Package 'TitanCNA'

March 21, 2025

Type Package

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

Version 1.44.0

Date 2019-10-03

Author Gavin Ha

Maintainer Gavin Ha <gha@fredhutch.org>

Depends R (>= 3.5.1)

Imports BiocGenerics (>= 0.31.6), IRanges (>= 2.6.1), GenomicRanges (>= 1.24.3), VariantAnnotation (>= 1.18.7), foreach (>= 1.4.3), GenomeInfoDb (>= 1.8.7), data.table (>= 1.10.4), dplyr (>= 0.5.0),

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

License GPL-3

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

URL https://github.com/gavinha/TitanCNA

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

git_branch RELEASE_3_20

git_last_commit 3826ddb

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2025-03-20

Contents

TitanCNA-package 2
computeSDbwIndex 4
correctIntegerCopyNumber 6
correctReadDepth 8

filterData 9
getPositionOverlap 11
haplotype-analysis-methods 12
loadAlleleCounts 15
loadDefaultParameters 17
output-methods 19
Plotting TITAN results 22
runEMclonalCN 25
TitanCNA trained dataset 29
viterbiClonalCN 30
WIG Import Functions 31

Index **33**

`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:	1.15.0
Date:	2017-05-13
License:	GPL-3

`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@broadinstitute.org>

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, 24: 1881-1893. (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, params,
                                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,
                              is.haplotypeData = FALSE,
                              correctResults = TRUE,
                              proportionThreshold = 0.05,
              proportionThresholdClonal = 0.05)
convergeParams <- results$convergeParams
results <- results$corrResults

#### GET SEGMENT RESULTS ####
segs <- outputTitanSegments(results, id = "test", convergeParams,
  filename = NULL, igvfilename = NULL)

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

```
ploidy <- tail(convergeParams$phi, 1)

par(mfrow=c(4, 1))
plotCNlogRByChr(results, chr = 2, segs = segs, ploidy = ploidy, normal = norm, 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")

plotSegmentMedians(segs, chr=2, resultType = "LogRatio", plotType = "CopyNumber",
                plot.new = TRUE, ylim = c(0, 4), 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", use.corrected.cn =TRUE,
    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.
`use.corrected.cn`	`TRUE`: Will use corrected copy number calls for computing S_Dbw validity index.

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. If `data.type='Both'`, then the sum of the S_Dbw for '`LogRatio`' and '`AllelicRatio`' are added together. This helps account for both data types when choosing the optimal solution.

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, 24: 1881-1893. (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,
                              correctResults = TRUE,
                              proportionThreshold = 0.05,
```

```
                proportionThresholdClonal = 0.05)

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

`correctIntegerCopyNumber` *Compute purity and ploidy corrected log ratios; recompute integer CN for high-level amplifications.*

Description

TitanCNA uses a finite state space that defines a maximum number of copies to model. High-level amplifications that exceed this defined maximum need to be corrected and reported as the likely copy number based on the observed data. `correctIntegerCN` performs two tasks: (1) correct log ratio based on purity and ploidy, and then convert to decimal CN value; (2) Correct bins (from `cn`) and segments (from `segs`) in which the original predicted integer copy number was assigned the maximum CN state; bins and segments for all of chromosome X are also corrected, if provided in the input.

Usage

```
correctIntegerCN(cn, segs, purity, ploidy, maxCNtoCorrect.autosomes = NULL,
 maxCNtoCorrect.X = NULL, correctHOMD = TRUE, minPurityToCorrect = 0.2, gender = "male",
  chrs = c(1:22, "X"))
```

Arguments

`cn`	data.table object output from the function outputTitanResults
`segs`	data.table object output from the function outputTitanSegments
`purity`	Float type of the 1 minus the normal contamination estimate from TitanCNA
`ploidy`	Float type of the average tumor ploidy estimate from TitanCNA
`maxCNtoCorrect.autosomes`	Bins and segments in autosomes with this copy number value or higher will be corrected. If `NULL`, then it will use the original copy number value from the input data.
`maxCNtoCorrect.X`	Bins and segments in chromosome X, if provided, with this copy number value or higher will be corrected. If `NULL`, then it will use the original copy number value from the input data.
`minPurityToCorrect`	If `purity` is less than `minPurityToCorrect`, then `Corrected_Copy_Number` will retain the same copy number values as the input copy number.
`correctHOMD`	If `TRUE`, then will correct the copy number of homozygous deletion bins and segments based on purity and ploidy corrected log ratios.
`gender`	data.frame containing list of centromere regions. This should contain 3 columns: chr, start, and end. If this argument is used, then data at and flanking the centromeres will be removed.
`chrs`	Chromosomes to consider for copy number correction.

Value

cn: data.table object that contains the same columns as the input object but also includes new columns logR_Copy_Number, Corrected_Copy_Number, Corrected_Call. segs: data.table object that contains the same columns as the input object but also includes new columns logR_Copy_Number, Corrected_Copy_Number, Corrected_Call, Corrected_MajorCN, Corrected_MinorCN. Column definitions:

logR_Copy_Number
: Purity and ploidy corrected log ratios that have been converted to a decimal-based copy number value.

Corrected_Copy_Number
: round(logR_Copy_Number)

Corrected_Call
: String representation of Corrected_Copy_Number; HLAMP=high-level amplification is assigned to bins/segments that have been corrected.

Corrected_MajorCN
: Purity and ploidy corrected integer (rounded) major copy number value.

Corrected_MinorCN
: Purity and ploidy corrected integer (rounded) minor copy number value.

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, 24: 1881-1893. (PMID: 25060187)

See Also

outputTitanResults, outputTitanSegments

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, correctResults = TRUE,
                              proportionThreshold = 0.05, recomputeLogLik = FALSE,
                              proportionThresholdClonal = 0.05,
                              is.haplotypeData = FALSE)
## use corrected parameters
convergeParams <- results$convergeParam
## use corrected results
results <- results$corrResults
## get normal contamination and ploidy estimates
```

```
norm <- tail(convergeParams$n,1)
ploidy <- tail(convergeParams$phi,1)

#### OUTPUT SEGMENTS ####
segs <- outputTitanSegments(results, id = "test", convergeParams,
  filename = NULL, igvfilename = NULL)
corrIntCN.results <- correctIntegerCN(results, segs, 1 - norm, ploidy, maxCNtoCorrect.autosomes = NULL,
 maxCNtoCorrect.X = NULL, correctHOMD = TRUE, minPurityToCorrect = 0.2, gender = "female", chrs = 2)
```

`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>, Yikan Wang <ykwang@bccrc.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, 24: 1881-1893. (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)
```

---

filterData *Filter list object based on read depth and missing data and returns a filtered data.table object.*

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,
    centromeres = NULL, centromere.flankLength = 0)
```

Arguments

data	data.table 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.
centromeres	data.frame containing list of centromere regions. This should contain 3 columns: chr, start, and end. If this argument is used, then data at and flanking the centromeres will be removed.
centromere.flankLength	Integer indicating the length (in base pairs) to the left and to the right of the centromere designated for removal of data.

Details

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

Value

The same data.table object 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, 24: 1881-1893. (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)
```

`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, dataVal)
```

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`.
`dataVal`	`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, 24: 1881-1893. (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)
```

---

haplotype-analysis-methods

*Function to load tumour allele counts from a text file or data.frame and returns a data.table (*loadHaplotypeAlleleCounts*). Function to load phased heterozygous sites from a VCF file (*getHaplotypesFromVCF*)*

Description

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

Usage

```
loadHaplotypeAlleleCounts(inCounts, cnfile, fun = "sum", haplotypeBinSize = 1e5,
   minSNPsInBin = 3, chrs = c(1:22, "X"), minNormQual = 200,
   genomeStyle = "NCBI", sep = "\t", header = TRUE, seqinfo = NULL,
 mapWig = NULL, mapThres = 0.9, centromere = NULL, minDepth = 10, maxDepth = 1000)

getHaplotypesFromVCF(vcfFile, chrs = c(1:22, "X"), build = "hg19", genomeStyle = "NCBI",
   filterFlags = c("PASS", "10X_RESCUED_MOLECULE_HIGH_DIVERSITY"),
   minQUAL = 100, minDepth = 10, minVAF = 0.25, altCountField = "AD",
   keepGenotypes = c("1|0", "0|1", "0/1"), snpDB = NULL)

 loadBXcountsFromBEDDir(bxDir, chrs = c(1:22, "X", "Y"), minReads = 2)
```

Arguments

inCounts	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**.
cnfile	Path to file containing GC-bias and maappability corrected molecule coverage for given bin size.
vcfFile	Path to phased variant VCF file from LongRanger 2.1. The file name must have the suffix *phase_variants.vcf.gz.
bxDir	Path to directory containing tumor bed files for each chromosome containing BX tags.
fun	The function ('SNP', 'sum', 'mean') to use to summarize within each user defined bin using haplotypeBinSize and haplotype block defined by the phaseSet ID from thte 9th column of inCounts. 'SNP' - uses the phased allele counts each individual SNP; phased allele for the higher coverage (determined within each bin) haplotype is chosen. 'sum' - uses the read count sum across all phased SNPs for the higher coverage haplotype within a bin normalized by the total depth across all SNPs in a bin; each SNP in the bin is assigned this fraction. 'mean' - uses the mean (rounded) read count across all phased SNPs for the higher coverage haplotype within a bin normalized by the mean (rounded) depth across all SNPs in a bin; each SNP in the bin is assigned this rounded count and depth.
haplotypeBinSize	Bin size used to summarize SNPs based on phased haplotypes. See fun for the summarization approaches within a bin.
minSNPsInBin	The minimum number of SNPs required in each haplotypeBinSize for analysis. See fun for the summarization approaches within a bin.
chrs	Vector containing list of chromosomes to include in output.
minNormQual	Quality threshold to use for filtering; SNPs with lower than this value are excluded. This quality is any metric that provides the confidence of the locus being a true germline heterozygous SNP.
minReads	Minimum number of reads per barcode.

genomeStyle	The genome style to use for chromosomes. Use one of 'NCBI' or 'UCSC'. It does not matter what style is found in inCounts, genomeStyle will be the style returned. Invokes setGenomeStyle.
build	Human genome reference build. Default: hg19.
snpDB	Path to SNP VCF file to use for specifying sites to retain.
minQUAL	Variants with quality (QUAL field) greater or equal to this value will be retained.
minDepth	Variants with read depth greater than or equal to this value will be retained.
maxDepth	Variants with read depth lower than or equal to this value will be retained.
minVAF	Variants with a variant/reference allele fraction of greater than or equal to this value will be retained.
altCountField	Specify the alternate count field name. Defaulat is "AD".
keepGenotypes	Genotypes to retain. Default is to keep these genotypes strings: 1|0, 0|1, 0/1
filterFlags	Specify the FILTER flags to retain.
sep	Character indicating the delimiter used for the columns for infile. Default is tab-delimited, "\t".
header	logical to indicate if the input tumour counts file contains a header line.
seqinfo	Seqinfo-class object describing chromosome information. If NULL, then will load seqinfo for hg19 system.files('extdata', 'Seqinfo_hg19.rda', package='TitanCNA'.
mapWig	Mappability score WIG file for binned data.
mapThres	Minimum mappability score of region/sequence overlapping variants to retain.
centromere	File containing reference genome gap file representing centromere locations. Usually obtained from UCSC.

Value

loadHaplotypeAlleleCounts returns a data.table containing components for

chr	Chromosome; character, genomeStyle naming convention
posn	Position; integer
phaseSet	Phase block identifier, numeric or character
refOriginal	Reference allele read count at SNP; numeric
tumDepthOriginal	Coverage at SNP; numeric
ref	Phased allele count values of higher coverage haplotype based on approach used (SNP, sum, mean); numeric
nonRef	Phased allele count values of lower coverage haplotype; tumDepth minus ref; numeric
tumDepth	Mean or sum of SNP read coverage; numeric
HapltypeRatio	Sum of read coverage of phased alleles of higher coverage haplotype normalized by tumDepth; numeric
haplotypeCount	Phased allele read count; numeric

getHaplotypesFromVCF returns a list containing 2 components

vcf.filtered	VCF object containing the list of heterozygous variants after filtering.
geno.gr	GRanges object containing the genotype information of the VCF

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, 24: 1881-1893. (PMID: 25060187)

See Also

loadDefaultParameters, plotHaplotypeFraction

Examples

```
  ## Not run:
  infile <- "test_alleleCounts_chr2_with_phaseInfo.txt"
  haplotypeBinSize <- 1e5
  phaseSummarizeFun <- "sum"
  ## will load seqinfo_hg19 provided by TitanCNA package
  data <- loadHaplotypeAlleleCounts(infile, fun = phaseSummarizeFun,
      haplotypeBinSize = haplotypeBinSize, minSNPsInBin = 3,
      chrs = c(1:22, "X"), minNormQual = 200,
      genomeStyle = "NCBI", seqinfo = NULL)

## End(Not run)

  ## Not run:
  vcfFile <- "test.vcf"
  hap <- getHaplotypesFromVCF(vcfFile, chrs = c(1:22,"X"), build = "hg19",
    filterFlags = c("PASS", "10X_RESCUED_MOLECULE_HIGH_DIVERSITY"),
    minQUAL = 100, minDepth = 10, minVAF = 0.25,
    keepGenotypes = ("1|0", "0|1", "0/1"))


## End(Not run)
```

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

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", header = TRUE)

setGenomeStyle(x, genomeStyle = "NCBI", species = "Homo_sapiens", filterExtraChr = TRUE)
```

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".
header	logical to indicate if the input tumour counts file contains a header line.
x	character vector of chromosome names to change.
species	character denoting the species
filterExtraChr	logical; if TRUE, then will return the list of chromosomes given by extractSeqlevelsByGroup for the species and for autosomes and sex chromosomes, which means that only the major chromosomes are returned (i.e. 1:22, X, Y).

Value

loadAlleleCounts returns a data.table 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, 24: 1881-1893. (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", header = TRUE)

## use the UCSC chromosome naming convention instead ##
data$chr <- setGenomeStyle(data$chr, genomeStyle = "UCSC")
## Not run:
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,
    hetBaselineSkew = NULL, alleleEmissionModel = "binomial",
    symmetric = TRUE, data = NULL)
```

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 in the analysis. Each cluster represents a potential clone. Using '1' treats the sample as being clonal (no subclonality). '2' or higher treats the tumour data as being subclonal.
`skew`	`numeric` float (between 0 to 0.5) 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. Use `0` or `NULL` for no skew.
`hetBaselineSkew`	Allelic reference base skew for heterozygous states (e.g. 1:1, 2:2, 3:3). Value is additive to baseline allelic ratios (which may already be adjusted by `skew`). Use `0` or `NULL` for no skew; use from range between 0 to 0.5.
`alleleEmissionModel`	Specify the emission model to use for the allelic input data. `"binomial"` or `"Gaussian"`.
`symmetric`	`logical`; if `TRUE`, then treat genotypes as symmetric. This should always be `TRUE` because `symmetric=FALSE` is deprecated. See Details.
`data`	`data` is the output of function `loadAlleleCounts`. If provided and `symmetric=TRUE`, then it will compute the median allelic ratio to use as the baseline for heterozygous genotypes; otherwise, the baseline will default to 0.55 (reference/depth) if `data=NULL`. If `symmetric=FALSE`, this argument will not be used.

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, 24: 1881-1893. (PMID: 25060187)

See Also

`loadAlleleCounts`

Examples

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

output-methods *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,
    is.haplotypeData = FALSE, posteriorProbs = FALSE, subcloneProfiles = TRUE,
    correctResults = TRUE, proportionThreshold = 0.05,
  proportionThresholdClonal = 0.05, recomputeLogLik = TRUE, rerunViterbi = FALSE,
    verbose = TRUE)

outputModelParameters(convergeParams, results, filename,
  S_Dbw.scale = 1, S_Dbw.method = "Tong", S_Dbw.useCorrectedCN = TRUE)

outputTitanSegments(results, id, convergeParams, filename = NULL,
  igvfilename = NULL)
```

Arguments

id	Character string identifier for sample
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.
igvfilename	Path of the file to write the IGV seg file.
posteriorProbs	Logical TRUE to include the posterior marginal probabilities in printing to filename.
is.haplotypeData	Logical TRUE if the data contains the haplotype information. In particular, the column headers HaplotypeCount, HaplotypeDepth, HaplotypeRatio are included.
subcloneProfiles	Logical TRUE to include the subclone profiles to the output data.frame. Currently, this only works for 1 or 2 clonal clusters.
correctResults	Logical TRUE to correct the results by removing empty clusters and adjusting cellular prevalence and normal contamination parameters accordingly.
recomputeLogLik	Logical TRUE to re-run forwards-backwards to re-estimate the log-likelihood after correcting results (e.g. correctResults is TRUE)

`rerunViterbi` Logical TRUE to re-run viterbi to segment the results again after correcting results (e.g. `correctResults` is TRUE)

`proportionThreshold` Minimum proportion of the genome altered (by SNPs) for a cluster to be retained. Clonal clusters having lower proportion of alteration are removed.

`proportionThresholdClonal` Minimum proportion of genome altered by clonal events (by SNPs) for the highest cellular prevalence cluster. If the highest prevalence cluster contains lower proportion of events than this threshold, this cluster will be removed and the next highest (subclonal) cluster will be readjusted to be the clonal cluster.

`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`.

`S_Dbw.useCorrectedCN` `TRUE`: Will use corrected copy number calls for computing S_Dbw validity index.

`verbose` Print status messages.

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= S_Dbw.scale * dens.bw + scat

12) S_Dbw dens.bw, scat, validity index is computed for `LogRatio` and `AllelicRatio` datatypes, as well as the combination of `Both`. For `Both`, the values are summed for both datatypes.

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

Value

outputTitanResults also returns a list containing the following:

results	TITAN results, uncorrected for cluster number and parameters
corrResults	TITAN results, corrected by removing empty clusters and parameters adjusted accordingly.
convergeParams	Corrected parameter object

The results and corrResults are data.table objects, 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.

outputModelParameters returns a list containing the S_Dbw model selection:

dens.bw	
scat	
S_Dbw	S_Dbw.scale * dens.bw + scat

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, 24: 1881-1893. (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, correctResults = TRUE,
                              proportionThreshold = 0.05, recomputeLogLik = FALSE,
                              proportionThresholdClonal = 0.05,
                              is.haplotypeData = FALSE)
## use corrected parameters
convergeParams <- results$convergeParam
## use corrected results
results <- results$corrResults

#### OUTPUT RESULTS TO FILE ####
outparam <- paste0("cluster2_params.txt")
outputModelParameters(convergeParams, results, outparam)

#### OUTPUT SEGMENTS TO FILE ####
outseg <- paste0("cluster2_segs.txt")
outigv <- paste0("cluster2.seg")
segs <- outputTitanSegments(results, id = "test", convergeParams,
  filename = outseg, igvfilename = outigv)
# segment results also stored in data.frame "segs"
```

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 = c(1:22), geneAnnot = NULL, spacing = 4,
    xlim = NULL, ...)
plotClonalFrequency(dataIn, chr = c(1:22), normal = NULL, geneAnnot = NULL,
    spacing = 4, xlim = NULL, ...)
plotCNlogRByChr(dataIn, chr = c(1:22), segs = NULL, plotCorrectedCN = TRUE,
  geneAnnot = NULL, ploidy = NULL, normal = NULL, spacing = 4, alphaVal = 1, xlim = NULL, ...)
plotSubcloneProfiles(dataIn, chr = c(1:22), geneAnnot = NULL,
   spacing = 4, xlim = NULL, ...)
plotSegmentMedians(dataIn, resultType = "LogRatio", plotType = "CopyNumber", plotCorrectedCN = TRU
   chr = c(1:22), geneAnnot = NULL, ploidy = NULL, spacing = 4, alphaVal = 1, xlim = NULL,
   plot.new = FALSE, lwd = 8, ...)
plotHaplotypeFraction(dataIn, chr = c(1:22), resultType = "HaplotypeRatio", colType = "Haplotypes"
   phaseBlockCol = c("#9ad0f3", "#CC79A7"), geneAnnot = NULL, spacing = 4,  xlim = NULL,  ...)
```

Arguments

`dataIn`	Formatted **TitanCNA** results output from `outputTitanResults`. See Example.
`chr`	Plot results for specified `chr`. The default is to plot chromosomes 1 to 22. The chromosome naming style will be automatically set to the input `dataIn`.
`segs`	`data.frame` containing named columns: `Chromosome`, `Median_logR`, `Start_Position.bp.`, `End_Position.bp.`. This data can be read in from the segments generated by the TITANRunner pipeline. These segments will be overlaid in the plots as lines at the median log ratio for each segment.
`resultType`	For `plotSegmentMedians`: specify the data type ('LogRatio' or 'AllelicRatio') to plot. For `plotHaplotypeFraction`: specify the data type ('HaplotypeRatio' or 'AllelicRatio') to plot.
`plotType`	Specify whether to plot the 'CopyNumber' or 'Ratio' values for the `resultType`.
`colType`	Specify the color scheme to use: 'Haplotypes' or 'CopyNumber'. For 'Haplotypes', alternating blue and red used to illustrated the data within phased haplotype blocks. For 'CopyNumber', the same colors as `plotAllelicRatio` are used for allelic copy number events.
`plotCorrectedCN`	`TRUE` if the plot will use 'Corrected_Copy_Number' for color of data points or lines.
`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.
`phaseBlockCol`	Two-element vector specifying the color to plot for alternating haplotype phase blocks.
`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.
lwd	Explicitly specify the line width for segments being plot.
plot.new	TRUE if a new plot is used. Set to FALSE to overlay an existing 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.

plotSegmentMedians plots the segment means for 'LogRatio' or 'AllelicRatio' data. There are also two types of plots for each of the datatypes: 'Ratio' or 'CopyNumber'. For 'Ratio', the logRatio or allelic ratios in the output results files are plot. For 'CopyNumber', the y-axis is converted to the exponentiated absolute copy number levels for the easier interpretability of the results.

plotHaplotypeFraction plots the phased SNP read count of the higher coverage haplotype, normalized by the total coverage of the haplotype. For 'Haplotypes', alternating colors of blue and red are used to illustrate the phased haplotype blocks provided from the input data (see loadHaplotypeAlleleCounts).

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, 24: 1881-1893. (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,
  correctResults = TRUE, proportionThreshold = 0.05,
  proportionThresholdClonal = 0.05)
convergeParams <- results$convergeParams
results <- results$corrResults # use corrected results
#### PLOT RESULTS ####
norm <- tail(convergeParams$n, 1)
ploidy <- tail(convergeParams$phi, 1)

par(mfrow=c(4, 1))
plotCNlogRByChr(results, chr = 2, segs = NULL, ploidy = ploidy, normal = norm,
  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")

segs <- outputTitanSegments(results, id = "test", convergeParams)

plotSegmentMedians(segs, chr=2, resultType = "LogRatio",
  plotType = "CopyNumber", plot.new = TRUE)
```

---

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, params,
              txnExpLen = 1e15, txnZstrength = 5e05, maxiter = 15,
              maxiterUpdate = 1500, pseudoCounts = 1e-300,
              normalEstimateMethod = "map", estimateS = TRUE,
              estimatePloidy = TRUE, useOutlierState = FALSE,
              likChangeThreshold = 0.001, 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).
params	list object that contains major parameters: list object containing copy number and allelic ratio genotype parameters. list object containing the normal contamination parameters. list object containing the tumour ploidy parameters. list object containing the subclonality (cellular prevalence and clonal cluster) parameters. params 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.
likChangeThreshold	EM convergence criteria - stop EM when complete log likelihood changes less than the proportion specified by this argument.
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(numberClona`
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, 24: 1881-1893. (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, params,
```

```
maxiter = 3, maxiterUpdate = 500,
txnExpLen = 1e15, txnZstrength = 5e5,
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 anonymize 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 are generated by `runEMclonalCN`

Usage

```
data(EMresults)
```

Format

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

References

Shah SP et al. (2012). The clonal and mutational evolution spectrum of primary triple-negative breast cancers. Nature, 486(7403): 395-399. (PMID: 22495314)

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, 24: 1881-1893. (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, 24: 1881-1893. (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. wigToGRanges (new) and wigToRangedData (deprecated)*

Description

Fast fixedStep WIG file reading and parsing

Usage

```
wigToGRanges(wigfile, verbose = TRUE)
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

`GRanges` object with chromosome and position information, sorted in decreasing chromosal size and increasing position.

Author(s)

Gavin Ha, Daniel Lai

References

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

See Also

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

Examples

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

Index

∗ **IO**
correctReadDepth, 8
haplotype-analysis-methods, 12
loadAlleleCounts, 15
output-methods, 19
TitanCNA-package, 2
WIG Import Functions, 31
∗ **aplot**
Plotting TITAN results, 22
∗ **attribute**
loadDefaultParameters, 17
∗ **color**
Plotting TITAN results, 22
∗ **datasets**
TitanCNA trained dataset, 29
∗ **htest**
runEMclonalCN, 25
viterbiClonalCN, 30
∗ **iteration**
runEMclonalCN, 25
viterbiClonalCN, 30
∗ **manip**
computeSDbwIndex, 4
correctIntegerCopyNumber, 6
correctReadDepth, 8
filterData, 9
getPositionOverlap, 11
output-methods, 19
runEMclonalCN, 25
TitanCNA-package, 2
viterbiClonalCN, 30
∗ **models**
runEMclonalCN, 25
viterbiClonalCN, 30
∗ **multivariate**
runEMclonalCN, 25
viterbiClonalCN, 30
∗ **package**
TitanCNA-package, 2

array, *11, 12*, *19*, *30*

computeSDbwIndex, 4, *20*, *22*, *25*
convergeParams *(*TitanCNA trained dataset*)*, 29
correctIntegerCN *(*correctIntegerCopyNumber*)*, 6
correctIntegerCopyNumber, 6
correctReadcount, *8, 9*
correctReadDepth, 8, *12*, *19*, *26*, *29, 30*

data *(*TitanCNA trained dataset*)*, 29
data.frame, *9–12*, *23*
data.table, *6, 7*, *9, 10*, *12*, *14–16*, *21*

EMresults *(*TitanCNA trained dataset*)*, 29

filterData, 9
foreach, *27, 28*

getHaplotypesFromVCF *(*haplotype-analysis-methods*)*, 12
getPositionOverlap, 11, *19*, *26*, *30*
GRanges, *31*

haplotype-analysis-methods, 12

list, *5*, *14*, *18, 19*, *21*, *26, 27*, *30*
loadAlleleCounts, *4, 5*, *10–12*, 15, *17–19*, *26*, *28–30*
loadBXcountsFromBEDDir *(*haplotype-analysis-methods*)*, 12
loadDefaultParameters, *15, 16*, 17, *26–28*
loadHaplotypeAlleleCounts, *24*
loadHaplotypeAlleleCounts *(*haplotype-analysis-methods*)*, 12

output-methods, 19
outputModelParameters, *5*, *20, 21*
outputModelParameters *(*output-methods*)*, 19
outputTitanResults, *4*, *6, 7*, *19–21*, *23*, *25*, *27*, *30*
outputTitanResults *(*output-methods*)*, 19
outputTitanSegments, *6, 7*

outputTitanSegments *(*output-methods*)*, 19

plot, *24*
plotAllelicRatio, *24*
plotAllelicRatio *(*Plotting TITAN results*)*, 22
plotClonalFrequency, *24*
plotClonalFrequency *(*Plotting TITAN results*)*, 22
plotCNlogRByChr, *23, 24*
plotCNlogRByChr *(*Plotting TITAN results*)*, 22
plotHaplotypeFraction, *15*, *24*
plotHaplotypeFraction *(*Plotting TITAN results*)*, 22
plotSegmentMedians, *24*
plotSegmentMedians *(*Plotting TITAN results*)*, 22
plotSubcloneProfiles, *24*
plotSubcloneProfiles *(*Plotting TITAN results*)*, 22
Plotting TITAN results, 22

runEMclonalCN, *19*, *22, 23*, *25*, 25, *29, 30*

setGenomeStyle, *14*
setGenomeStyle *(*loadAlleleCounts*)*, 15

TitanCNA *(*TitanCNA-package*)*, 2
TitanCNA trained dataset, 29
TitanCNA-dataset *(*TitanCNA trained dataset*)*, 29
TitanCNA-package, 2

viterbiClonalCN, *19*, *22*, *28*, 30

WIG Import Functions, 31
wigToGRanges, *31*
wigToGRanges *(*WIG Import Functions*)*, 31
wigToRangedData, *8, 9*
wigToRangedData *(*WIG Import Functions*)*, 31