Package ‘Clomial’

November 20, 2024

Type Package

Title Infers clonal composition of a tumor

Version 1.43.0

Date 2018-05-31

Author Habil Zare and Alex Hu

Maintainer Habil Zare <zare@u.washington.edu>

Depends R (>= 2.10), matrixStats

Imports methods, permute

Description Clomial fits binomial distributions to counts obtained from Next Gen Sequencing data of multiple samples of the same tumor. The trained parameters can be interpreted to infer the clonal structure of the tumor.

License GPL (>= 2)

biocViews Genetics, GeneticVariability, Sequencing, Clustering, MultipleComparison, Bayesian, DNASeq, ExomeSeq, TargetedResequencing, ImmunoOncology

LazyLoad yes

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

git_branch devel

git_last_commit ed32b0f

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2024-11-20

Contents

`Clomial-package` *Fits a binomial model to data from multiple samples of a single tumor to infer its clonal decomposition.*

Description

Clomial fits binomial distributions to counts obtained from Next Gen Sequencing data of multiple samples of the same tumor. The trained parameters can be interpreted to infer the clonal structure of the tumor.

Details

Package:	Clomial
Type:	Package
Version:	0.99.0
Date:	2014-02-11
License:	GPL (>= 2)

The main function is Clomial() which requires 2 matrices Dt and Dc among its inputs. They contain the counts of the alternative allele, and the total number of processed reads, accordingly. Their rows correspond to the genomic loci, and their columns correspond to the samples. Several models should be trained using different initial values to escape from local optima, and the best one in terms of the likelihood can be chosen by choose.best() function.

Author(s)

Habil Zare and Alex Hu

Maintainer: Habil Zare <zare@u.washington.edu>

References

Inferring clonal composition from multiple sections of a breast cancer, Zare et al., PLoS Computational Biology 10.7 (2014): e1003703.

See Also

`Clomial`, `choose.best`, `Clomial.iterate`, `Clomial.likelihood`, `compute.bic`, `breastCancer`

Examples

```
set.seed(1)
data(breastCancer)
Dc <- breastCancer$Dc
Dt <- breastCancer$Dt
ClomialResult <-Clomial(Dc=Dc,Dt=Dt,maxIt=20,C=4,doParal=FALSE,binomTryNum=2)
chosen <- choose.best(models=ClomialResult$models)
M1 <- chosen$bestModel
print("Genotypes:")
print(round(M1$Mu))
print("Clone frequencies:")
print(M1$P)
```

`breastCancer` *Breast cancer data for clonal decomposition.*

Description

Counts data from multiple samples of a single primary breast cancer obtained by deep, next-generation sequencing. The file is consist of two matrices `Dt` and `Dc` which contain the counts of the alternative alleles, and the total number of counts on each genomic loci for every tumor samples, accordingly.

Usage

```
data(breastCancer)
```

Format

A list containing 2 matrices.

Details

Each matrix contains counts of reads mapped to 17 genomic loci for 12 tumor samples where the column A5-2 corresponds to the normal sample.

References

Inferring clonal composition from multiple sections of a breast cancer, Zare et al., Submitted.

See Also

`Clomial`

Examples

```
data(breastCancer)
breastCancer$Dt
```

`choose.best` *Chooses the best trained Clomial model.*

Description

Given the output of Clomial function, the likelihoods of all models are compared, and the best model is determined.

Usage

```
choose.best(models, U = NULL, PTrue = NULL, compareTo = NULL, upto =
"All", doTalk=FALSE)
```

Arguments

`models`	The models trained by `Clomial` function.
`U`	The optional genotype matrix used for comparison.
`PTrue`	The optional clone frequency matrix used for comparison.
`compareTo`	The index of the model against which all other models are compared. Set to `NULL` to disable.
`upto`	The models with index less than this value are considered. Set to "All" to include every model.
`doTalk`	If TRUE, information on number of analyzed models is reported.

Details

If `compareTo`, `U`, and `PTrue` are `NULL` no comparison will be done, and the function runs considerably faster.

Value

A list will be made with the following entries:

`err`	A list with 2 entries; err$P and err$U the vectors of clonal frequency errors, and genotype errors, accordingly.
`Li`	A vector of the best obtained log-likelihood for each model.
`bestInd`	The index of the best model in terms of log-likelihood.
`comparison`	If `compareTo` is not `NULL`, the result of comparison with the corresponding model is reported.
`bestModel`	The best model in terms of log-likelihood.
`seconds`	A vector of the time taken, in seconds, to train each model.

Note

When the number of assumed clones, `C`, is greater than 6, the comparison will be time taking because all possible permutations of clones should be considered. The running time will be slowed down by `C!`.

Author(s)

Habil Zare

References

Inferring clonal composition from multiple sections of a breast cancer, Zare et al., Submitted.

See Also

`Clomial`, `Clomial.likelihood`, `Clomial.iterate`

Examples

```
set.seed(4)
data(breastCancer)
Dc <- breastCancer$Dc
Dt <- breastCancer$Dt
ClomialResult <-Clomial(Dc=Dc,Dt=Dt,maxIt=20,C=4,doParal=FALSE,binomTryNum=5)
chosen <- choose.best(models=ClomialResult$models)
M1 <- chosen$bestModel
print("Genotypes:")
round(M1$Mu)
print("Clone frequencies:")
M1$P
bestInd <- chosen$bestInd
plot(chosen$Li,ylab="Log-likelihood",type="l")
points(x=bestInd,y=chosen$Li[bestInd],col="red",pch=19)
```

`Clomial` *Fits several binomial models to data from multiple samples of a single tumor.*

Description

Using EM, trains several models using different initial values to escape from local optima. The best one in terms of the likelihood can be later chosen by choose.best() function.

Usage

```
Clomial(Dt = NULL, Dc = NULL, DcDtFile = NULL, C, doParal=FALSE,
outPrefix = NULL, binomTryNum = 1000, maxIt = 100, llCutoff = 0.001,
jobNamePrefix = "Bi", qstatWait = 2, fitBinomJobFile = NULL,
jobShare = 10, ignoredSample = c(),  fliProb=0.05, conservative=TRUE,
doTalk=FALSE)
```

Arguments

`Dt`	A matrix which contains the counts of the alternative allele where rows correspond to the genomic loci, and columns correspond to the samples.
`Dc`	A matrix which contains the counts of the total number of mapped reads where rows correspond to the genomic loci, and columns correspond to the samples.
`DcDtFile`	A file from which the data can optionally be loaded. It should contain the matrices Dc and Dt.
`C`	The assumed number of clones.
`doParal`	Boolean where TRUE means, in Linux, models with different initialization are trained in parallel on a cluster using qsub.
`outPrefix`	A prefix for the path to save the results.
`binomTryNum`	The number of models trained using different initialization.
`maxIt`	The maximum number of EM iterations.
`llCutoff`	EM iterations stops if the relative improvement in the log-likelihood is not more than this threshold.
`jobNamePrefix`	If run in parallel, this prefix will be used to name the jobs on the cluster.
`qstatWait`	The waiting time between qstat commands to assess the number of running and waiting jobs.
`fitBinomJobFile`	If run in parallel, this is the script which loads data, trains a model using a random initialization, and saves the results.
`jobShare`	If run in parallel, the job_share option of qsub determines the priority of jobs over other submitted jobs.
`ignoredSample`	A vector of indices of samples which will be ignored in training. Used by experts only to measure the stability of the results.
`fliProb`	A "flipping probability" used for noise injection which can be disabled when `fliProb=0`. After the first EM iteration, each entry of the matrix `Mu` such as `m` may change to `1-m` with this probability. This probability decreases on subsequent iterations.
`conservative`	Boolean where TRUE means noise will be injected only if likelihood is improved after an EM iteration, otherwise the original Mu matrix will be used for the next iteration. For expert use only.
`doTalk`	If TRUE, information on the EM optimization iterations is reported.

Details

The likelihood of the model, given the hidden variables and the parameters, can be computed based on a combination of binomial distributions. In each EM iteration, the likelihood is increased, however, due to presence of local optima, several models should be tried using different random initialization. For higher number of assumed clones, `C`, the parameter binomTryNum should be increased because the dimension of the search space grows linearly with `C`.

Value

Returns a list containing the entry called `models`, which is a list of the length equal to binomTryNum where each element is a trained model. For each trained model, `Mu` models the matrix of genotypes, where rows and columns correspond to genomic loci and clones, accordingly. Also, `P` is the matrix of clonal frequency where rows and columns correspond to clones and samples, accordingly. The first column of `P` corresponds to the normal clone. The history of `Mu`, `P`, and the log-likelihood over iterations is saved in lists `Ps`, `Mus`, and `Likelihoods`, accordingly.

Note

The parallel mode works only in Linux, and when qsub and qstat commands are available on a cluster.

Author(s)

Habil Zare

References

Inferring clonal composition from multiple sections of a breast cancer, Zare et al., Submitted.

See Also

`Clomial`, `choose.best`, `Clomial.iterate`, `compute.bic`, `breastCancer`

Examples

```
set.seed(1)
data(breastCancer)
Dc <- breastCancer$Dc
Dt <- breastCancer$Dt
ClomialResult <-Clomial(Dc=Dc,Dt=Dt,maxIt=20,C=4,binomTryNum=2)
chosen <- choose.best(models=ClomialResult$models)
M1 <- chosen$bestModel
print("Genotypes:")
round(M1$Mu)
print("Clone frequencies:")
M1$P
```

`Clomial.generate.data` *Generates simulated data to test performance of Clomial algorithm.*

Description

Data sets are simulated based on binomial distribution using random parameters for the model. The accuracy of the EM procedure can be estimated by comparing the inferred parameters vs. the known ones which were used to generate the data.

Usage

```
Clomial.generate.data(N, C, S, averageCoverage, mutFraction,
doSample1Normal = FALSE,erroRate=0,doCheckDc=TRUE)
```

Arguments

- `N` The number of genomic loci.
- `C` The number of clones.
- `S` The number of samples.
- `averageCoverage` The average coverage over each loci, each sample.
- `mutFraction` Should be in range 0-1. Each loci in every sample can be mutated with this probability.
- `doSample1Normal` If TRUE, no contamination with the tumor content is allowed for the normal sample. I.e. the first column of the generated `P` matrix will start with 1, and the rest of its entries will be equal to 0.
- `erroRate` The sequencing noise can be simulated by assigning a positive value to this parameter, which is the probability of reading a normal allele as the alternative allele, and vica versa.
- `doCheckDc` If TRUE, generating with be repeated until no row of Dc is all zeros to guarantee all loci have positive coverage in at least one sample.

Details

See the reference below for details.

Value

A list will be made with the following entries:

- `Dc` A matrix of simulated coverage for all loci and samples.
- `Dt` A matrix of alternative allele counts for all loci and samples.
- `Ptrue` The true clone frequency matrix used for generating the data.
- `U` The true genotype matrix used for generating the data.

`Likelihood`	The log-likelihood of the model with the true parameters.
`Phi`	The matrix of the second parameters of the binomial distributions; each entry is the probability that a read contains the variant allele at a locus in a sample.

Author(s)

Habil Zare

References

Inferring clonal composition from multiple sections of a breast cancer, Zare et al., Submitted.

See Also

`Clomial`, `Clomial.likelihood`

Examples

```
set.seed(1)
simulated <- Clomial.generate.data(N=20, C=4, S=10,
  averageCoverage=1000, mutFraction=0.1)
simulated$Dc
```

`Clomial.iterate` *Runs EM iterations until convergence of the Clomial model.*

Description

Given the data and the initial values for the model parameters, runs EM iterations until convergence of the Clomial model.

Usage

```
Clomial.iterate(Dt, Dc, Mu, P, maxIt=100, U = NULL, PTrue = NULL,
llCutoff = 10^(-3), computePFunction = compute.P.reparam,
doSilentOptim = TRUE, doTalk = TRUE, doLog = TRUE, debug = FALSE,
noiseReductionRate = 0.01, fliProb=0.05,conservative=TRUE)
```

Arguments

`maxIt`	The maximum number of EM iterations.
`Dt`	A matrix which contains the counts of the alternative allele where rows correspond to the genomic loci, and columns correspond to the samples.
`Dc`	A matrix which contains the counts of the total number of mapped reads where rows correspond to the genomic loci, and columns correspond to the samples.
`Mu`	The initial value for the Mu matrix which models the genotypes, where rows and columns correspond to genomic loci and clones, accordingly.

- `P` — The initial matrix of clonal frequency where rows and columns correspond to clones and samples, accordingly.
- `U` — The true value for `Mu`, used for debugging purposes only.
- `PTrue` — The true value for `P`, used for debugging purposes only.
- `llCutoff` — EM iterations stops if the relative improvement in the log-likelihood is not more than this threshold.
- `computePFunction` — The function used for updating `P`. For advanced development use only.
- `doSilentOptim` — If TRUE, the optimization massages will not be reported.
- `doTalk` — If FALSE, the function will be run in silent mode.
- `doLog` — Highly recommended to set to TRUE. Then, the computations will be done in log space to avoid numerical issues.
- `debug` — If TRUE, the debug mode will be turned on.
- `noiseReductionRate` — The noise will be reduce by this rate after each EM iteration.
- `fliProb` — A "flipping probability" used for noise injection which can be disabled when `fliProb=0`. After the first EM iteration, each entry of the matrix `Mu` such as `m` may change to `1-m` with this probability. This probability decreases on subsequent iterations.
- `conservative` — Boolean where TRUE means noise will be injected only if likelihood is improved after an EM iteration, otherwise the original Mu matrix will be used for the next iteration. For expert use only.

Details

Injecting noise can be done by assigning a positive value to `fliProb`, and can be disabled by `fliProb=0`. Noise injection is recommended for training models with a high number of clones (>4).

Value

A list will be made with the following entries:

- `Qs` — The history of matrices containing the posterior `Q` values.
- `Ps` — The history of `P` matrices.
- `Mus` — The history of `Mu` matrices.
- `Mu` — The value of `Mu` after convergence.
- `P` — The value of `P` after convergence.
- `llCutoff` — The threshold used to decide convergence.
- `LRatio` — The final relative improvement in the log likelihood which lead to convergence.
- `Likelihoods` — The history of log-likelihoods.
- `fliProb` — The final value of `fliProb` used for noise injection.
- `timeTaken` — An object of class “difftime” which reports the total computational time for EM iterations.
- `endTaken` — An object of class “POSIXct” (see DateTimeClasses) which reports the time EM iterations finished.

Author(s)

Habil Zare

References

Inferring clonal composition from multiple sections of a breast cancer, Zare et al., Submitted.

See Also

Clomial, Clomial, breastCancer

Examples

```
set.seed(1)
## Getting data:
data(breastCancer)
Dc <- breastCancer$Dc
Dt <- breastCancer$Dt
freq1 <- Dt/Dc
N <- nrow(Dc)
S <- ncol(Dc)
Cnum <- 4 ## assumed number of clones.
## Random initialization:
random1 <- runif(n=N*(Cnum-1),min=rowMins(freq1)*0.9,max=rowMaxs(freq1)*1.1)
random1[random1>1] <- 1
random1[random1<0] <- 0
Mu <- matrix(random1,N,Cnum-1)
Mu <- cbind( matrix(0,N,1), Mu )
rownames(Mu) <- rownames(Dc)
colnames(Mu) <- paste("C",1:Cnum,sep="")
P <- matrix(runif(Cnum*S),Cnum,S)
rownames(P) <- colnames(Mu)
colnames(P) <- colnames(Dc)
## Normalizing P:
for( t in 1:S ){
 s <- sum(P[,t])
 P[,t] <- P[,t]/s
}##End for.

## Running EM:
model1 <- Clomial.iterate(Dt=Dt, Dc=Dc, Mu=Mu, P=P)
print("Genotypes:")
round(model1$Mu)
print("Clone frequencies:")
model1$P
```

Clomial.likelihood *Computes the complete data log-likelihood of a Clomial model.*

Description

Computes the expected complete data log-likelihood of a Clomial model over all possible values of the hidden variables.

Usage

```
Clomial.likelihood(Dc, Dt, Mu, P)
```

Arguments

Dt	A matrix which contains the counts of the alternative allele where rows correspond to the genomic loci, and columns correspond to the samples.
Dc	A matrix which contains the counts of the total number of mapped reads where rows correspond to the genomic loci, and columns correspond to the samples.
Mu	The matrix which models the genotypes, where rows and columns correspond to genomic loci and clones, accordingly.
P	The matrix of clonal frequency where rows and columns correspond to clones and samples, accordingly.

Details

By assuming that the genomic loci and the samples are independent given the model parameters, the computation is simplified by first summing over the samples for a locus, and then summing over all the loci. This strategy avoids exploring the exponentially huge probability space.

Value

A list will be made with the following entries:

ll	The expectation of complete log-likelihood over the hidden variables.
llS	A vector of computed log-likelihoods at all loci.

Note

The likelihood is computed assuming the heterozygosity is 2.

Author(s)

Habil Zare

References

Inferring clonal composition from multiple sections of a breast cancer, Zare et al., Submitted.

See Also

Clomial, choose.best, compute.bic, breastCancer

Examples

```
set.seed(1)
data(breastCancer)
Dc <- breastCancer$Dc
Dt <- breastCancer$Dt
ClomialResult <-Clomial(Dc=Dc,Dt=Dt,maxIt=20,C=4,doParal=FALSE,binomTryNum=1)
model1 <- ClomialResult$models[[1]]
likelihood <- Clomial.likelihood(Dc=Dc, Dt=Dt, Mu=model1$Mu, P=model1$P)$ll
print(likelihood)
```

Clomial1000 *Pre-computed results of Clomial.*

Description

Pre-computed results of Clomial function are provided for demo purposes. It contains 1000 trained models on counts data from multiple samples of a single primary breast cancer obtained by deep, next-generation sequencing.

Usage

```
data(Clomial1000)
```

Format

`Clomial1000[["models"]]` is the list of trained models.

Details

Each model is the output of `Clomial.iterate()` function on the `breastCancer` data assuming there are 4 clones.

References

Inferring clonal composition from multiple sections of a breast cancer, Zare et al., Submitted.

See Also

Clomial, Clomial.iterate, choose.best, breastCancer

Examples

```
data(Clomial1000)
chosen <- choose.best(models=Clomial1000$models)
M1 <- chosen$bestModel
print("Genotypes:")
round(M1$Mu)
print("Clone frequencies:")
M1$P
bestInd <- chosen$bestInd
plot(chosen$Li,ylab="Log-likelihood",type="l")
points(x=bestInd,y=chosen$Li[bestInd],col="red",pch=19)
```

---

compute.bic *Computes BIC for a Clomial model.*

Description

Computes the Bayesian Information Criterion (BIC) for a Clomial model, which might be useful to estimate the number of clones. A "significantly" smaller BIC is usually interpreted as a better fit to the data.

Usage

```
compute.bic(Dc, Dt, Mu, P)
```

Arguments

`Dt`	A matrix which contains the counts of the alternative allele where rows correspond to the genomic loci, and columns correspond to the samples.
`Dc`	A matrix which contains the counts of the total number of mapped reads where rows correspond to the genomic loci, and columns correspond to the samples.
`Mu`	The matrix which models the genotypes, where rows and columns correspond to genomic loci and clones, accordingly.
`P`	The matrix of clonal frequency where rows and columns correspond to clones and samples, accordingly.

Details

The Bayesian Information Criterion (BIC) for a model is computed by subtracting the expected log-likelihood times 2, from the number of free parameters of the model times logarithm of the total number of observations. For a Clomial model, we have `BIC = (NC+SC-S)log(sum(Dc))-2L`, where `L` is the likelihood, `N` is the number of genomic loci, `C` is the assumed number of clones, `S` is the number of samples, and `sum(Dc)` is the total number of observed reads.

Value

A list will be made with the following entries:

bic	The BIC value.
aic	The AIC value.
obsNum	The total number of observed reades.

Note

Theoretically, a method such as the Bayesian information criterion (BIC) or the Akaike information criterion (AIC) may be applied to estimate the number of clones. However, in practice, the outcome of such approaches should be interpreted with great caution because some of the underlying assumptions of the statistical analysis may not be necessarily true for a given model. For example, while a "small" improvement in the BIC is generally considered as a sign to stop making the model more complicated, making such decisions is very objective, and requires relying on thresholds with little statistical basis.

Author(s)

Habil Zare

References

Inferring clonal composition from multiple sections of a breast cancer, Zare et al., Submitted.

See Also

Clomial

Examples

```
set.seed(1)
data(breastCancer)
Dc <- breastCancer$Dc
Dt <- breastCancer$Dt
bics <- c()
Clomial3 <-Clomial(Dc=Dc,Dt=Dt,maxIt=20,C=3,doParal=FALSE,binomTryNum=1)
model3 <- Clomial3$models[[1]]
bics[3] <- compute.bic(Dc=Dc,Dt=Dt, Mu=model3$Mu, P=model3$P)$bic
Clomial4 <-Clomial(Dc=Dc,Dt=Dt,maxIt=20,C=4,doParal=FALSE,binomTryNum=1)
model4 <- Clomial4$models[[1]]
bics[4] <- compute.bic(Dc=Dc,Dt=Dt, Mu=model4$Mu, P=model4$P)$bic
print(bics) ## 4 is a better estimate for the number of clones.
```

`compute.errors` *Computes the error of a Clomial model.*

Description

Given the true genotype and frequency matrices, finds the permutation of genotypes matrix which best matches the true genotypes and returns the corresponding errors.

Usage

```
compute.errors(Mu, U, P, PTrue)
```

Arguments

`Mu`	The matrix which models the genotypes, where rows and columns correspond to genomic loci and clones, accordingly.
`U`	The true genotype matrix defined similar to `Mu`.
`P`	The matrix of clonal frequency where rows and columns correspond to clones and samples, accordingly.
`PTrue`	The true clonal frequency matrix defined similar to `P`.

Details

Computing the error is useful for estimating the performance of inference on simulated, and for comparing different trained models. Genotype and frequency errors are defined as the normalized `l1`-error in reconstructing the genotype, and the clone frequency matrices, accordingly, where by normalized `l1`-error we mean the sum of absolute values of an error matrix divided by the size of the matrix.

Value

A list will be made with the following entries:

`UError` The `l1`-error of the genotype matrix normalized by the size of matrix.

`discretizedUError`
The `l1`-error of the rounded genotype matrix, i.e. the number of mismatching genotypes, normalized by the size of matrix

.

`PErrorAbsolute` The normalized `l1`-error of the clone frequency matrix.

`PErrorRelative` Each entry of the error clone frequency matrix is normalized by the corresponding entry in `PTrue`, and then the normalized `l1` norm is computed.

Note

The use of `UError` and `PErrorAbsolute` is recommended. Computing the error is not feasible for more than 7 clones because the number of all possible permutations is factorial in the number of clones which grows super fast. Such input will trigger an error message.

Author(s)

Habil Zare

References

Inferring clonal composition from multiple sections of a breast cancer, Zare et al., Submitted.

See Also

Clomial

Examples

```
set.seed(1)
data(breastCancer)
Dc <- breastCancer$Dc
Dt <- breastCancer$Dt
bics <- c()
ClomialResult <-Clomial(Dc=Dc,Dt=Dt,maxIt=20,C=3,doParal=FALSE,binomTryNum=2)
model1 <- ClomialResult$models[[1]]
model2 <- ClomialResult$models[[2]]
## Comparing 2 trained models:
compute.errors(Mu=model1$Mu,U=model2$Mu,P=model1$P,PTrue=model2$P)
```

Index