Package 'exomeCopy'

March 22, 2023

Type Package

Title Copy number variant detection from exome sequencing read depth

Version 1.44.0

Date 2021-11-20

Author Michael Love

Maintainer Michael Love <michaelisaiahlove@gmail.com>

Description Detection of copy number variants (CNV) from exome sequencing samples, including unpaired samples. The package implements a hidden Markov model which uses positional covariates, such as background read depth and GC-content, to simultaneously normalize and segment the samples into regions of constant copy count.

License GPL (>= 2)

LazyLoad yes

Imports stats4, methods, GenomeInfoDb

Depends IRanges (>= 2.5.27), GenomicRanges (>= 1.23.16), Rsamtools

Suggests Biostrings

biocViews CopyNumberVariation, Sequencing, Genetics

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

git_branch RELEASE_3_16

git_last_commit 2dd6598

git_last_commit_date 2022-11-01

Date/Publication 2023-03-22

R topics documented:

`exomeCopy-package` *Copy number variant detection from exome sequencing read depth*

Description

Detection of copy number variants (CNV) from exome sequencing samples, including unpaired samples. The package implements a hidden Markov model which uses positional covariates, such as background read depth and GC-content, to simultaneously normalize and segment the samples into regions of constant copy count.

Details

exomeCopy fits a hidden Markov model to observed read counts using covariates. It returns the Viterbi path, the most likely path of hidden states, which is the predicted copy count at each window.

`exomeCopy` is designed to run on read counts from consecutive genomic ranges on a single chromosome, as it tries to identify higher or lower read depth relative to a baseline. Please see the vignette for an example of how to prepare input data for `exomeCopy`, how to loop the function over multiple chromosomes and samples, and how to extract the resulting predicted CNVs.

References

Love, Michael I.; Mysickova, Alena; Sun, Ruping; Kalscheuer, Vera; Vingron, Martin; and Haas, Stefan A. (2011) "Modeling Read Counts for CNV Detection in Exome Sequencing Data," Statistical Applications in Genetics and Molecular Biology: Vol. 10 : Iss. 1, Article 52. DOI: 10.2202/1544-6115.1732 http://cmb.molgen.mpg.de/publications/Love_2011_exomeCopy.pdf.

See Also

`exomeCopy`

compileCopyCountSegments *Compile segments across samples*

Description

A short function which extracts the segments of constant copy count using the copyCountSegments function on a named list of named lists containing fitted ExomeCopy objects. See vignette for a full example of multiple samples and chromosomes.

Usage

```
compileCopyCountSegments(fit.list)
```

Arguments

fit.list	A named list of named lists. The outer list indexes patients, while the inner list indexes sequences/chromosomes.

Value

A GRanges object of all segments across samples and chromosomes.

Examples

```
example(exomeCopy)

# this function requires a named list of named lists
# as constructed in the vignette
fit.list <- list(sample1 = list(chr1 = fit))
CNV.segments <- compileCopyCountSegments(fit.list)
```

copyCountSegments *Segments of identical copy count from exomeCopy*

Description

Unpacks an ExomeCopy object and returns a GRanges object with segments of identical predicted copy count in genomic coordinates.

Usage

```
copyCountSegments(object)
```

Arguments

object	ExomeCopy object

Details

The log odds column is calculated by summing the log ratios over the contained ranges. The log ratios at each range is the log of the emission probability for the given read count for the predicted state divided by the emission probability for the normal state. The higher the value, the more likely that the read counts in this range could not have been generated from the normal state.

Value

Returns a GRanges object with the predicted copy count, the log odds of predicted copy count over normal copy count, a combined p-value, the number of genomic ranges spanned by the segment, the number of targeted basepairs in the segment, and the sample name.

See Also

exomeCopy ExomeCopy-class

Examples

```
example(exomeCopy)
copyCountSegments(fit)
```

---

countBamInGRanges *Count reads from BAM file in genomic ranges*

Description

Counts the number of reads with a specified minimum mapping quality from BAM files in genomic ranges specified by a GRanges object. This is a convenience function for counting the reads in ranges covering the targeted regions, such as the exons in exome enrichment experiments, from each sample. These read counts are used by exomeCopy in predicting CNVs in samples.

With the default setting (read.width=1), only the read starts are used for counting purposes (the leftmost position regardless of the strandedness of the read).

With the accurate read width, or with get.width = TRUE, then the function returns the number of overlapping reads, as returned by countOverlaps in the GenomicRanges package.

The function subdivideGRanges can be used first to subdivide ranges of different size into ranges of nearly equal width.

The BAM file requires a associated index file (see the man page for indexBam in the Rsamtools package).

Usage

```
countBamInGRanges(bam.file,granges,min.mapq=1,read.width=1,stranded.start=FALSE,get.width=FALSE,re
```

Arguments

bam.file	The path of the BAM file for the sample to be counted.
granges	An object of type GRanges with the ranges in which to count reads.
min.mapq	The minimum mapping quality to count a read. Defaults to 1. Set to 0 for counting all reads.
read.width	The width of a read, used in counting overlaps of mapped reads with the genomic ranges. The default is 1, resulting in the counting of only read starts in genomic ranges. If the length of fixed width reads is used, e.g. 100 for 100bp reads, then the function will return the count of all overlapping reads with the genomic ranges. However, counting all overlapping reads introduces dependency between the counts in adjacent windows.
stranded.start	If true, the function will create reads of length read.width using the strand to determine the read location. A read with + or * strand will start at the given start position, and a read with - strand will end at (start position + CIGAR width - 1).
get.width	If true, the function should retrieve the read width from the CIGAR encoding rather than assign the value from read.width.
remove.dup	If true, the function will count only one read for each unique combination of position, strand and read width.

Value

An integer vector giving the number of reads over the input GRanges

See Also

Rsamtools GRanges subdivideGRanges

Examples

```
## get subdivided genomic ranges covering targeted region
## using subdivideGRanges()
example(subdivideGRanges)

## BAM file included in Rsamtools package
bam.file <- system.file("extdata", "mapping.bam", package="exomeCopy")

## extract read counts from the BAM file in these genomic ranges
mcols(target.sub)$sample <- countBamInGRanges(bam.file,target.sub)
```

---

exomeCopy *Fit the exomeCopy or exomeCopyVar model to the observed counts.*

Description

Fits a hidden Markov model to observed read counts using positional covariates. It returns an object containing the fitted parameters and the Viterbi path, the most likely path of hidden states, which is the predicted copy count at each window.

exomeCopy is designed to run on read counts from consecutive genomic ranges on a single chromosome, as it tries to identify higher or lower read depth relative to a baseline. Please see the vignette for an example of how to prepare input data for exomeCopy, how to loop the function over multiple chromosomes and samples, and how to extract the resulting predicted CNVs.

exomeCopy requires as input a GRanges object containing read counts in genomic ranges along with the covariates. Some convenience functions are provided for preparing input for exomeCopy:

1. subdivideGRanges, to subdivide a GRanges object containing the genomic ranges of the targeted region into genomic ranges of nearly equal width,
2. countBamInGRanges, to count the number of read starts from a BAM read mapping file in a GRanges object,
3. getGCcontent, to get the GC-content in the ranges given a FASTA file of the reference sequence,
4. generateBackground, to calculate median normalized read depth over a set of control samples, and also any statistic over normalized read depth.

Usage

```
  exomeCopy(gr, sample.name, X.names, Y.names, fit.var=FALSE, reltol
  = 0.0001, S = 0:4, d = 2, goto.cnv = 1e-4, goto.normal = 1/20,
init.phi="norm")
```

Arguments

gr	A GRanges object with the sample counts and positional covariates over the genomic ranges.
sample.name	The name of the metadata column of gr with the sample read counts.
X.names	The names of the metadata columns of gr with covariates for estimating mu.
Y.names	(optional) the names of the metadata columns of gr with covariates for estimating phi, only required if fit.var = TRUE.
fit.var	A logical, whether the model should fit the overdispersion parameter phi with a linear combination of covariates (exomeCopyVar) or with a scalar (exomeCopy). Defaults to FALSE (exomeCopy).
reltol	The relative tolerance for convergence used in the optim function for optimizing the parameter settings. From testing, the default value was sufficient for fitting parameters, but lower relative tolerances can be used.
S	A vector of possible copy numbers for the different states.
d	The expected copy number for the normal state. This should be set to 2 for autosomes and 1 for haploid data.
goto.cnv	The initial setting for probability to transfer to a CNV state.

`goto.normal`	The initial setting for probability to transfer to a normal state.
`init.phi`	Either "norm" or "counts": initialize phi with the moment estimate using residuals from a linear model of read counts on covariates or with the raw counts.

Details

`exomeCopy` fits transitional and emission parameters of an HMM to best explain the observed counts of a sample from exome or targeted sequencing. The set of underlying copy number states, S, in the sample must be provided before running the algorithm.

The emission probabilities are given as a negative binomial distribution using positional covariates, such as log background read depth, quadratic terms for GC-content, and range width, which are stored in a matrix X. Optionally, for fitting the variance of the distribution, the standard deviation and/or variance of the background set can be included in a matrix Y. All covariates are normalized within `exomeCopy` for improved optimization.

For the observed count at range t, O_t, the emission probability is given by:

$$O_t \sim \text{NB}(\mu_{ti}, \phi)$$

The mean parameter μ_{ti} is given by:

$$\mu_{ti} = \frac{S_i}{d} e^{(x_{t*}\vec{\beta})}$$

Here S_i is the i-th possible copy number state, d is the expected background copy number (d = 2 for diploid sequence), and $\vec{\beta}$ is a vector of coefficients fitted by the model. x_{t*} is the t-th row of the matrix X.

μ must be positive, so it is replaced with a small positive number if the value is less than zero.

For exomeCopyVar, which also fits the variance, the emission probability includes a location-dependent dispersion parameter.

where

$$\log(\phi_t) = y_{t*}\vec{\gamma}$$

or a small positive number if this is less than zero.

Two transition probabilities are fitted in the model: the probabilities of transitioning to a normal state and to a CNV state.

`exomeCopy` calls `negLogLike` to evaluate the likelihood of the HMM. The parameters are fit using Nelder-Mead optimization with the `optim` function on the negative likelihood. The Viterbi path is calculated by calling `viterbiPath`.

Value

Returns an `ExomeCopy-class` object. See this page for the slot descriptions. Also see the vignette and `copyCountSegments` on how to extract segments.

Author(s)

Michael Love

References

Love, Michael I.; Mysickova, Alena; Sun, Ruping; Kalscheuer, Vera; Vingron, Martin; and Haas, Stefan A. (2011) "Modeling Read Counts for CNV Detection in Exome Sequencing Data," Statistical Applications in Genetics and Molecular Biology: Vol. 10 : Iss. 1, Article 52. DOI: 10.2202/1544-6115.1732 http://cmb.molgen.mpg.de/publications/Love_2011_exomeCopy.pdf.

References for HMM algorithms and use of HMM for segmentation of genomic data by copy number:

Rabiner, L. R. (1989): "A tutorial on hidden Markov models and selected applications in speech recognition," Proceedings of the IEEE, 77, 257, 286, http://dx.doi.org/10.1109/5.18626.

Fridlyand, J., A. M. Snijders, D. Pinkel, D. G. Albertson, and Jain (2004): "Hidden Markov models approach to the analysis of array CGH data," Journal of Multivariate Analysis, 90, 132, 153, http://dx.doi.org/10.1016/j.jmva.2004.02.008.

Marioni, J. C., N. P. Thorne, and S. Tavare (2006):"BioHMM: a heterogeneous hidden Markov model for segmenting array CGH data." Bioinformatics, 22, 1144, 1146, http://view.ncbi.nlm.nih.gov/pubmed/16533818.

See Also

ExomeCopy-class subdivideGRanges countBamInGRanges copyCountSegments plot.ExomeCopy negLogLike IRanges GRanges

Examples

```
## The following is an example of running exomeCopy on simulated
## read counts using the model parameters defined above.  For an example
## using real exome sequencing read counts (with simulated CNV) please
## see the vignette.

## create GRanges for storing genomic ranges and covariate data
## (background, background stdev, GC-content)

m <- 5000
gr <- GRanges("chr1", IRanges(start=0:(m-1)*100+1,width=100),
              log.bg=rnorm(m), log.bg.var=rnorm(m), gc=runif(m,30,50))
genome(gr) <- "hg19"

## create read depth distributional parameters mu and phi
gr$gc.sq <- gr$gc^2
X <- cbind(bg=gr$log.bg,gc=gr$gc,gc.sq=gr$gc.sq)
Y <- cbind(bg.sd=gr$log.bg.var)
beta <- c(5,1,.01,-.01)
gamma <- c(-3,.1)
gr$mu <- exp(beta[1] + scale(X) %*% beta[2:4])
gr$phi <- exp(gamma[1] + scale(Y) %*% gamma[2])

## create observed counts with simulated heterozygous duplication
cnv.nranges <- 200
bounds <- (round(m/2)+1):(round(m/2)+cnv.nranges)
```

```
O <- rnbinom(length(gr),mu=gr$mu,size=1/gr$phi)
O[bounds] <- O[bounds] + rbinom(cnv.nranges,prob=0.5,size=O[bounds])
gr$sample1 <- O

## run exomeCopy() and list segments
fit <- exomeCopy(gr,"sample1",X.names=c("log.bg","gc","gc.sq"))

# an example call with variance fitting.
# see paper: this does not necessarily improve the fit
fit <- exomeCopy(gr,"sample1",X.names=c("log.bg","gc","gc.sq"),
                 Y.names="log.bg",fit.var=TRUE)

## see man page for copyCountSegments() for summary of
## the predicted segments of constant copy count, and
## for plot.ExomeCopy() for plotting fitted objects
```

`ExomeCopy-class` *Class "ExomeCopy"*

Description

Object returned by exomeCopy

Objects from the Class

Objects can be created by calls of the form `new("ExomeCopy")`.

Slots

`sample.name`: Object of class `"character"`: an identifier for the sample

`type`: Object of class `"character"`: the type of model used, either "exomeCopy" or "exomeCopyVar"

`path`: Object of class `"Rle"`: the index of the predicted state for each window

`ranges`: Object of class `"IRangesList"`: the corresponding ranges for the observed counts and covariates

`O.norm`: Object of class `"numeric"`: the input vector of counts divided by X * beta

`log.odds`: Object of class `"numeric"`: the log ratio of the emission probabilities along the predicted path over the emission probabilities of the normal state

`fx.par`: Object of class `"list"`: a list of the settings S, d, normal.state and fit.var

`init.par`: Object of class `"list"`: a list of the initial parameters goto.cnv, goto.normal, beta.hat and phi.hat

`final.par`: Object of class `"list"`: a list of the final parameters goto.cnv, goto.normal, beta (and gamma for exomeCopyVar)

`counts`: Object of class `"numeric"`: the number of evaluations of the log likelihood performed by `optim`

`convergence`: Object of class `"numeric"`: the integer for convergence of `optim`, 0 for convergence

`nll`: Object of class `"numeric"`: the final value of the negative log likelihood

Methods

plot `signature(x = "ExomeCopy", y = "missing")`: ...

show `signature(object = "ExomeCopy")`: ...

See Also

`exomeCopy`

Examples

```
showClass("ExomeCopy")
```

`exomecounts` *Sample counts from 16 exome sequencing samples from 1000 Genomes Project*

Description

This data set gives sample read counts in 1000 genomic ranges for 16 exome sequencing samples from the PUR population of the 1000 Genomes Project, along with the GC-content in the ranges. For instructions on how to prepare read count and covariate data, please see the example code in the man pages for `subdivideGRanges` and `countBamInGRanges`.

The genomic ranges are generated from small portion of the CCDS regions of chromosome 1 (hg19). The CCDS regions are subdivided evenly into ranges around 100bp using the `subdivideGRanges` function with default settings. Only ranges with positive counts across samples are retained. These regions were downloaded as a BED file from the UCSC Genome Browser (`http://genome.ucsc.edu/cgi-bin/hgGateway`). The mapping files for the exome sequencing data and descriptions of the experiments are available at the 1000 Genomes Project website (`http://www.1000genomes.org/data`). The directories used are listed in the file `1000Genomes_files.txt` in the `extdata` directory.

The column names are the sample names from the 1000 Genomes Project. Library format is paired-end reads and sample counts reflect both sequenced reads counted in their respective genomic ranges.

Usage

```
data(exomecounts)
```

Format

A GRanges object.

Source

1000 Genomes Project and Consensus Coding Sequence Project

References

1000 Genomes Project Consortium. A map of human genome variation from population-scale sequencing. Nature 467, 1061-1073 (2010). http://dx.doi.org/10.1038/nature09534.

1000 Genomes Project: Release of phase 1 exome alignments http://www.1000genomes.org/announcements/release-phase-1-exome-alignments-2011-07-19

Pruitt, K. D. et al. The consensus coding sequence (CCDS) project: Identifying a common protein-coding gene set for the human and mouse genomes. Genome research 19, 1316-1323 (2009). http://dx.doi.org/10.1101/gr.080531.108.

See Also

GRanges

`generateBackground` *Generate median background read depth*

Description

Normalizes a set of columns representing read counts from different samples by their mean. Then calculates a statistic across the rows of normalized counts.

Usage

```
generateBackground(sample.names, gr, fn=median)
```

Arguments

`sample.names`	A vector of metadata column names in gr to be used as background samples
`gr`	A GRanges object containing the read counts
`fn`	The statistic to be applied across the rows of normalized counts. Defaults to median, but the standard deviation can also be calculated in this way.

Value

The value of fn applied across the rows of normalized read counts.

Examples

```
data(exomecounts)
sample.names <- grep("HG.+",colnames(mcols(exomecounts)),value=TRUE)
exomecounts$bg <- generateBackground(sample.names, exomecounts, median)
exomecounts$bg.sd <- generateBackground(sample.names, exomecounts, sd)
```

getGCcontent *Get the GC content of target ranges from a reference FASTA file*

Description

A short function using scanFa from the Rsamtools package on a target GRanges and a reference FASTA file

Usage

```
getGCcontent(target, reference.file)
```

Arguments

target	GRanges object
reference.file	the path to the reference FASTA file

Value

Returns a vector of the ratio of G and C basepairs to total basepairs (not counting N's).

Examples

```
target.file <- system.file("extdata","targets.bed",package="exomeCopy")
target.df <- read.delim(target.file,header=FALSE,col.names=c("seqname","start","end"))
target <- GRanges(seqname=target.df$seqname,IRanges(start=target.df$start+1,end=target.df$end))
reference.file <- system.file("extdata","reference.fa",package="exomeCopy")
GCcontent <- getGCcontent(target, reference.file)
```

negLogLike *Generalized negative log likelihood and Viterbi algorithms*

Description

negLogLike: Returns the negative log likelihood calculated with the forward equations.

viterbiPath: Calculates the most likely sequence of hidden states for the Markov model given the current parameters.

Usage

```
negLogLike(par,fx.par,data,nstates,stFn,trFn,emFn)
viterbiPath(par,fx.par,data,nstates,stFn,trFn,emFn)
```

Arguments

par	A list of parameters, over which the likelihood will be optimized.
fx.par	A list of fixed parameters.
data	A list of data objects, which must contain a vector O, which represents the observed sequence of the HMM.
nstates	The number of states of the HMM.
stFn	A function which takes arguments par, fx.par, data, and nstates, and returns a vector of length nstates of starting probabilities.
trFn	A function which takes arguments par, fx.par, data, and nstates, and returns a matrix of dimension (nstates,nstates) of the transition probabilities.
emFn	A function which takes arguments par, fx.par, data, and nstates, and returns a matrix of dimension (nstates,length(O)) of the emission probabilities.

Value

negLogLike: The negative log likelihood of the HMM. The likelihood is slightly modified to account for ranges with read counts which have zero probability of originating from any of the states. In this case the likelihood is lowered and the range is skipped.

viterbiPath: The Viterbi path through the states given the parameters.

References

On the forward equations and the Viterbi algorithm:

Rabiner, L. R. (1989): "A tutorial on hidden Markov models and selected applications in speech recognition," Proceedings of the IEEE, 77, 257, 286, http://dx.doi.org/10.1109/5.18626.

Examples

```
## functions for starting, transition, and emission probabilities
stFn <- function(par,fx.par,data,nstates) rep(1/nstates,nstates)
trFn <- function(par,fx.par,data,nstates) {
  A <- matrix(1/(nstates*10),ncol=nstates,nrow=nstates)
  diag(A) <- 1 - rowSums(A)
  A
}
emFn <- function(par,fx.par,data,nstates) {
  t(sapply(1:nstates,function(j) dnorm(data$O,par$means[j],fx.par$sdev)))
}

## simulate some observations from two states
Q <- c(rep(1,100),rep(2,100),rep(1,100),rep(2,100))
T <- length(Q)
means <- c(-0.5,0.5)
sdev <- 1
O <- rnorm(T,means[Q],sdev)

## use viterbiPath() to recover the state chain using parameters
viterbi.path <- viterbiPath(par=list(means=means),
```

```
fx.par=list(sdev=sdev), data=list(O=O), nstates=2,stFn,trFn,emFn)
plot(O,pch=Q,col=c("darkgreen","orange")[viterbi.path])
```

`plot.ExomeCopy` *Plot function for exomeCopy*

Description

Plots the predicted copy count segments of an ExomeCopy object

Usage

```
## S3 method for class 'ExomeCopy'
plot(x, points = TRUE, cols = NULL, show.legend = TRUE,
main = "exomeCopy predicted segments", xlab = "genomic position",
ylab = "normalized read count", xlim = NULL, ylim = NULL, cex = 1, lwd = 4, ...)
```

Arguments

`x`	The ExomeCopy object.
`points`	Logical, whether normalized read counts should be drawn.
`cols`	A vector of the same length as b, specifying a color for each of the states of the HMM.
`show.legend`	Logical, whether a default legend should be shown.
`main`	main title
`xlab`	x axis label
`ylab`	y axis label
`xlim`	x limits
`ylim`	y limits
`cex`	size of the points (if plotted)
`lwd`	line width
`...`	Other arguments passed to plot()

See Also

`exomeCopy` `ExomeCopy-class` `copyCountSegments`

Examples

```
example(exomeCopy)
plot(fit)
```

plotCompiledCNV *Plot compiled CNV segments for one sequence/chromosome*

Description

This function takes a GRanges object as produced by compileCopyCountSegments and plots the CNV segments for one sequence/chromosomes across the samples with CNV segments.

The segments in the normal state should be removed as shown below in the example to produce a cleaned GRanges object. See the vignette for a more complete example.

Usage

```
plotCompiledCNV(CNV.segments, seq.name, xlim=NULL, col=NULL,
copy.counts=0:6, normal.state = 2)
```

Arguments

CNV.segments	A GRanges object as produced by compileCopyCountSegments and with normal state removed.
seq.name	The name of the sequence to plot
xlim	The genomic coordinates for the x axis. If not included, the plotting window will cover the range of the CNVs in CNV.segments
col	The colors to use for the different copy count states
copy.counts	The corresponding copy counts for the colors
normal.state	The copy count of the normal state

Value

Produces a plot.

Examples

```
example(compileCopyCountSegments)
CNV.clean <- CNV.segments[CNV.segments$copy.count != 2]
chr.start <- start(range(fit@ranges))
chr.end <- end(range(fit@ranges))
plotCompiledCNV(CNV.clean, "chr1", xlim=c(chr.start,chr.end))
```

subdivideGRanges *Subdivide ranges of a GRanges object into nearly equal width ranges*

Description

Takes an input GRanges object and, splits each range into multiple ranges of nearly equal width. For an input range of width `w` and subdividing size `s`, it will subdivide the range into `max(1,floor(w/s))` nearly equal width ranges. The output is then a new GRanges object. This function can be used to split the targeted region (such as exons in exome enrichment experiments) into nearly equal width ranges. The ranges will be sorted and reduced if they are not already so.

Usage

```
subdivideGRanges(x,subsize=100)
```

Arguments

`x`	An object of type GRanges.
`subsize`	The desired width for the ranges in the output GRanges object.

Value

A GRanges object with ranges from the input GRanges object subdivided to nearly `subsize`.

See Also

GRanges

Examples

```
  ## read in target region BED file
  target.file <- system.file("extdata", "targets.bed", package="exomeCopy")
  target.df <- read.delim(target.file, header=FALSE,
col.names=c("seqname","start","end"))

  ## create GRanges object with 5 ranges over 2 sequences
  target <- GRanges(seqname=target.df$seqname,
              IRanges(start=target.df$start,end=target.df$end))

  ## subdivide into 7 smaller genomic ranges
  target.sub <- subdivideGRanges(target)
```

Index