is lower than cutoff.
return.vcf	Returns an annotated CollapsedVCF object. Note that this VCF will only contain variants not filtered out by the <code>filterVcf</code> functions. Variants outside segments or intervals might be included or not depending on runAbsoluteCN arguments.

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
purecn.snvs <- predictSomatic(purecn.example.output)

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

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:

- createExonWeightFile: [createTargetWeights](#)

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.

readCoverageGatk	<i>Read GATK coverage files</i>
------------------	---------------------------------

Description

Read coverage file produced by The Genome Analysis Toolkit or by [calculateBamCoverageByInterval](#).

Usage

```
readCoverageGatk(file)
```

Arguments

file Exon coverage file as produced by GATK.

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 <- readCoverageGatk(tumor.coverage.file)
```

readCurationFile	<i>Read curation file</i>
------------------	---------------------------

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 <code>runAbsoluteCN</code> function, serialized with <code>saveRDS</code> .
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 and allelic fractions of germline variants and somatic mutations. Coverage data is provided in GATK DepthOfCoverage format, allelic fraction in VCF format (e.g. obtained by MuTect). Normal control does not need to be matched (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, 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, coverage.cutoff = NULL, min.coverage = 15,
  max.coverage.vcf = 300, max.non.clonal = 0.2, max.homozygous.loss = 0.1,
  iterations = 30, log.ratio.calibration = 0.25,
  remove.off.target.snvs = TRUE, gc.gene.file = NULL,
  max.dropout = c(0.95, 1.1), max.logr.sdev = 0.75,
  max.segments = 200, plot.cnv = TRUE, cosmic.vcf.file = NULL,
  post.optimize = FALSE, verbose = TRUE, gatk.tumor.file = NULL,
  gatk.normal.file = NULL, ...)
```

Arguments

normal.coverage.file	GATK 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 readCoverageGatk function.
tumor.coverage.file	GATK 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 readCoverageGatk 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.
genome	Genome version, for example hg19.
centromeres	A data.frame with centromere positions in first three columns. 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 and verbose are required in the filter function and are automatically set (do NOT set them here again).
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 log.ratio, tumor, gc.data, seg.file, verbose are required and automatically set (do NOT set them here again).
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, min.coverage, sampleid, vcf, tumor.id.in.vcf, verbose are required in the segmentation function and automatically set (do NOT set them here again).
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.
coverage.cutoff	Deprecated, use min.coverage instead.
min.coverage	Minimum coverage in both normal and tumor. Targets with lower coverage are ingored.
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	Maximum genomic fraction assigned to homozygous loss. This is set to a fairly high default value to not exclude correct solutions, especially in noisy segmentations.
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.
log.ratio.calibration	Re-calibrate log-ratios in the window $sd(\log.ratio) * \log.ratio.calibration$.
remove.off.target.snvs	If set to a true value, will remove all SNVs outside the covered regions.
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 regions ($GC \geq 0.5$). High drop-out might indicate that data was not GC-normalized or that the sample quality might be insufficient. Requires gc.gene.file.
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.

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 SNVs 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.
post.optimize	Optimize purity using final SCNA-fit and SNVs. This might take a long time when lots of SNVs 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.
verbose	Verbose output.
gatk.tumor.file	Deprecated, use tumor.coverage.file instead.
gatk.normal.file	Deprecated, use normal.coverage.file instead.
...	Additional parameters passed to the segmentation function.

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

See Also

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

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,
  min.coverage, sampleid = sampleid, exon.weight.file = NULL,
  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, verbose = TRUE)

```

Arguments

<code>normal</code>	GATK coverage data for normal sample.
<code>tumor</code>	GATK coverage data for tumor sample.
<code>log.ratio</code>	Copy number log-ratios, one for each target in the coverage files.
<code>seg</code>	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.
<code>plot.cnv</code>	Segmentation plots.
<code>min.coverage</code>	Minimum coverage in both normal and tumor.
<code>sampleid</code>	Sample id, used in output files.
<code>exon.weight.file</code>	Can be used to assign weights to targets. Deprecated, use <code>target.weight.file</code> instead.
<code>target.weight.file</code>	Can be used to assign weights to targets.
<code>alpha</code>	Alpha value for CBS, see documentation for the segment function.

<code>undo.SD</code>	undo.SD for CBS, see documentation of the segment function. If NULL, try to find a sensible default.
<code>vcf</code>	Optional CollapsedVCF object with germline allelic ratios.
<code>tumor.id.in.vcf</code>	Id of tumor in case multiple samples are stored in VCF.
<code>normal.id.in.vcf</code>	Id of normal in in VCF. Currently not used.
<code>max.segments</code>	If not NULL, try a higher undo.SD parameter if number of segments exceeds the threshold.
<code>prune.hclust.h</code>	Height in the hclust pruning step. Increasing this value will merge segments more aggressively. If NULL, try to find a sensible default.
<code>prune.hclust.method</code>	Cluster method used in the hclust pruning step. See documentation for the hclust function.
<code>chr.hash</code>	Mapping of non-numerical chromosome names to numerical names (e.g. chr1 to 1, chr2 to 2, etc.). If NULL, assume chromosomes are properly ordered.
<code>verbose</code>	Verbose output.

Value

data.frame containing the segmentation.

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),
  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,
  min.coverage, sampleid = sampleid, exon.weight.file = 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,
  verbose = TRUE, ...)
```

Arguments

<code>normal</code>	GATK coverage data for normal sample.
<code>tumor</code>	GATK coverage data for tumor sample.
<code>log.ratio</code>	Copy number log-ratios, one for each exon in coverage file.
<code>seg</code>	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.
<code>plot.cnv</code>	Segmentation plots.
<code>min.coverage</code>	Minimum coverage in both normal and tumor.
<code>sampleid</code>	Sample id, used in output files.
<code>exon.weight.file</code>	Can be used to assign weights to exons. NOT SUPPORTED YET.
<code>flavor</code>	Flavor value for PSCBS. See <code>segmentByNonPairedPSCBS</code> .
<code>tauA</code>	<code>tauA</code> argument for PSCBS. See <code>segmentByNonPairedPSCBS</code> .
<code>vcf</code>	Optional VCF object with germline allelic ratios.
<code>tumor.id.in.vcf</code>	Id of tumor in case multiple samples are stored in VCF.
<code>normal.id.in.vcf</code>	Id of normal in in VCF. If NULL, use unpaired PSCBS.
<code>max.segments</code>	If not NULL, try a higher <code>undo.SD</code> parameter if number of segments exceeds the threshold.
<code>prune.hclust.h</code>	Height in the <code>hclust</code> pruning step. Increasing this value will merge segments more aggressively. If NULL, try to find a sensible default.
<code>prune.hclust.method</code>	Cluster method used in the <code>hclust</code> pruning step. See documentation for the <code>hclust</code> function.
<code>chr.hash</code>	Mapping of non-numerical chromosome names to numerical names (e.g. <code>chr1</code> to 1, <code>chr2</code> to 2, etc.). If NULL, assume chromosomes are properly ordered.
<code>verbose</code>	Verbose output.
<code>...</code>	Additional parameters passed to the <code>segmentByNonPairedPSCBS</code> function.

Value

A list with elements `seg` and `size`. "`seg`" contains the segmentation, "`size`" the size of all segments in base pairs.

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,
  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, max.bias = 1,
  normal.panel.vcf.file = NULL, min.normals = 10,
  verbose = TRUE)
```

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.
max.bias	Defines the maximum value for the mapping bias scaling factor. The default of 1 assumes that the reference allele can never have a lower mappability than the alt allele.
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.
verbose	Verbose output.

Value

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).

Author(s)

Markus Riester

Examples

```
# This function is typically only called by runAbsoluteCN via the
# fun.setMappingBiasVcf and args.setMappingBiasVcf comments.
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, 0.01), tumor.id.in.vcf = NULL, verbose = TRUE)
```

Arguments

<code>vcf</code>	CollapsedVCF object, read in with the <code>readVcf</code> function from the VariantAnnotation package.
<code>prior.somatic</code>	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 <code>vcf</code> contains a flag <code>Cosmic.CNT</code> , it will set the prior probability for variants with <code>CNT > 2</code> 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.
<code>tumor.id.in.vcf</code>	Id of tumor in case multiple samples are stored in VCF.
<code>verbose</code>	Verbose output.

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

*Topic **datasets**

- centromeres, 11
 - purecn.example.output, 30
 - segmentationCBS, 34, 36, 37
 - segmentationPSCBS, 39
 - setMappingBiasVcf, 34, 40
 - setPriorVcf, 34, 41
- autoCurateResults, 2
- bootstrapResults, 3
- calculateBamCoverageByInterval, 4, 31
- calculateGCCContentByInterval, 4, 5, 9, 12, 35
- calculateLogRatio, 6
- calculatePowerDetectSomatic, 6, 19
- callAlterations, 8, 9
- callAlterationsFromSegmentation, 9
- callLOH, 10
- centromeres, 11
- correctCoverageBias, 4, 11, 33, 36
- createCurationFile, 12, 32
- createExonWeightFile, 13
- createNormalDatabase, 14, 17, 21, 27
- createSNPBlacklist, 15, 18
- createTargetWeights, 13, 16, 30
- filterTargets, 17, 34
- filterVcfBasic, 18, 20, 34
- filterVcfMuTect, 19, 34
- findBestNormal, 14, 20, 27
- findFocal, 9, 21, 34
- getDiploid, 22
- getSexFromCoverage, 21, 23, 25, 34
- getSexFromVcf, 24, 24
- plotAbs, 25
- plotBestNormal, 27
- poolCoverage, 28
- predictSomatic, 29
- PureCN-deprecated, 30
- purecn.example.output, 30
- readCoverageGatk, 4, 6, 11, 12, 24, 28, 31, 33
- readCurationFile, 31
- runAbsoluteCN, 2–4, 6, 8, 10, 12, 13, 17, 21–23, 25, 26, 29–32, 32, 37–40