Package 'HMMcopy'

January 2, 2025

Type Package

Title Copy number prediction with correction for GC and mappability bias for HTS data

Version 1.49.0

Date 2021-11-22

Author Daniel Lai, Gavin Ha, Sohrab Shah

Maintainer Daniel Lai <dalai@bccrc.ca>

Import data.table

Depends R (>= 2.10.0), data.table (>= 1.11.8)

Description Corrects GC and mappability biases for readcounts (i.e. coverage) in non-overlapping windows of fixed length for single whole genome samples, yielding a rough estimate of copy number for furthur analysis. Designed for rapid correction of high coverage whole genome tumour and normal samples.

License GPL-3

biocViews Sequencing, Preprocessing, Visualization, CopyNumberVariation, Microarray

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

git_branch devel

git_last_commit 9dbb51b

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2025-01-02

Contents

HMMcopy-package 2
correctReadcount 3
HMMcopy example dataset 5
HMMcopy internal functions 6

HMMcopy Segmentation 6
HMMcopy Visualization 9
WIG Import Functions 10
WIG Output Functions 11
wigsToRangedData 12

Index **14**

`HMMcopy-package` *Bias-free copy number estimation and robust CNA detection in tumour samples from WGS HTS data*

Description

HMMcopy is a package for making bias-free copy number estimation by correcting for GC-content and mappability bias in HTS readcounts. It also contains an implementation of the Hidden Markov Model to robustly segment a copy number profile into non-overlapping segments predicted to be of the same copy number state, and attributes a biological copy number aberration events to the segments.

Details

HMMcopy takes as input WIG format files generated by fast C++ tools distributed as part of the *HMMcopy Suite*, namely readcount, GC-content and mappability values for non-overlapping fixed width "bins" across the reference genome of interest. It then uses a filtering and LOESS model to correct the GC-content and mappability biases observed in the readcounts (Benjamini and Speed, 2012), and uses the corrected readcounts as a proxy of copy number. The resultant copy number profile is then segmented with a six state Hidden Markov Model, with a handful of quick visualization functions for quick viewing.

Package:	HMMcopy
Type:	Package
Version:	0.1.0
Date:	2011-09-06
License:	GPL-3

`example("HMMcopy-package")` for quick tour of functionality and visualization

`vignette("HMMcopy")` for detailed example

Author(s)

Daniel Lai, Gavin Ha, Sohrab Shah

Maintainer: Daniel Lai <jujubix@cs.ubc.ca> and Gavin Ha <gha@bccrc.ca>

References

Yuval Benjamini and Terence P Speed. Summarizing and correcting the gc content bias in high-throughput sequencing. Nucleic Acids Res, **40(10)**:e72, May 2012.

Examples

```
# Read WIG file input
rfile <- system.file("extdata", "tumour.wig", package = "HMMcopy")
gfile <- system.file("extdata", "gc.wig", package = "HMMcopy")
mfile <- system.file("extdata", "map.wig", package = "HMMcopy")
uncorrected_reads <- wigsToRangedData(rfile, gfile, mfile)

# Correct reads into copy number
corrected_copy <- correctReadcount(uncorrected_reads)

# Segment copy number profile
segmented_copy <- HMMsegment(corrected_copy)

# Visualize one at a time
par(ask = TRUE)
plotBias(corrected_copy)
plotCorrection(corrected_copy)
plotSegments(corrected_copy, segmented_copy)
```

`correctReadcount` *Readcount correction for GC and mappability bias*

Description

Corrects readcounts for GC and mappability bias using the binning/loess method optimized for speed.

Usage

```
correctReadcount(x, mappability = 0.9, samplesize = 50000, verbose = TRUE)
```

Arguments

`x`	`RangedData` object returned by `wigsToRangedData`
`mappability`	Mappability threshold [0, 1] below which points are ignored during creating the correction curve.
`samplesize`	The number of points sampled during LOESS fitting, decreasing reduces runtime and memory usage, at the expense of robustness to data randomness.
`verbose`	Set to FALSE it messages are not desired.

Details

Input read counts are contained in the IRanges object, where number of reads within bins (or sometimes called windows) of defined genomic size are specified. GC content should also be computed using the exact same boundaries for each bin.

Ensure that the GC content and mappability files have been mapped to the same genome build (e.g. hg18) as the tumour and normal read libraries.

Here is a summary of the correction procedure.

1. Filter out bins with 0 reads and 0 GC content
2. Filter out bins with reads within the top and bottom 1% quantile (assumed to be outliers)
3. Filter out bins with GC content within the top and bottom 1% quantile
4. Filter out bins with a mappability score of greater than 0.9 ('mappability' argument).
5. Randomly sample up to 50000 ('samplesize' argument) of the remaining high-quality bins for the purposes keeping a good runtime.
6. The first loess (on the reads-by-GC curve) with a small span (smoothing window) is performed, obtaining typically a highly sensitive curve (follows low density tails of distribution, but gets jagged in high density center).
7. A second loess (on the first loess results) with a larger span is performed, recapitulating the curve in the low density tails and smoothing out the jagged regions in the high density center.
8. 'cor.gc' is obtained by correcting each bin for GC content. The number of observed reads is divided by the number of reads predicted by the loess curve given an observed GC proportion.
9. Filter out just the top 1% quantile of the cor.gc bins, then _randomly_ sample up to 50000 ('samplesize' argument) bins.
10. A separate lowess curve is computed for mappability-by-GC (cor.gc).
11. 'cor.map' is obtained by correcting each bin for mappability. The cor.gc value is divided by the cor.gc value predicted by the mappability lowess curve generated in the previous step.
12. 'copy' is obtained by setting all cor.map values <= to NA (i.e. NaN), then apply log2

Value

The original A `RangedData` object appended with 5 new columns:

valid Valid bins, which have valid read, gc, and mappability values

ideal Ideal bins of high mappability and no outliers

cor.gc GC-corrected readcounts

cor.map Mappability corrected GC-corrected readcounts

copy cor.map values after log base 2

Author(s)

Daniel Lai

References

Yuval Benjamini and Terence P Speed. Summarizing and correcting the gc content bias in high-throughput sequencing. Nucleic Acids Res, **40(10)**:e72, May 2012.

See Also

`wigsToRangedData` to easily generate the proper input.

Examples

```
data(tumour) # Load tumour_reads
tumour_copy <- correctReadcount(tumour_reads)
```

`HMMcopy example dataset`

HMMcopy example dataset

Description

A set of data of chromosome 6 of matched tumour normal pair. The dataset that is described here belongs to a female triple-negative breast cancer patient from the dataset published in *Shah2012,Ha2012*. This genome library was sequenced on the ABI/Life SOLiD platform, generating hybrid lengths of 25bp and 50bp paired-end reads. The reads were aligned using BioScope https://products.appliedbiosystems.com/ where reads mapped to multiple sites were ignored.

tumour_reads The number of short reads in fixed width windows across the chromosome, generated with `wigsToRangedData` from WIG files

tumour_copy Tumour copy number profile generated by correcting 'tumour_reads' with `correctReadcount`

normal_copy Normal copy number profile generated via `correctReadcount`

tumour_param Parameters for segmenting 'tumour_copy' in `HMMsegment`

tumour_segments Segmented output of 'tumour_copy' `HMMsegment` using 'tumour_param'

Usage

```
data(tumour)
```

Format

'tumour_reads', 'tumour_copy', and 'normal_copy' are `RangedData` objects.

'tumour_param' is a numeric matrix.

'tumour_segments' is a list.

References

Gavin Ha, Andrew Roth, Daniel Lai, Ali Bashashati, Jiarui Ding, Rodrigo Goya, Ryan Giuliany, Jamie Rosner, Arusha Oloumi, Karey Shumansky, Suet-Feung Chin, Gulisa Turashvili, Martin Hirst, Carlos Caldas, Marco A Marra, Samuel Aparicio, and Sohrab P Shah. Integrative analysis of genome-wide loss of heterozygosity and mono-allelic expression at nucleotide resolution reveals disrupted pathways in triple negative breast cancer. Genome Research, (advanced online publication), May 2012.

S P Shah, A Roth, R Goya, A Oloumi, G Ha, Y Zhao, G Turashvili, J Ding, K Tse, G Hafari, A Bashashati, L M Prentice, J Khattra, A Burleigh, D Yap, V Bernard, A McPherson, K Shumansky, A Crisan, R Giuliany, A Heravi-Moussavi, J Rosner, D Lai, I Birol, R Varhol, A Tam, N Dhalla, T Zeng, K Ma, S K Chan, M Griffth, A Moradian, S W Cheng, G B Morin, P Watson,

K Gelmon, S Chia, S F Chin, C Curtis, O M Rueda, P D Pharoah, S Damaraju, J Mackey, K Hoon, T Harkins, V Tadigotla, M Sigaroudinia, P Gascard, T Tlsty, J F Costello, I M Meyer, C J Eaves, W W Wasserman, S Jones, D Huntsman, M Hirst, C Caldas, M A Marra, and S Aparicio. The clonal and mutational evolution spectrum of primary triple-negative breast cancers. Nature, 486(7403):395-399, Jun 2012.

`HMMcopy internal functions` *HMMcopy internal functions*

Description

HMMcopy internal functions for HMMsegment

Usage

```
normalize(A)
dirichletpdf(x, alpha)
tdistPDF(x, mu, lambda, nu)
estimateTNoiseParamsMap(y, mu, lambda, nu, rho, eta, m, gamma, S, kappa)
```

Details

Not for users

`HMMcopy Segmentation` *Segmentation and classification of copy number profiles*

Description

Takes in a copy number profile and segments it into predicted regions of equal copy number, and assigns a biologically motivated copy number state to each region using a Hidden Markov Model (HMM). This is an extension to the HMM described in Shah et al., 2006.

Usage

```
HMMsegment(correctOut, param = NULL, autosomes = NULL,
    maxiter = 50, getparam = FALSE, verbose = TRUE)
```

Arguments

`correctOut`	Output value from `correctReadcount`
`param`	If none is provided, will generate a reasonable set of parameters based on the input data, which can optionally be returned for inspection and manual adjustment by setting 'getparam' to TRUE. See Details for more information on parameters. A matrix with parameters values in columns for each state in rows: **e** Probability of extending a segment, increase to lengthen segments, decrase to shorten segments. Range: (0, 1) **strength** Strength of initial e suggestion, reducing allows e to change, increasing makes e undefiable. Range: [0, Inf) **mu** Suggested median for copy numbers in state, change to readjust classification of states. Range: (-Inf, Inf) **lambda** Suggested precision (inversed variance) for copy numbers in state, increase to reduce overlap between states. Range: [0, Inf) **nu** Suggested degree of freedom between states, increase to reduce overlap between states. Range: [0, Inf) **kappa** Suggested distribution of states. Should sum to 1. **m** Optimal value for mu, difference from corresponding mu value determines elasticity of the mu value. *i.e.* Set to identical value as mu if you don't want mu to move much. **eta** Mobility of mu, increase to allow more movement. Range: [0, Inf) **gamma** Prior shape on lambda, gamma distribution. Effects flexibility of lambda. **S** Prior scale on lambda, gamma distribution. Effects flexibility of lambda.
`autosomes`	Array of LOGICAL values corresponding to the 'chr' argument where an element is TRUE if the chromosome is an autosome, otherwise FALSE. If not provided, will automatically set the following chromosomes to false: "X", "Y", "23", "24", "chrX", chrY", "M", "MT", "chrM".
`maxiter`	The maximum number of iterations allows for the Maximum-Expectation algorithm, reduce to decrease running time at the expense of robustness.
`getparam`	If TRUE, generates and returns parameters without running segmentation.
`verbose`	Set to FALSE if messages are not desired

Details

`HMMsegment` is a two stage algorithm that first runs an Expectation-Maximization algorithm to find the optimal set of parameters based on suggested parameter inputs, and allowed flexibilities. After iteratively finding the optimal parameters, the actual segmentation of the data is conducted with the Viterbi algorithm, finally output segmented states. This is an extension to the hidden Markov model described in Shah et al., 2006.

Parameters are divided into two main categories:

- Initial parameters: e, mu, lambda, nu, kappa
- Flexibility parameters: strength, m, eta, gamma, S

Where *initial parameters* are treated as starting suggestions for the parameter optimization algorithm, and flexibility parameters (hyperparameters) define how much the initial parameters are allowed to deviate during the search for the optimal parameters.

With a good copy number dataset, in theory, given enough flexibility, the algorithm should always find a similar set of optimal parameters regardless of initial parameters (although running times will vary).

If for some reason you wish to *manually* set the parameters for the final segmentation process, one should tune all flexibility parameters to minimal values. For example, if you wish to increase the length of segments, you could set:

```
param$e <- 0.9999999999999999
param$strength <- 1e30
```

Which suggests that segments should be very long, and gives minimal to non-existant flexibility to your suggestion.

See vignette for diagrammed example:

```
vignette("HMMcopy")
```

Value

A list object containing multiple values, although in practice only the state assigned to each copy number value in 'states' and the segments of non-overlapping states in 'segs' are of interest.

By default, there are 6 states, which in a diploid sample corresponds to the following chromosomal copies and biological state:

1 <=0 copies, homozogous deletion

2 1 copy, heterozogous deletion

3 2 copies, neutral

4 3 copies, gain

5 4 copies, amplification

6 >=5 copies, high level amplification

The full list of output is as follows:

states The state assigned to each copy number value

segs Non-overlapping segments and medians of each segment

mus Optimal median of of copy numbers in state

lambda Optimal precision of copy numbers in state

pi Optimal state distribution

loglik The likelihood values of each EM algorithm iteration

rho Posterior marginals (responsibilities) for each position and state

Author(s)

Daniel Lai, Gavin Ha, Sohrab Shah

References

Sohrab P Shah, Xiang Xuan, Ron J DeLeeuw, Mehrnoush Khojasteh, Wan L Lam, Raymond Ng, and Kevin P Murphy. Integrating copy number polymorphisms into array cgh analysis using a robust hmm. Bioinformatics, **22(14)**:e431-9, Jul 2006.

See Also

`correctReadcount`, to correct the readcounts prior to segmentation and classification for better results.

Examples

```
data(tumour) # Load tumour_copy
tumour_segments <- HMMsegment(tumour_copy)
```

HMMcopy Visualization *Visualization functions for correctReadcount results*

Description

Convinience functions for creating plots for the analysis of the readcount correction process by `correctReadcount`

Usage

```
plotBias(correctOutput, points = 10000, ...)
plotCorrection(correctOutput, chr = correctOutput$chr[1], ...)
plotSegments(correctOutput, segmentOutput, chr = correctOutput$chr[1], ...)
plotParam(segmentOutput, param, ...)
stateCols()
```

Arguments

`correctOutput`	Output value from `correctReadcount`
`segmentOutput`	Output value from `HMMsegment`
`points`	Number of random sampled points to plot, decreasing reduces runtime
`chr`	Chromosome name to plot. A single value for `plotCorrection` and a vector for `plotSegments`.
`param`	Input parameters to call that produced segmentOutput
`...`	Furthur arguments are passed to `plot`.

Details

plotBias shows the effects of the correction process on GC bias and mappability bias in HTS readcounts.

plotCorrection shows the effects of the correction on the copy number profile. Defaultly plotting the entire first chromosome found in the list.

plotSegments shows the resultant segments and states assigned to each segment.

plotParam shows the initial suggested distribution of copy number in each state (dashed), and the optimal distribution of copy number in each state (solid)

stateCols returns a vector of six colours used in plotSegments and plotParam

Author(s)

Daniel Lai

See Also

correctReadcount and HMMsegment for generating intended output.

Examples

```
data(tumour)

# Visualize one at a time
par(ask = TRUE)
plotBias(normal_copy)
plotCorrection(tumour_copy)
par(mfrow = c(1, 1))
plotSegments(tumour_copy, tumour_segments)
plotParam(tumour_segments, tumour_param)
```

WIG Import Functions *WIG Import Functions*

Description

Fast fixedStep WIG file reading and parsing

Usage

```
wigToRangedData(wigfile, verbose = TRUE)
wigToArray(wigfile, verbose = TRUE)
```

Arguments

wigfile	Filepath to fixedStep WIG format file
verbose	Set to FALSE to suppress messages

Details

Reads the entire file into memory, then processes the file in rapid fashion, thus performance will be limited by memory capacity.

The WIG file is expected to conform to the minimal fixedStep WIG format (see References), where each chromsome is started by a "fixedStep" declaration line. The function assumes only a single track in the WIG file, and will ignore any lines before the first line starting with "fixedStep".

Value

`RangedData` for `wigToRangedData` with chromosome and position information, sorted in decreasing chromosal size and increasing position.

Numeric `array` for `wigToArray` sorted in decreasing chromosal size and increasing position.

Author(s)

Daniel Lai

References

WIG http://genome.ucsc.edu/goldenPath/help/wiggle.html

See Also

`wigsToRangedData` is a wrapper around these functions for easy WIG file loading and structure formatting.

Examples

```
wigfile <- system.file("extdata", "tumour.wig", package = "HMMcopy")
posAndValues <- wigToRangedData(wigfile)
justValues <- wigToArray(wigfile)
```

WIG Output Functions *WIG Output Functions*

Description

Fast fixedStep WIG file formatting and output

Usage

```
rangedDataToWig(correctOutput, file, column = "copy", sample = "R",
  verbose = TRUE)
rangedDataToSeg(correctOutput, file, column = "copy", sample = "R",
  verbose = TRUE)
```

Arguments

correctOutput	RangedData object for output, default options expect output from correctReadcount.
file	Filepath to write output to.
column	Column in input object to export. Defaults to corrected copy number.
sample	Sample name of the exported dataset, defaults to "R"
verbose	Set to FALSE to suppress messages.

Details

Assumes that all ranges in data set are non-overlapping windows of fixed width covering the entire genome. Note that positions in WIG files are 1-based while those in SEG files are 0-based.

Author(s)

Daniel Lai

References

WIG http://genome.ucsc.edu/goldenPath/help/wiggle.html

SEG http://www.broadinstitute.org/igv/SEG

See Also

correctReadcount output is the intended input

Examples

```
data(tumour) # Load tumour_copy
rangedDataToWig(tumour_copy, file = "test.wig")
rangedDataToSeg(tumour_copy, file = "test.seg")
```

---

wigsToRangedData *Parsing and sorting of uncorrected read and sequence information files*

Description

Loads WIG files for readcount, GC, and mappability data for non-overlapping windows of fixed length (i.e. bins), and returns a structure ready to used for readcount correction. See Details for specifics about file assumptions.

Usage

```
wigsToRangedData(readfile, gcfile, mapfile, verbose = FALSE)
```

Arguments

readfile	Pathname to WIG file containing readcounts per bin.
gcfile	Pathname to WIG file containing GC content per bin.
mapfile	Pathname to WIG file containing average mappability per bin.
verbose	Set to TRUE if messages are desired

Details

The number of lines in the three input files are expected to be identical, although the order and names of chromsomes in the file need not be identical. Chromosome lengths are required to be identical and unique, and if the latter is not true, the order of the chromosomes must then be identical.

At present, these three WIG files are expected to be generated by external programs, namely those from the HMMcopy suite (see See Also), rather than by existing R packages out of space and memory considerations when working with high coverage full genome samples.

Value

A RangedData object, where each row entry represents a bin, with the three values from the input as columns named reads, gc, and map.

Author(s)

Daniel Lai

References

correctedReadcount Suite TBA

WIG http://genome.ucsc.edu/goldenPath/help/wiggle.html

See Also

correctReadcount, to correct the readcounts in the resultant value.

Examples

```
rfile <- system.file("extdata", "tumour.wig", package = "HMMcopy")
gfile <- system.file("extdata", "gc.wig", package = "HMMcopy")
mfile <- system.file("extdata", "map.wig", package = "HMMcopy")

uncorrected_reads <- wigsToRangedData(rfile, gfile, mfile)
```

Index

∗ **IO**
  HMMcopy Segmentation, 6
  HMMcopy-package, 2
  WIG Import Functions, 10
  WIG Output Functions, 11
  wigsToRangedData, 12
∗ **datasets**
  HMMcopy example dataset, 5
∗ **hplot**
  HMMcopy Visualization, 9
∗ **internal**
  HMMcopy internal functions, 6
∗ **manip**
  correctReadcount, 3
  HMMcopy-package, 2

array, *11*

correctReadcount, 3, *5*, *7*, *9, 10*, *12, 13*

HMMcopy *(*HMMcopy-package*)*, 2
HMMcopy example dataset, 5
HMMcopy internal functions, 6
HMMcopy Segmentation, 6
HMMcopy Visualization, 9
HMMcopy-dataset *(*HMMcopy example dataset*)*, 5
HMMcopy-internal *(*HMMcopy internal functions*)*, 6
HMMcopy-package, 2
HMMsegment, *5*, *7*, *9, 10*
HMMsegment *(*HMMcopy Segmentation*)*, 6

normal_copy *(*HMMcopy example dataset*)*, 5
normalize *(*HMMcopy internal functions*)*, 6

plot, *9*
plotBias, *10*
plotBias *(*HMMcopy Visualization*)*, 9
plotCorrection, *9, 10*
plotCorrection *(*HMMcopy Visualization*)*, 9
plotParam, *10*
plotParam *(*HMMcopy Visualization*)*, 9
plotSegments, *9, 10*
plotSegments *(*HMMcopy Visualization*)*, 9

RangedData, *3–5*, *11–13*
rangedDataToSeg *(*WIG Output Functions*)*, 11
rangedDataToWig *(*WIG Output Functions*)*, 11

stateCols, *10*
stateCols *(*HMMcopy Visualization*)*, 9

tumour *(*HMMcopy example dataset*)*, 5
tumour_copy *(*HMMcopy example dataset*)*, 5
tumour_param *(*HMMcopy example dataset*)*, 5
tumour_reads *(*HMMcopy example dataset*)*, 5
tumour_segments *(*HMMcopy example dataset*)*, 5

WIG Import Functions, 10
WIG Output Functions, 11
wigsToRangedData, *3–5*, *11*, 12
wigToArray, *11*
wigToArray *(*WIG Import Functions*)*, 10
wigToRangedData, *11*
wigToRangedData *(*WIG Import Functions*)*, 10