Package 'TitanCNA'

April 10, 2015

Type Package

Title Subclonal copy number and LOH prediction from whole genome sequencing of tumours

Version 1.4.0

Date 2014-08-07

Author Gavin Ha, Sohrab P Shah

Maintainer Gavin Ha <`gavinha@gmail.com`>, Sohrab P Shah <`sshah@bccrc.ca`>

Depends R (>= 3.1.0), foreach (>= 1.4.0), IRanges (>= 1.99.1), Rsamtools (>= 1.17.11), GenomeInfoDb (>= 1.1.3)

Description Hidden Markov model to segment and predict regions of subclonal copy number alterations (CNA) and loss of heterozygosity (LOH), and estimate cellular prevalenece of clonal clusters in tumour whole genome sequencing data.

License file LICENSE

biocViews Sequencing, WholeGenome, DNASeq, ExomeSeq, StatisticalMethod, CopyNumberVariation, HiddenMarkovModel, Genetics, GenomicVariation

R topics documented:

TitanCNA-package *TITAN: Subclonal copy number and LOH prediction whole genome sequencing of tumours*

Description

TITAN is a software tool for inferring subclonal copy number alterations (CNA) and loss of heterozygosity (LOH). The algorithm also infers clonal group cluster membership for each event and the tumour proportion, or cellular prevalence, for each event.

Details

Package:	TitanCNA
Type:	Package
Version:	0.99.0
Date:	2014-03-12
License:	see LICENSE

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

`vignette("TitanCNA")` for detailed example

Author(s)

Gavin Ha, Sohrab P Shah Maintainer: Gavin Ha <gavinha@gmail.com>

References

Ha, G., Roth, A., Khattra, J., Ho, J., Yap, D., Prentice, L. M., Melnyk, N., McPherson, A., Bashashati, A., Laks, E., Biele, J., Ding, J., Le, A., Rosner, J., Shumansky, K., Marra, M. A., Huntsman, D. G., McAlpine, J. N., Aparicio, S. A. J. R., and Shah, S. P. (2014). TITAN: Inference of copy number architectures in clonal cell populations from tumour whole genome sequence data. Genome Research, Published in advance July 24 2014. (PMID: 25060187)

Examples

```
message(Running TITAN ...)
#### LOAD DATA ####
infile <- system.file("extdata", "test_alleleCounts_chr2.txt", package = "TitanCNA")
data <- loadAlleleCounts(infile)

#### LOAD PARAMETERS ####
message(titan: Loading default parameters)
numClusters <- 2
params <- loadDefaultParameters(copyNumber = 5,
                                numberClonalClusters = numClusters, skew = 0.1)
```

```
#### READ COPY NUMBER FROM HMMCOPY FILE ####
message(titan: Correcting GC content and mappability biases...)
tumWig <- system.file("extdata", "test_tum_chr2.wig", package = "TitanCNA")
normWig <- system.file("extdata", "test_norm_chr2.wig", package = "TitanCNA")
gc <- system.file("extdata", "gc_chr2.wig", package = "TitanCNA")
map <- system.file("extdata", "map_chr2.wig", package = "TitanCNA")
cnData <- correctReadDepth(tumWig, normWig, gc, map)
logR <- getPositionOverlap(data$chr, data$posn, cnData)
data$logR <- log(2^logR) #transform to natural log

#### FILTER DATA FOR DEPTH, MAPPABILITY, NA, etc ####
data <- filterData(data, c(1:22,"X"), minDepth = 10, maxDepth = 200, map = NULL)

#### EM (FWD-BACK) TO TRAIN PARAMETERS ####
#### Can use parallelization packages ####
K <- length(params$genotypeParams$alphaKHyper)
params$genotypeParams$alphaKHyper <- rep(500, K)
params$ploidyParams$phi_0 <- 1.5
convergeParams <- runEMclonalCN(data, gParams = params$genotypeParams,
                                nParams = params$normalParams,
                                pParams = params$ploidyParams,
                                sParams = params$cellPrevParams,
                                maxiter = 3, maxiterUpdate = 500,
                                txnExpLen = 1e9, txnZstrength = 1e9,
                                useOutlierState = FALSE,
                                normalEstimateMethod = "map",
                                estimateS = TRUE, estimatePloidy = TRUE)
#### COMPUTE OPTIMAL STATE PATH USING VITERBI ####
optimalPath <- viterbiClonalCN(data, convergeParams)

#### FORMAT RESULTS ####
results <- outputTitanResults(data, convergeParams, optimalPath,
                              filename = NULL, posteriorProbs = FALSE,
                              subcloneProfiles = TRUE)

#### PLOT RESULTS ####
norm <- tail(convergeParams$n, 1)
ploidy <- tail(convergeParams$phi, 1)

par(mfrow=c(4, 1))
plotCNlogRByChr(results, chr = 2, ploidy = ploidy, geneAnnot = NULL,
                ylim = c(-2, 2), cex = 0.5, xlab = "", main = "Chr 2")
plotAllelicRatio(results, chr = 2, geneAnnot = NULL, ylim = c(0, 1), cex = 0.5,
                xlab = "", main = "Chr 2")
plotClonalFrequency(results, chr = 2, normal = norm, geneAnnot = NULL,
                    ylim = c(0, 1), cex = 0.5, xlab = "", main = "Chr 2")
plotSubcloneProfiles(results, chr = 2, cex = 2, main = "Chr 2")
```

computeSDbwIndex *Compute the S_Dbw Validity Index for* **TitanCNA** *model selection*

Description

Compute the S_Dbw Validity Index internal cluster validation from the **TitanCNA** results to use for model selection.

Usage

```
computeSDbwIndex(x, centroid.method = "median",
 data.type = "LogRatio", S_Dbw.method = "Halkidi",
 symmetric = TRUE)
```

Arguments

`x`	Formatted **TitanCNA** results output from `outputTitanResults`. See Example.
`centroid.method`	`median` or `mean` method to compute cluster centroids during internal cluster validation.
`data.type`	Compute S_Dbw validity index based on copy number (use '`LogRatio`') or allelic ratio (use '`AllelicRatio`').
`symmetric`	`TRUE` if the TITAN analysis was carried out using symmetric genotypes. See `loadAlleleCounts`.
`S_Dbw.method`	Compute S_Dbw validity index using `Halkidi` or `Tong` method. See details and references.

Details

S_Dbw Validity Index is an internal clustering evaluation that is used for model selection (Halkidi et al. 2002). It attempts to choose the model that minimizes within cluster variances (scat) and maximizes density-based cluster separation (Dens). Then, S_Dbw(|c_T|x z)=Dens(|c_T|x z)+scat(|c_T|x z).

In the context of **TitanCNA**, if `data.type=`'`LogRatio`', then the S_Dbw internal data consists of copy number log ratios, and the resulting joint states of copy number (c_T, forall c_T in {0 : max.copy.number}) and clonal cluster (z) make up the clusters in the internal evaluation. If `data.type=`'`AllelicRatio`', then the S_Dbw internal data consists of the allelic ratios. The optimal **TitanCNA** run is chosen as the run with the minimum S_Dbw.

Note that for `S_Dbw.method`, the `Tong` method has an incorrect formulation of the `scat(c)` function. The function should be a weighted sum, but that is not the formulation shown in the publication. `computeSDbwIndex` uses `(ni/N)` instead of `(N-ni)/N` in the `scat` formula, where `ni` is the number of datapoints in cluster `i` and `N` is the total number of datapoints.

Value

`list` with components:

`dens.bw`	density component of S_Dbw index
`scat`	scatter component of S_Dbw index
`S_DbwIndex`	Sum of dens.bw and scat.

Author(s)

Gavin Ha <gavinha@gmail.com>

References

Halkidi, M., Batistakis, Y., and Vazirgiannis, M. (2002). Clustering validity checking methods: part ii. SIGMOD Rec., 31(3):19–27.

Tong, J. and Tan, H. Clustering validity based on the improved S_Dbw* index. (2009). Journal of Electronics (China), Volume 26, Issue 2, pp 258-264.

Ha, G., Roth, A., Khattra, J., Ho, J., Yap, D., Prentice, L. M., Melnyk, N., McPherson, A., Bashashati, A., Laks, E., Biele, J., Ding, J., Le, A., Rosner, J., Shumansky, K., Marra, M. A., Huntsman, D. G., McAlpine, J. N., Aparicio, S. A. J. R., and Shah, S. P. (2014). TITAN: Inference of copy number architectures in clonal cell populations from tumour whole genome sequence data. Genome Research, Published in advance July 24 2014. (PMID: 25060187)

See Also

`outputModelParameters`, `loadAlleleCounts`

Examples

```
data(EMresults)

#### COMPUTE OPTIMAL STATE PATH USING VITERBI ####
#options(cores=1)
optimalPath <- viterbiClonalCN(data, convergeParams)

#### FORMAT RESULTS ####
results <- outputTitanResults(data, convergeParams, optimalPath,
                              filename = NULL, posteriorProbs = FALSE)

#### COMPUTE S_Dbw Validity Index FOR MODEL SELECTION ####
s_dbw <- computeSDbwIndex(results, data.type = "LogRatio",
centroid.method = "median", S_Dbw.method = "Tong")
```

`correctReadDepth` *Correct GC content and mappability biases in sequencing data read counts*

Description

Correct GC content and mappability biases in tumour sequence read counts using Loess curve fitting. Wrapper for function in **HMMcopy**.

Usage

```
correctReadDepth(tumWig, normWig, gcWig, mapWig, genomeStyle = "NCBI", targetedSequence = NULL)
```

Arguments

`tumWig`	File path to fixedStep WIG format file for the tumour sample. See `wigToRangedData` in the **HMMcopy** for more details.
`normWig`	File path to fixedStep WIG format file for the normal sample.
`gcWig`	File path to fixedStep WIG format file for the GC content based on the specific reference genome sequence used.
`mapWig`	File path to fixedStep WIG format file for the mappability scores computed on the specific reference genome used.
`genomeStyle`	The genome style to use for chromosomes by **TitanCNA**. Use one of 'NCBI' or 'UCSC'. It does not matter what style is found in `inCounts`, `genomeStyle` will be the style returned.
`targetedSequence`	data.frame with 3 columns: chr, start position, stop position. Use this argument for exome capture sequencing or targeted deep sequencing data. This is experimental and may not work as desired.

Details

Wrapper for `correctReadcount` in **HMMcopy** package. It uses a sampling of 50000 bins to find the Loess fit. Then, the log ratio for every bin is returned as the log base 2 of the ratio between the corrected tumour read count and the corrected normal read count.

Value

`data.frame` containing columns:

`chr`	Chromosome; uses 'X' and 'Y' for sex chromosomes
`start`	Start genomic coordinate for bin in which read count is corrected
`end`	End genomic coordinate for bin in which read count is corrected
`logR`	Log ratio, log2(tumour:normal), for bin in which read count is corrected

Author(s)

Gavin Ha <gavinha@gmail.com>, Daniel Lai <jujubix@cs.ubc.ca>

References

Ha, G., Roth, A., Lai, D., Bashashati, A., Ding, J., Goya, R., Giuliany, R., Rosner, J., Oloumi, A., Shumansky, K., Chin, S.F., Turashvili, G., Hirst, M., Caldas, C., Marra, M. A., Aparicio, S., and Shah, S. P. (2012). Integrative analysis of genome wide loss of heterozygosity and monoallelic expression at nucleotide resolution reveals disrupted pathways in triple negative breast cancer. Genome Research, 22(10):1995,2007. (PMID: 22637570)

Ha, G., Roth, A., Khattra, J., Ho, J., Yap, D., Prentice, L. M., Melnyk, N., McPherson, A., Bashashati, A., Laks, E., Biele, J., Ding, J., Le, A., Rosner, J., Shumansky, K., Marra, M. A., Huntsman, D. G., McAlpine, J. N., Aparicio, S. A. J. R., and Shah, S. P. (2014). TITAN: Inference of copy number architectures in clonal cell populations from tumour whole genome sequence data. Genome Research, Published in advance July 24 2014. (PMID: 25060187)

See Also

correctReadcount and wigToRangedData in the **HMMcopy** package. WIG: http://genome.ucsc.edu/goldenPath/help/wiggle.html

Examples

```
tumWig <- system.file("extdata", "test_tum_chr2.wig", package = "TitanCNA")
normWig <- system.file("extdata", "test_norm_chr2.wig", package = "TitanCNA")
gc <- system.file("extdata", "gc_chr2.wig", package = "TitanCNA")
map <- system.file("extdata", "map_chr2.wig", package = "TitanCNA")

#### GC AND MAPPABILITY CORRECTION ####
cnData <- correctReadDepth(tumWig, normWig, gc, map)
```

---

extractAlleleReadCounts

Function to extract allele read counts from a sequence alignment (BAM) file

---

Description

Function to extract allele read counts from a sequence alignment (BAM) file at specific positions of interest. The positions are passed in as the file path to a file in variant call format (VCF).

Usage

```
extractAlleleReadCounts(bamFile, bamIndex, positions,
 outputFilename = NULL, pileupParam = PileupParam())
```

Arguments

`bamFile`	File path location to the sequencing alignment file (BAM format) from which to extract read counts.
`bamIndex`	File path location to the BAM index file (usually with extension .bai) corresponding to the sequencing alignment file `bamFile`.
`positions`	File path location to the variant call format (VCF) file containing the positions at which read counts are to be extracted.
`outputFilename`	If given, will specify the file path to which the result will be output as tab-delimited text. Otherwise, the no output is written to file.
`pileupParam`	`PileupParam` object from the **Rsamtools**. See Details.

Details

The `pileupParam` object allows users to specify the sequencing parameters to consider when generating the pileup from which read counts are extracted. This includes 'max_depth', 'min_base_quality', 'min_mapq', 'min_nucleotide_depth'=10 (recommended), 'min_minor_allele_depth', 'distinguish_strands', 'distinguish_nucleotides'=TRUE (must be TRUE).

Value

data.frame containing columns:

chr	Chromosome; character
position	Position; numeric
ref	Reference counts; character
refCount	Reference counts; numeric
Nref	Non-reference counts; character
NrefCount	Non-reference counts; numeric

Author(s)

Gavin Ha <gavinha@gmail.com>

References

Ha, G., Roth, A., Khattra, J., Ho, J., Yap, D., Prentice, L. M., Melnyk, N., McPherson, A., Bashashati, A., Laks, E., Biele, J., Ding, J., Le, A., Rosner, J., Shumansky, K., Marra, M. A., Huntsman, D. G., McAlpine, J. N., Aparicio, S. A. J. R., and Shah, S. P. (2014). TITAN: Inference of copy number architectures in clonal cell populations from tumour whole genome sequence data. Genome Research, Published in advance July 24 2014. (PMID: 25060187)

See Also

PileupParam; http://samtools.sourceforge.net/

Examples

```
  ## Not run:
countsDF <- extractAlleleReadCounts(bamFile, bamIndex,
positions, outputFilename = NULL,
pileupParam = PileupParam())
data <- loadAlleleCounts(countsDF, symmetric = TRUE,
   genomeStyle = "NCBI")

## End(Not run)
```

filterData *Filter list object based on read depth and missing data*

Description

Filters all vectors in list based on specified chromosome(s) of interest, minimum and maximum read depths, missing data, mappability score threshold

Usage

```
filterData(data ,chrs = NULL, minDepth = 10, maxDepth = 200,
    positionList = NULL, map = NULL, mapThres = 0.9)
```

Arguments

`data`	`list` object that contains an arbitrary number of components. Should include 'chr', 'tumDepth'. All vector elements must have the same number of rows where each row corresponds to information pertaining to a chromosomal position.
`chrs`	`character` or vector of `character` specifying the chromosomes to keep. Chromosomes not included in this `array` will be filtered out. Chromosome style must match the `genomeStyle` used when running `loadAlleleCounts`
`minDepth`	`Numeric integer` specifying the minimum tumour read depth to include. Positions >= `minDepth` are kept.
`maxDepth`	`Numeric integer` specifying the maximum tumour read depth to include. Positions <= `maxDepth` are kept.
`positionList`	`data.frame` with two columns: 'chr' and 'posn'. `positionList` lists the chromosomal positions to use in the analysis. All positions not overlapping this list will be excluded. Use `NULL` to use all current positions in `data`.
`map`	`Numeric array` containing map scores corresponding to each position in `data`. Optional for filtering positions based on mappability scores.
`mapThres`	`Numeric float` specifying the mappability score threshold. Only applies if `map` is specified. `map` scores >= `mapThres` are kept.

Details

All vectors in the input `data` list object, and `map`, must all have the same number of rows.

Value

The same `list data` containing filtered components.

Author(s)

Gavin Ha <gavinha@gmail.com>

References

Ha, G., Roth, A., Khattra, J., Ho, J., Yap, D., Prentice, L. M., Melnyk, N., McPherson, A., Bashashati, A., Laks, E., Biele, J., Ding, J., Le, A., Rosner, J., Shumansky, K., Marra, M. A., Huntsman, D. G., McAlpine, J. N., Aparicio, S. A. J. R., and Shah, S. P. (2014). TITAN: Inference of copy number architectures in clonal cell populations from tumour whole genome sequence data. Genome Research, Published in advance July 24 2014. (PMID: 25060187)

See Also

`loadAlleleCounts`

Examples

```
infile <- system.file("extdata", "test_alleleCounts_chr2.txt",
                      package = "TitanCNA")
tumWig <- system.file("extdata", "test_tum_chr2.wig", package = "TitanCNA")
normWig <- system.file("extdata", "test_norm_chr2.wig", package = "TitanCNA")
gc <- system.file("extdata", "gc_chr2.wig", package = "TitanCNA")
map <- system.file("extdata", "map_chr2.wig", package = "TitanCNA")

#### LOAD DATA ####
data <-  loadAlleleCounts(infile, genomeStyle = "NCBI")

#### GC AND MAPPABILITY CORRECTION ####
cnData <- correctReadDepth(tumWig, normWig, gc, map)


#### READ COPY NUMBER FROM HMMCOPY FILE ####
logR <- getPositionOverlap(data$chr, data$posn, cnData)
data$logR <- log(2^logR) #use natural logs

#### FILTER DATA FOR DEPTH, MAPPABILITY, NA, etc ####
filtereData <- filterData(data, as.character(1:24), minDepth = 10,
maxDepth = 200, map = NULL, mapThres=0.9)
```

`Formatting and output of Titan results`

Formatting and printing **TitanCNA** *results.*

Description

Function to format **TitanCNA** results in to a data.frame and output the results to a tab-delimited file.

Usage

```
  outputTitanResults(data, convergeParams, optimalPath, filename = NULL,
      posteriorProbs = FALSE, subcloneProfiles = TRUE)
  outputModelParameters(convergeParams, results, filename,
   S_Dbw.data.type = "LogRatio", S_Dbw.scale = 1, S_Dbw.method = "Tong")
```

Arguments

`data`	`list` object that contains the components for the data to be analyzed. `chr`, `posn`, `ref`, and `tumDepth` that can be obtained using `loadAlleleCounts`, and `logR` that can be obtained using `correctReadDepth` and `getPositionOverlap` (see Example).
`convergeParams`	`list` object that is returned from the function `runEMclonalCN` in **TitanCNA**.

`optimalPath`	`numeric` `array` containing the optimal **TitanCNA** genotype and clonal cluster states for each data point in the analysis. `optimalPath` is obtained from running `viterbiClonalCN`.
`results`	Formatted **TitanCNA** results output from `outputTitanResults`.
`filename`	Path of the file to write the **TitanCNA** results.
`posteriorProbs`	`Logical` `TRUE` to include the posterior marginal probabilities in printing to `filename`.
`subcloneProfiles`	`Logical` `TRUE` to include the subclone profiles to the output `data.frame`. Currently, this only works for 1 or 2 clonal clusters.
`S_Dbw.data.type`	Compute S_Dbw validity index based on copy number (use '`LogRatio`') or allelic ratio (use '`AllelicRatio`'). See `computeSDbwIndex`.
`S_Dbw.scale`	The S_Dbw validity index can be adjusted to account for differences between datasets. `SDbw.scale` can be used to penalize the S_Dbw `dens.bw` component. The default is 1.
`S_Dbw.method`	Compute S_Dbw validity index using `Halkidi` or `Tong` method. See `computeSDbwIndex`.

Details

`outputModelParameters` outputs to a file with the estimated TITAN model parameters and model selection index. Each row contains information regarding different parameters:

1) Normal contamination estimate - proportion of normal content in the sample; tumour content is 1 minus this number

2) Average tumour ploidy estimate - average number of estimated copies in the genome; 2 represents diploid

3) Clonal cluster cellular prevalence - Z denotes the number of clonal clusters; each value (space-delimited) following are the cellular prevalence estimates for each cluster. Cellular prevalence here is defined as the proportion of tumour sample that does contain the aberrant genotype.

4) Genotype binomial means for clonal cluster Z - set of 21 binomial estimated parameters for each specified cluster

5) Genotype Gaussian means for clonal cluster Z - set of 21 Gaussian estimated means for each specified cluster

6) Genotype Gaussian variance - set of 21 Gaussian estimated variances; variances are shared for across all clusters

7) Number of iterations - number of EM iterations needed for convergence

8) Log likelihood - complete data log-likelihood for current cluster run

9) S_Dbw dens.bw - density component of S_Dbw index; see `computeSDbwIndex`

10) S_Dbw scat - scatter component of S_Dbw index; see `computeSDbwIndex`

11) S_Dbw validity index - used for model selection where the run with optimal number of clusters based on lowest S_Dbw index. This value is slightly modified from that computed from `computeSDbwIndex`. It is computed as S_Dbw=25 x dens.bw+scat.

`outputTitanResults` outputs a file that has the similar format described in 'Value' section.

Value

`outputTitanResults` also returns a `data.frame`, where each row corresponds to a position in the analysis, and with the following columns:

`Chr`	character denoting chromosome number. ChrX and ChrY uses 'X' and 'Y'.
`Position`	genomic coordinate
`RefCount`	number of reads matching the reference base
`NRefCount`	number of reads matching the non-reference base
`Depth`	total read depth at the position
`AllelicRatio`	RefCount/Depth
`LogRatio`	log2 ratio between normalized tumour and normal read depths
`CopyNumber`	predicted **TitanCNA** copy number
`TITANstate`	internal state number used by **TitanCNA**; see Reference
`TITANcall`	interpretable **TitanCNA** state; string (HOMD,DLOH,HET,NLOH,ALOH,ASCNA,BCNA,UBCNA); See Reference
`ClonalCluster`	predicted **TitanCNA** clonal cluster; lower cluster numbers represent clusters with higher cellular prevalence
`CellularPrevalence`	proportion of tumour cells containing event; not to be mistaken as proportion of sample (including normal)

If `subcloneProfiles` is set to `TRUE`, then the subclone profiles will be appended to the output `data.frame`.

`Subclone1.CopyNumber`	Integer copy number for Subclone 1.
`Subclone1.TITANcall`	States for Subclone 1
`Subclone1.Prevalence`	The cellular prevalence of Subclone 1, or sometimes referred to as the subclone fraction.

Author(s)

Gavin Ha <gavinha@gmail.com>

References

Ha, G., Roth, A., Khattra, J., Ho, J., Yap, D., Prentice, L. M., Melnyk, N., McPherson, A., Bashashati, A., Laks, E., Biele, J., Ding, J., Le, A., Rosner, J., Shumansky, K., Marra, M. A., Huntsman, D. G., McAlpine, J. N., Aparicio, S. A. J. R., and Shah, S. P. (2014). TITAN: Inference of copy number architectures in clonal cell populations from tumour whole genome sequence data. Genome Research, Published in advance July 24 2014. (PMID: 25060187)

See Also

`runEMclonalCN`, `viterbiClonalCN`, `computeSDbwIndex`

Examples

```
data(EMresults)

#### COMPUTE OPTIMAL STATE PATH USING VITERBI ####
optimalPath <- viterbiClonalCN(data, convergeParams)

#### FORMAT RESULTS ####
results <- outputTitanResults(data, convergeParams, optimalPath,
                              filename = NULL, posteriorProbs = FALSE,
                              subcloneProfiles = TRUE)

#### OUTPUT RESULTS TO FILE ####
outparam <- paste("cluster2_params.txt", sep = "")
outputModelParameters(convergeParams, results, outparam)
```

---

getPositionOverlap — *Function to assign values to given chromosome-position that overlaps a list of chromosomal segments*

Description

Given a list of chromosomes and positions, uses a C-based function that searches a list of segments to find the overlapping segment. Then, takes the value (4th column in segment data.frame) of the overlapping segment and assigns to the given chromosome and position.

Usage

```
getPositionOverlap(chr, posn, cnData)
```

Arguments

chr	`Numeric` `array` denoting the chromosome for a list of positions. Must have the same number of elements as `posn`.
posn	`Numeric` `array` denoting the position in the chromosome for a list of positions. Must have the same number of elements as `chr`.
cnData	`data.frame` containing a list of segments described with 4 columns: chromosome, start coordinate, end coordinate, value of interest (e.g. log ratio). Chromosome can be all numeric or chrX and chrY can use 'X' and 'Y'.

Value

`Numeric` `array` of values from the 4th column of `data.frame` `cnData`. Each element corresponds to a genomic location from `chr` and `posn` that overlapped the segment in `cnData`.

Author(s)

Gavin Ha <gavinha@gmail.com>

References

Ha, G., Roth, A., Khattra, J., Ho, J., Yap, D., Prentice, L. M., Melnyk, N., McPherson, A., Bashashati, A., Laks, E., Biele, J., Ding, J., Le, A., Rosner, J., Shumansky, K., Marra, M. A., Huntsman, D. G., McAlpine, J. N., Aparicio, S. A. J. R., and Shah, S. P. (2014). TITAN: Inference of copy number architectures in clonal cell populations from tumour whole genome sequence data. Genome Research, Published in advance July 24 2014. (PMID: 25060187)

See Also

`loadAlleleCounts`, `correctReadDepth`

Examples

```
infile <- system.file("extdata", "test_alleleCounts_chr2.txt",
                      package = "TitanCNA")
tumWig <- system.file("extdata", "test_tum_chr2.wig", package = "TitanCNA")
normWig <- system.file("extdata", "test_norm_chr2.wig", package = "TitanCNA")
gc <- system.file("extdata", "gc_chr2.wig", package = "TitanCNA")
map <- system.file("extdata", "map_chr2.wig", package = "TitanCNA")

#### LOAD DATA ####
data <- loadAlleleCounts(infile)

#### GC AND MAPPABILITY CORRECTION ####
cnData <- correctReadDepth(tumWig, normWig, gc, map)

#### READ COPY NUMBER FROM HMMCOPY FILE ####
logR <- getPositionOverlap(data$chr, data$posn, cnData)
```

`loadAlleleCounts` *Function to load tumour allele counts from a text file or data.frame*

Description

Function to load in the allele counts from tumour sequencing data from a delimited text file or data.frame object.

Usage

```
loadAlleleCounts(inCounts, symmetric = TRUE, genomeStyle = "NCBI", sep = "\t")
```

Arguments

`inCounts`	Full file path to text file or data.frame containing tumour allele count data. `inCounts` must be 6 columns: chromosome, position, reference base, reference read counts, non-reference base, non-reference read counts. ‘chromosome’ column can be in ‘NCBI’ or ‘UCSC’ genome style; only autosomes, sex chromosomes, and mitochondrial chromosome are included (e.g. 1-22,X,Y,MT). The

	reference and non-reference base columns can be any arbitrary character; it is not used by **TitanCNA**.
`symmetric`	`logical`; if `TRUE`, then the symmetric allelic counts will be used. `ref` will equal `max(ref,nonRef)`. This parameter must be the same as the `symmetric` parameter for `loadDefaultParameters`.
`genomeStyle`	The genome style to use for chromosomes by **TitanCNA**. Use one of 'NCBI' or 'UCSC'. It does not matter what style is found in `inCounts`, `genomeStyle` will be the style returned.
`sep`	Character indicating the delimiter used for the columns for `infile`. Default is tab-delimited, "\t".

Value

`list` containing components for

`chr`	Chromosome; character, NCBI or UCSC genome style format
`posn`	Position; integer
`ref`	Reference counts; numeric
`nonRef`	Non-reference counts; numeric
`tumDepth`	Tumour depth; numeric

Author(s)

Gavin Ha <gavinha@gmail.com>

References

Ha, G., Roth, A., Khattra, J., Ho, J., Yap, D., Prentice, L. M., Melnyk, N., McPherson, A., Bashashati, A., Laks, E., Biele, J., Ding, J., Le, A., Rosner, J., Shumansky, K., Marra, M. A., Huntsman, D. G., McAlpine, J. N., Aparicio, S. A. J. R., and Shah, S. P. (2014). TITAN: Inference of copy number architectures in clonal cell populations from tumour whole genome sequence data. Genome Research, Published in advance July 24 2014. (PMID: 25060187)

See Also

`loadDefaultParameters`

Examples

```
  infile <- system.file("extdata", "test_alleleCounts_chr2.txt",
                        package = "TitanCNA")
  #### LOAD DATA FROM TEXT FILE ####
  data <- loadAlleleCounts(infile, symmetric = TRUE,
   genomeStyle = "NCBI")

  ## Not run:
countsDF <- extractAlleleReadCounts(tumBam, tumBamIndex,
germlineHetPosns, outputFilename = NULL,
pileupParam = PileupParam())
```

```
data <- loadAlleleCounts(countsDF, symmetric = TRUE,
   genomeStyle = "NCBI")

## End(Not run)
```

loadDefaultParameters *Load TITAN parameters*

Description

Load TITAN model parameters based on maximum copy number and number of clonal clusters.

Usage

```
loadDefaultParameters(copyNumber = 5, numberClonalClusters = 1, skew = 0,
    symmetric = TRUE)
```

Arguments

copyNumber	Maximum number of absolute copies to account for in the model. Default (and recommended) is 5.
numberClonalClusters	Number of clonal clusters to use. Using ‘1’ represents more tumour subclonality. ‘2’ or higher treats the tumour data as being subclonal.
skew	numeric float indicating the heterozygous baseline shift for the allelic ratios towards 1. This is may be required for SOLiD data, but for most cases, this argument can be omitted.
symmetric	logical; if TRUE, then treat genotypes as symmetric. See Details.

Details

Generally, **TitanCNA** should be run once for each number of clonal clusters in the range of 1 to 5. Then, use model selection to choose the run with the optimal number of clusters.

If the allelic ratio data is skewed towards one allele, then use skew to help define the baseline. For example, if the data is skewed towards the reference, then use 0.1 so that the heterozygous baseline is at 0.6. The allelic ratio baseline is normally at 0.5.

sParams, which represents the parameters for estimation of subclonality, always contains values for one cluster that represents the clonally dominant cluster (events present in nearly all tumour cells) with an initial value of sParams$s_0[1] = 0.001.

Setting symmetric to TRUE will treat reference and non-reference alleles the same. For example, genotypes AA (homozygous for reference allele) and BB (homozygous for non-reference allele) as being equivalent. This will reduce the state space substantially.

Value

list containing 4 sets of parameters, each as a component:

genotypeParams	Parameters for copy number and allelic ratios geneotype states
normalParams	Parameters for normal contamination
ploidyParams	Parameters for average tumour ploidy
sParams	Parameters for modeling subclonality: clonal clusters and cellular prevalence

Author(s)

Gavin Ha <gavinha@gmail.com>

References

Ha, G., Roth, A., Khattra, J., Ho, J., Yap, D., Prentice, L. M., Melnyk, N., McPherson, A., Bashashati, A., Laks, E., Biele, J., Ding, J., Le, A., Rosner, J., Shumansky, K., Marra, M. A., Huntsman, D. G., McAlpine, J. N., Aparicio, S. A. J. R., and Shah, S. P. (2014). TITAN: Inference of copy number architectures in clonal cell populations from tumour whole genome sequence data. Genome Research, Published in advance July 24 2014. (PMID: 25060187)

See Also

loadAlleleCounts

Examples

```
#### LOAD PARAMETERS ####
numClusters <- 2
params <- loadDefaultParameters(copyNumber = 5,
                                numberClonalClusters = numClusters)
```

`Plotting TITAN results` *Plotting functions for* **TitanCNA** *results.*

Description

Three plotting functions for **TitanCNA** results. `plotCNlogRByChr` plots the copy number results from log ratio data. `plotAllelicRatio` plots the allelic imbalance and loss of heterozygosity (LOH) from allelic ratio data. `plotClonalFrequency` plots the clonal cluster and cellular prevalence results for each data point.

Usage

```
plotAllelicRatio(dataIn, chr = NULL, geneAnnot = NULL, spacing = 4,
    xlim = NULL, ...)
plotClonalFrequency(dataIn, chr = NULL, normal = NULL, geneAnnot = NULL,
    spacing = 4, xlim = NULL, ...)
plotCNlogRByChr(dataIn, chr = NULL, geneAnnot = NULL, ploidy = NULL,
    spacing = 4, alphaVal = 1, xlim = NULL, ...)
plotSubcloneProfiles(dataIn, chr = NULL, geneAnnot = NULL,
   spacing = 4, xlim = NULL, ...)
```

Arguments

`dataIn`	Formatted **TitanCNA** results output from `outputTitanResults`. See Example.
`chr`	Plot results for specified `chr`. If `chr` is `NULL`, then results for the entire genome is plot.
`geneAnnot`	`data.frame` specifying the genes to annotate in the plot. Gene boundaries are indicated using vertical dotted grey lines and gene symbols are shown at the top of the plot. `geneAnnot` must have four columns: gene symbol, chr, start coordinate, stop coordinate.
`normal`	`numeric` scalar indicating the normal contamination. This can be obtained from converge parameters output using `runEMclonalCN`. See Example.
`ploidy`	`numeric` scalar indicating the tumour ploidy used to adjust the copy number plot `plotCNlogRByChr`. This can be obtained from converge parameters output using `runEMclonalCN`. See Example. If `NULL` is used, then ploidy adjustment is not used in the plot.
`spacing`	Number of lines of spacing for the margin spacing at the bottom of the plot. Useful if an idiogram/karogram is plot underneath.
`alphaVal`	Set an alpha value between 0 and 1 to allow transparency in the points being plot.
`xlim`	Two element vector to specify the xlim for the plot. If `NULL`, then entire chromosome is plot.
`...`	Additional arguments used in the `plot` function.

Details

`plotCNlogRByChr` plots the copy number alterations from log ratio data. The Y-axis is based on log ratios. Log ratios are computed ratios between normalized tumour and normal read depths. Data points close to 0 represent diploid, above 0 are copy gains, below 0 are deletions. `ploidy` argument adjusts the baseline of the data points. Colours represent the copy number state. Bright Green - Homozygous deletion (HOMD) Green - Hemizygous deletion (DLOH) Blue - Diploid heterozygous (HET), Copy-neutral LOH (NLOH) Dark Red - GAIN Red - Allele-specific CNA (ASCNA), Unbalanced CNA (UBCNA), Balanced CNA (BCNA)

`plotAllelicRatio` plots the allelic imbalance and loss of heterozygosity from allelic ratio data. The Y-axis is based on allelic ratios. Allelic ratios are computed as RefCount/Depth. Data points close to 1 represent homozygous reference base, close to 0 represent homozygous non-reference base, and close to 0.5 represent heterozygous. Normal contamination influences the divergence away from 0.5 for LOH events. No adjustments are made to the plot as the original data from `dataIn` are shown. Colours represent the allelic imbalance and LOH state. Grey - HET, BCNA Bright Green - HOMD Green - DLOH, ALOH Blue - NLOH Dark Red - GAIN Red - ASCNA, UBCNA

`plotClonalFrequency` plots the cellular prevalence and clonal clusters from the results. The Y-axis is the cellular prevalence that includes the normal proportion. Therefore, the cellular prevalence here refers to the proportion in the sample (including normal). Lines are drawn for each data point indicating the cellular prevalence. Heterozygous diploid are not shown because it is a normal genotype and is not categorized as being subclonal (this means 100% of cells are normal). The black horizontal line represents the tumour content labeled as 'T'. Each horizontal grey line represents the cellular prevalence of the clonal clusters labeled as Z1, Z2, etc. Colours are the sames for allelic ratio plots.

`plotSubcloneProfiles` plots the predicted copy number profile for individual subclones inferred by TITAN. Currently, this only works for solutions having 1 or 2 clonal clusters. The colours are the same as for `plotAllelicRatio`.

Author(s)

Gavin Ha <gavinha@gmail.com>

References

Ha, G., Roth, A., Khattra, J., Ho, J., Yap, D., Prentice, L. M., Melnyk, N., McPherson, A., Bashashati, A., Laks, E., Biele, J., Ding, J., Le, A., Rosner, J., Shumansky, K., Marra, M. A., Huntsman, D. G., McAlpine, J. N., Aparicio, S. A. J. R., and Shah, S. P. (2014). TITAN: Inference of copy number architectures in clonal cell populations from tumour whole genome sequence data. Genome Research, Published in advance July 24 2014. (PMID: 25060187)

See Also

`outputTitanResults`, `runEMclonalCN`, `computeSDbwIndex`

Examples

```
data(EMresults)

#### COMPUTE OPTIMAL STATE PATH USING VITERBI ####
optimalPath <- viterbiClonalCN(data, convergeParams)

#### FORMAT RESULTS ####
results <- outputTitanResults(data, convergeParams, optimalPath,
                              filename = NULL, posteriorProbs = FALSE)

#### PLOT RESULTS ####
norm <- tail(convergeParams$n, 1)
ploidy <- tail(convergeParams$phi, 1)

par(mfrow=c(4, 1))
plotCNlogRByChr(results, chr = 2, ploidy = ploidy, geneAnnot = NULL,
                ylim = c(-2, 2), cex = 0.5, xlab = "", main = "Chr 2")
```

```
plotAllelicRatio(results, chr = 2, geneAnnot = NULL, ylim = c(0, 1), cex = 0.5,
                xlab = "", main = "Chr 2")
plotClonalFrequency(results, chr = 2, normal = norm, geneAnnot = NULL,
                    ylim = c(0, 1), cex = 0.5, xlab = "", main = "Chr 2")
plotSubcloneProfiles(results, chr = 2, cex = 2, main = "Chr 2")
```

runEMclonalCN — *Function to run the Expectation Maximization Algorithm in* **TitanCNA**.

Description

Function to run the Expectation Maximization Algorithm for inference of model parameters: cellular prevalence, normal proportion, tumour ploidy. This is a key function in the **TitanCNA** package and is the most computationally intense. This function makes calls to a C subroutine that allows the algorithm to be run more efficiently.

Usage

```
runEMclonalCN(data, gParams, nParams, pParams, sParams,
              txnExpLen = 1e15, txnZstrength = 5e05, maxiter = 15,
              maxiterUpdate = 1500, pseudoCounts = 1e-300,
              normalEstimateMethod = "map", estimateS = TRUE,
              estimatePloidy = TRUE, useOutlierState = FALSE, verbose = TRUE)
```

Arguments

data	list object that contains the components for the data to be analyzed. chr, posn, ref, and tumDepth that can be obtained using loadAlleleCounts, and logR that can be obtained using correctReadDepth and getPositionOverlap (see Example).
gParams	list object that contains the copy number and allelic ratio genotype parameters. Can be obtained from loadDefaultParameters.
nParams	list object that contains the normal contamination parameters. Can be obtained from loadDefaultParameters.
pParams	list object that contains the tumour ploidy parameters. Can be obtained from loadDefaultParameters.
sParams	list object that contains the subclonality (cellular prevalence and clonal cluster) parameters. Can be obtained from loadDefaultParameters.
txnExpLen	Influences prior probability of genotype transitions in the HMM. Smaller value have lower tendency to change state; however, too small and it produces underflow problems. 1e-9 works well for up to 3 million total positions.
txnZstrength	Influences prior probability of clonal cluster transitions in the HMM. Smaller value have lower tendency to change clonal cluster state. 1e-9 works well for up to 3 million total positions.

`pseudoCounts`	Small, machine precision values to add to probabilities to avoid underflow. For example, `.Machine$double.eps`.
`maxiter`	Maximum number of expectation-maximization iterations allowed. In practice, for **TitanCNA**, it will usually not exceed 20.
`maxiterUpdate`	Maximum number of coordinate descent iterations during the M-step (of EM algorithm) when parameters are estimated.
`normalEstimateMethod`	Specifies how to handle normal proportion estimation. Using `map` will use the maximum a posteriori estimation. Using `fixed` will not estimate the normal proportion; the normal proportion will be fixed to whatever is specified in `params$normalParams$n_0`. See Details.
`estimateS`	Logical indicating whether to account for clonality and estimate subclonal events. See Details.
`estimatePloidy`	Logical indicating whether to estimate and account for tumour ploidy.
`useOutlierState`	Logical indicating whether an additional outlier state should be used. In practice, this is usually not necessary.
`verbose`	Set to FALSE to suppress program messages.

Details

This function is implemented with the "`foreach`" package and therefore supports parallelization. See "`doMC`" or "`doMPI`" for some parallelization packages.

The forwards-backwards algorithm is used for the E-step in the EM algorithm. This is done using a call to a C subroutine for each chromosome. The maximization step uses maximum a posteriori (MAP) for estimation of parameters.

If the sample has absolutely no normal contamination, then assign `nParams$n_0 <- 0` and use argument `normalEstimateMethod="fixed"`.

`estimateS` should always be set to `TRUE`. If no subclonality is expected, then use `loadDefaultParameters(numberClonalCl`
Using `estimateS=FALSE` and `loadDefaultParameters(numberClonalClusters=0)` is gives more or less the same results.

Value

`list` with components for results returned from the EM algorithm, including converged parameters, posterior marginal responsibilities, log likelihood, and original parameter settings.

`n`	Converged estimate for normal contamination parameter. `numeric array` containing estimates at each EM iteration.
`s`	Converged estimate(s) for cellular prevalence parameter(s). This value is defined as the proportion of tumour sample that does *not* contain the aberrant genotype. This will contrast what is output in `outputTitanResults`. `numeric array` containing estimates at each EM iteration. If more than one cluster is specified, then `s` is a `numeric matrix`.
`var`	Converged estimates for variance parameter of the Gaussian mixtures used to model the log ratio data. `numeric matrix` containing estimates at each EM iteration.

`phi`	Converged estimate for tumour ploidy parameter. `numeric array` containing estimates at each EM iteration.
`piG`	Converged estimate for initial genotype state distribution. `numeric matrix` containing estimates at each EM iteration.
`piZ`	Converged estimate for initial clonal cluster state distribution. `numeric matrix` containing estimates at each EM iteration.
`muR`	Mean of binomial mixtures computed as a function of `s` and `n`. `numeric matrix` containing estimates at each EM iteration. See References for mathematical details.
`muC`	Mean of Gaussian mixtures computed as a function of `s`, `n`, and `phi`. `numeric matrix` containing estimates at each EM iteration. See References for mathematical details.
`loglik`	Posterior Log-likelihood that includes data likelihood and the priors. `numeric array` containing estimates at each EM iteration.
`rhoG`	Posterior marginal probabilities for the genotype states computed during the E-step. Only the final iteration is returned as a `numeric matrix`.
`rhoZ`	Posterior marginal probabilities for the clonal cluster states computed during the E-step. Only the final iteration is returned as a `numeric matrix`.
`genotypeParams`	Original genotype parameters. See `loadDefaultParameters`.
`ploidyParams`	Original tumour ploidy parameters. See `loadDefaultParameters`.
`normalParams`	Original normal contamination parameters. See `loadDefaultParameters`.
`clonalParams`	Original subclonal parameters. See `loadDefaultParameters`.
`txnExpLen`	Original genotype transition expected length. See `loadDefaultParameters`.
`txnZstrength`	Original clonal cluster transition expected length. See `loadDefaultParameters`.
`useOutlierState`	Original setting indicating usage of outlier state. See `loadDefaultParameters`.

Author(s)

Gavin Ha <gavinha@gmail.com>

References

Ha, G., Roth, A., Khattra, J., Ho, J., Yap, D., Prentice, L. M., Melnyk, N., McPherson, A., Bashashati, A., Laks, E., Biele, J., Ding, J., Le, A., Rosner, J., Shumansky, K., Marra, M. A., Huntsman, D. G., McAlpine, J. N., Aparicio, S. A. J. R., and Shah, S. P. (2014). TITAN: Inference of copy number architectures in clonal cell populations from tumour whole genome sequence data. Genome Research, Published in advance July 24 2014. (PMID: 25060187)

See Also

`"foreach"`, `"doMC"`, `"doMPI"`, `loadAlleleCounts`, `loadDefaultParameters`, `viterbiClonalCN`

Examples

```
message(Running TITAN ...)
#### LOAD DATA ####
infile <- system.file("extdata", "test_alleleCounts_chr2.txt",
              package = "TitanCNA")
data <- loadAlleleCounts(infile)

#### LOAD PARAMETERS ####
message(titan: Loading default parameters)
numClusters <- 2
params <- loadDefaultParameters(copyNumber = 5,
              numberClonalClusters = numClusters, skew = 0.1)

#### READ COPY NUMBER FROM HMMCOPY FILE ####
message(titan: Correcting GC content and mappability biases...)
tumWig <- system.file("extdata", "test_tum_chr2.wig", package = "TitanCNA")
normWig <- system.file("extdata", "test_norm_chr2.wig", package = "TitanCNA")
gc <- system.file("extdata", "gc_chr2.wig", package = "TitanCNA")
map <- system.file("extdata", "map_chr2.wig", package = "TitanCNA")
cnData <- correctReadDepth(tumWig, normWig, gc, map)
logR <- getPositionOverlap(data$chr, data$posn, cnData)
data$logR <- log(2^logR) #transform to natural log

#### FILTER DATA FOR DEPTH, MAPPABILITY, NA, etc ####
data <- filterData(data, 1:24, minDepth = 10, maxDepth = 200, map = NULL)

#### EM (FWD-BACK) TO TRAIN PARAMETERS ####
#### Can use parallelization packages ####
K <- length(params$genotypeParams$alphaKHyper)
params$genotypeParams$alphaKHyper <- rep(500, K)
params$ploidyParams$phi_0 <- 1.5
convergeParams <- runEMclonalCN(data, gParams = params$genotypeParams,
                                nParams = params$normalParams,
                                pParams = params$ploidyParams,
                                sParams = params$cellPrevParams,
                                maxiter = 3, maxiterUpdate = 500,
                                txnExpLen = 1e9, txnZstrength = 1e9,
                                useOutlierState = FALSE,
                                normalEstimateMethod = "map",
                                estimateS = TRUE, estimatePloidy = TRUE)
```

`TitanCNA trained dataset`

TITAN EM trained results for an example dataset

Description

Data for chromosome 2 for a triple-negative breast cancer dataset and the expectation-maximization (EM) trained results. Only 20,000 datapoints are included and the data has been scrambled to anonymous patient SNPs.

data Processed input data that is first generated by `loadAlleleCounts`, and includes log ratios that have been GC content and mappability corrected using `correctReadDepth`.

convergeParams EM results that is generated by `runEMclonalCN`

Usage

```
data(EMresults)
```

Format

'data' is a list. 'convergeParams' is a list.

References

Ha, G., Roth, A., Khattra, J., Ho, J., Yap, D., Prentice, L. M., Melnyk, N., McPherson, A., Bashashati, A., Laks, E., Biele, J., Ding, J., Le, A., Rosner, J., Shumansky, K., Marra, M. A., Huntsman, D. G., McAlpine, J. N., Aparicio, S. A. J. R., and Shah, S. P. (2014). TITAN: Inference of copy number architectures in clonal cell populations from tumour whole genome sequence data. Genome Research, Published in advance July 24 2014. (PMID: 25060187)

`viterbiClonalCN` *Function to run the Viterbi algorithm for* **TitanCNA**.

Description

Function to run the Viterbi algorithm to find the optimal state path in the **TitanCNA** hidden Markov model (HMM). The states returned will indicate the optimal copy number and LOH state as well as the most likely clonal cluster for each data point. After running EM, use the converge parameters and the input data to infer the optimal state for each position. This function makes calls to a C subroutine that allows the algorithm to be run more efficiently.

Usage

```
viterbiClonalCN(data, convergeParams, genotypeParams = NULL)
```

Arguments

`data`	`list` object that contains the components for the data to be analyzed. `chr`, `posn`, `ref`, and `tumDepth` that can be obtained using `loadAlleleCounts`, and `logR` that can be obtained using `correctReadDepth` and `getPositionOverlap` (see Example).
`convergeParams`	`list` object that is returned from the function `runEMclonalCN` in **TitanCNA**.
`genotypeParams`	If `convergeParams` does not contain a `genotypeParams` element, then the user can pass this as an argument.

Details

It is difficult to interpret the output of this function directly. The user should use the function `outputTitanResults` after.

Value

`numeric` `array` containing the integer states corresponding to each data point in `data`.

Author(s)

Gavin Ha <gavinha@gmail.com>

References

Ha, G., Roth, A., Khattra, J., Ho, J., Yap, D., Prentice, L. M., Melnyk, N., McPherson, A., Bashashati, A., Laks, E., Biele, J., Ding, J., Le, A., Rosner, J., Shumansky, K., Marra, M. A., Huntsman, D. G., McAlpine, J. N., Aparicio, S. A. J. R., and Shah, S. P. (2014). TITAN: Inference of copy number architectures in clonal cell populations from tumour whole genome sequence data. Genome Research, Published in advance July 24 2014. (PMID: 25060187)

See Also

`outputTitanResults`, `loadAlleleCounts`

Examples

```
data(EMresults)

#### COMPUTE OPTIMAL STATE PATH USING VITERBI ####
optimalPath <- viterbiClonalCN(data, convergeParams)
```

`WIG Import Functions` *WIG Import Functions*

Description

Fast fixedStep WIG file reading and parsing

Usage

```
wigToRangedData(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.

Author(s)

Daniel Lai

References

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

See Also

`wigToRangedData` is a wrapper around these functions for easy WIG file loading and structure formatting. It is taken from the **HMMcopy** package.

Examples

```
map <- system.file("extdata", "map_chr2.wig", package = "TitanCNA")
mScore <- as.data.frame(wigToRangedData(map))
```

Index