Package 'QuantumClone'

November 13, 2017

Title Clustering Mutations using High Throughput Sequencing (HTS) Data

Version 1.0.0.6

Description Using HTS data, clusters mutations in order to recreate putative clones from the data provided. It requires genotype at the location of the variant as well as the depth of coverage and number of reads supporting the mutation. Additional information may be provided, such as the contamination in the tumor sample. This package also provides a function QuantumCat() which simulates data obtained from tumor sequencing.

URL https://github.com/DeveauP/QuantumClone

BugReports https://github.com/DeveauP/QuantumClone/issues

Depends R (>= 3.1.0), compiler, foreach

Imports doParallel, parallel, gridExtra, ggplot2, NbClust, DEoptim, optimx, fpc

Suggests knitr,rmarkdown,testthat

License GPL-2

LazyData true

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

Author Paul Deveau [aut, cre]

Maintainer Paul Deveau <quantumclone.package@gmail.com>

Repository CRAN

Date/Publication 2017-11-13 10:25:32 UTC

R topics documented:

`add_leaf_list` *Phylogenetic tree leaf*

Description

Adds a leaf to an already built tree. Output is a list of all possibilities.

Usage

```
add_leaf_list(leaf, connexion_list, timepoints, d, selector_position)
```

Arguments

`leaf`	A vector of cellularities (ranging from 0 to 1)
`connexion_list`	List containing 1. An interaction matrix concatenated with the cellularity of each cluster (one line per cluster)
`timepoints`	A numeric vector giving the spatial and/or temporal distribution of the samples
`d`	The initial number of clusters
`selector_position`	The row of the studied leaf in the data frame.

`BIC_criterion` *Bayesian Information Criterion*

Description

Computes BIC from a list of outputs of EM algorithm, then returns the position with minimal BIC

Usage

```
BIC_criterion(EM_out_list, model.selection)
```

Arguments

`EM_out_list`	list of outputs from EM.algo or FullEM
`model.selection`	The function to minimize for the model selection: can be "AIC", "BIC", or numeric. In numeric, the BIC function is modified. If variance: returns max(abs(1 - Var(cluster)/expected(Var)))

BIC_criterion_FLASH *Compute criterion FLASH*

Description

Computes BIC from a list of outputs of EM algorithm, then returns the position with minimal BIC

Usage

```
BIC_criterion_FLASH(Obj, Mut_num, k, model.selection, s)
```

Arguments

Obj	Numeric vector with objective function values
Mut_num	Number of mutations to cluster
k	the number of clusters (in the same order as Obj)
model.selection	The function to minimize for the model selection: can be "AIC", "BIC", or numeric. In numeric, the function
s	Number of samples

CellularitiesFromFreq *Cellularities from allele frequency*

Description

Creates all possibilities for one mutation in one sample (given a genotype), then computes the cellularity associated with each possibility and finally the probability of each possibility

Usage

```
CellularitiesFromFreq(chr, position, Alt, Depth, Freec_ratio = NULL,
  Genotype = NULL, subclone.genotype = NULL, subclone.cell = NULL,
  contamination, restrict.to.AB = FALSE, force.single.copy = FALSE)
```

Arguments

chr	The chromosome on which is the position (numeric value, not chr1 as in BED files)
position	The genomic position of the mutation
Alt	Number of reads supporting the variation
Depth	Number of reads mapped at the position
Freec_ratio	The FREEC output associated with the sample of interest

`Genotype`	If the FREEC output is not given, the genotype associated with the locus (for example AAB)
`subclone.genotype`	If existing, the genotype of the subclone. Else NULL
`subclone.cell`	The cellular prevalence of the subclone which has a different Copy Number at this site
`contamination`	The fraction of normal cells in the sample
`restrict.to.AB`	Should the analysis keep only sites located in A and AB sites in all samples?
`force.single.copy`	Should all mutations in overdiploid regions set to single copy? Default is FALSE

`Cellular_preclustering` *Preclustering method*

Description

This method clusters mutations based on the probability that they are from the same distribution. It first computes the zscore associated with a normalized number of alternative reads and depth. The "normalized" number of reads is the number of alternative reads expected if the mutation was at a single copy in a diploid genome.

Usage

```
Cellular_preclustering(Schrod_cells)
```

Arguments

`Schrod_cells`	The classic output from Schrodinger function

Value

returns a list with:

similarityMatrix The matrix of probabilities

distance The dissimilarity matrix

tree The tree obtained by hierachical clustering of the dissimilarity matrix using "ward.D2" method

check_leaf *Check created leaf*

Description

Checks that created leaf has cellularity >1 existing clone in at least one sample, and that cellularities in all samples are greater than or equal to 0.

Usage

```
check_leaf(new_leaf, Proportions_mutated)
```

Arguments

new_leaf	A numeric vector to be added
Proportions_mutated	Matrix with samples in columns, clones (carrying mutations) in rows.

check_split *Check*

Description

Check if node can be split further, i.e. cellularity > 10

Usage

```
check_split(leaf)
```

Arguments

leaf	Numeric vector

Cluster_plot_from_cell *Cellularity clustering*

Description

Clustering cellularities based on the most likely presence of a clone, using the pamk algorithm (fpc package). Clustering can be guided by toggling manual_clustering on and/or giving a range of number of clusters.

Usage

```
Cluster_plot_from_cell(Cell, Sample_names, simulated, save_plot = TRUE,
  contamination, clone_priors, prior_weight, nclone_range, Initializations,
  preclustering = TRUE, epsilon = 5 * (10^(-3)), ncores = 2,
  output_directory = NULL, model.selection = "BIC", optim = "default",
  keep.all.models = FALSE)
```

Arguments

Cell	Output from Return_one_cell_by_mut, list of cellularities (one list-element per sample)
Sample_names	Name of the sample
simulated	Was the data generated by QuantumCat?
save_plot	Should the clustering plots be saved? Default is True
contamination	The fraction of normal cells in the samples
clone_priors	If known a list of priors (cell prevalence) to be used in the clustering
prior_weight	If known a list of priors (fraction of mutations in a clone) to be used in the clustering
nclone_range	Number of clusters to look for
Initializations	Maximal number of independant initial condition tests to be tried
preclustering	The type of preclustering used for priors: "Flash","kmedoid" or NULL. NULL will generate centers using uniform distribution.
epsilon	Stop value: maximal admitted value of the difference in cluster position and weights between two optimization steps.
ncores	Number of CPUs to be used
output_directory	Directory in which to save results
model.selection	The function to minimize for the model selection: can be "AIC", "BIC", or numeric. In numeric, the function uses a variant of the BIC by multiplication of the k*ln(n) factor. If >1, it will select models with lower complexity.

optim	use L-BFS-G optimization from R ("default"), or from optimx ("optimx"), or Differential Evolution ("DEoptim")
keep.all.models	Should the function output the best model (default; FALSE), or all models tested (if set to true)

Compute_NMI *Normalized Mutual Information*

Description

Compute the normalized mutual information to assess clustering quality

Usage

```
Compute_NMI(QC_out)
```

Arguments

QC_out	output from QuantumClone clustering

Examples

```
Compute_NMI(QC_output)
```

Compute_objective *Compute value of objective function*

Description

Compute the value of clustering based on same principles as QuantumClone EM

Usage

```
Compute_objective(tree, nclus, Schrod, conta)
```

Arguments

tree	Tree from hierarchical clustering
nclus	Number of clusters used for cutting (numeric of length 1)
Schrod	Output from Schrodinger cellularities
conta	Numeric value of contamination fraction in each sample

create_priors *Clonal fraction prior creation*

Description

Semi-random generation of clonal priors

Usage

```
create_priors(nclust, nsample, prior = NULL)
```

Arguments

nclust	Number of clones to look for.
nsample	Number of samples
prior	Possible priors known (the position of each element in a list corresponds to 1 clone)

Create_prior_cutTree *Create priors from hierarchical clustering*

Description

Creates weights and position priors from the hierachical clustering (tree) given a number of clusters Nclust. The centers of each cluster is found by $Center = \frac{\sum_{m \in cluster} NormalizedAlt_m}{\sum_{m \in cluster} NormalizedAlt_m}$

Usage

```
Create_prior_cutTree(tree, Schrod_cells, NClus, jitter = FALSE)
```

Arguments

tree	The tree generated by Cellular_preclustering
Schrod_cells	The classic output from Schrodinger function
NClus	the number of clusters to cut the data
jitter	Should it jitter weights and centers around values found?

Value

returns a list with:

weigths The proportion of mutations in each cluster

centers A list with a numeric vector for each sample, with the cellularity of each cluster

@importFrom stats cutree

e.step *Expectation step calculation*

Description

Expectation step calculation

Usage

```
e.step(Schrod, centers, weights, adj.factor)
```

Arguments

Schrod	A list of dataframes (one for each sample), generated by the Patient_schrodinger_cellularities() function.
centers	Coordinates of the clones: a list of numeric vectors (1 per sample), with coordinates between 0 and 1.
weights	Proportion of mutation in a clone
adj.factor	Factor to compute the probability: makes transition between the cellularity of the clone and the frequency observed

EM.algo *Expectation Maximization algorithm*

Description

Optimization of clone positions and proportion of mutations in each clone.

Usage

```
EM.algo(Schrod, nclust = NULL, prior_center = NULL, prior_weight = NULL,
  contamination, epsilon = 10^(-2), optim = "default")
```

Arguments

Schrod	A list of dataframes (one for each sample), generated by the Patient_schrodinger_cellularities() function.
nclust	Number of clones to look for (mandatory if prior_center or prior_weight are null)
prior_center	Clone coordinates (from another analysis) to be used
prior_weight	Prior on the fraction of mutation in each clone
contamination	Numeric vector with the fraction of normal cells contaminating the sample
epsilon	Stop value: maximal admitted value of the difference in cluster position and weights between two optimization steps. If NULL, will take 1/(median depth).
optim	use L-BFS-G optimization from R ("default"), or from optimx ("optimx")

EM_clustering *Expectation Maximization*

Description

Maximization of the likelihood given a mixture of binomial distributions

Usage

```
EM_clustering(Schrod, contamination, prior_weight = NULL,
  clone_priors = NULL, Initializations = 1, nclone_range = 2:5,
  epsilon = 0.01, ncores = 2, model.selection = "BIC",
  optim = "default", keep.all.models = FALSE, FLASH = FALSE)
```

Arguments

Schrod	List of dataframes, output of the Schrodinger function or the EM algorithm
contamination	The fraction of normal cells in the sample
prior_weight	If known a list of priors (fraction of mutations in a clone) to be used in the clustering
clone_priors	If known a list of priors (cell prevalence) to be used in the clustering
Initializations	Maximal number of independant initial condition tests to be tried
nclone_range	Number of clusters to look for
epsilon	Stop value: maximal admitted value of the difference in cluster position and weights between two optimization steps.
ncores	Number of CPUs to be used
model.selection	The function to minimize for the model selection: can be "AIC", "BIC", or numeric. In numeric, the function uses a variant of the BIC by multiplication of the k*ln(n) factor. If >1, it will select models with lower complexity.
optim	use L-BFS-G optimization from R ("default"), or from optimx ("optimx"), or Differential Evolution ("DEoptim")
keep.all.models	Should the function output the best model (default; FALSE), or all models tested (if set to true)
FLASH	should it use FLASH algorithm to create priors

eval.fik.m *Eval probability for M step Computes the log directly as log density is faster to compute*

Description

Eval probability for M step Computes the log directly as log density is faster to compute

Usage

```
eval.fik.m(Schrod, centers, weights, adj.factor, log = TRUE)
```

Arguments

Schrod	The shcrodinger list of matrices
centers	centers of the clusters
weights	weight of each cluster
adj.factor	The adjusting factor, taking into account contamination, copy number, number of copies
log	Should it compute the log distribution (TRUE) or probability (FALSE) between two optimization steps. If NULL, will take 1/(median depth).

evolution_plot *Evolution plot*

Description

Plots evolution in time of clones

Usage

```
evolution_plot(QC_out, Sample_names = NULL)
```

Arguments

QC_out	: Output from One_step_clustering
Sample_names	: character vector of the names of each sample (in the same order as the data)

Examples

```
require(ggplot2)
evolution_plot(QC_output)
```

`filter_on_fik` *Data filter*

Description

Keep one possibility per position and ajust weight accordingly

Usage

```
filter_on_fik(Schrod, fik)
```

Arguments

`Schrod`	A list of dataframes (one for each sample), generated by the Patient_schrodinger_cellularities.
`fik`	matrix of probability of each possibility to belong to a clone

`find_x_position` *Graphic position*

Description

Computes the position of a node on the graph, based on the interaction matrix.

Usage

```
find_x_position(matrix, n, d)
```

Arguments

`matrix`	The interaction matrix of the tree (1 on the i-th row j-th column means "clone j is the progeny of clone i")
`n`	Index of the clone of interest in the matrix
`d`	Initial number of clones

FlashQC *Flash QuantumClone*

Description

Fast method to find clones without filtering for multiple states

Usage

```
FlashQC(Cells, conta, Nclus, model.selection = "tree")
```

Arguments

Cells	Input for QuantumClone with genotype required
conta	vector with contamination fraction in each sample
Nclus	vector with the number of clusters to test (alternatively only min and max values)
model.selection	One of "tree", "AIC", "BIC" or numeric. "tree" will use "ccc","ch" and "gap" methods from NbClust to determine the number of clusters. "BIC","AIC" or numeric values will use methods from QuantumClone.

See Also

QuantumClone

Examples

```
set.seed(123)
#1: Cluster data
In<-QuantumClone::Input_Example
FQC<-FlashQC(In,conta = c(0,0),Nclus = 2:10)

#2: Get order variants by clones:
ord<-order(In[[1]]$Chr)
#3: Visualize clustering:
image(
 1:nrow(In[[1]]),
 1:nrow(In[[1]]),
 FQC$similarity[ord,ord],
 xlab="", ylab="")
#4: add limit of real clusters:
abline(h = cumsum(table(In[[1]]$Chr[ord]))+1)
abline(v = cumsum(table(In[[1]]$Chr[ord]))+1)

#5: alternatively add clusters found:
ord<-order(FQC$cluster)
image(
 1:nrow(In[[1]]),
```

```
 1:nrow(In[[1]]),
 FQC$similarity[ord,ord],
 xlab="", ylab="")
abline(h = cumsum(table(FQC$cluster[ord]))+1)
abline(v = cumsum(table(FQC$cluster[ord]))+1)
# Show clustering quality:
NMI_cutree( FQC$cluster,chr = In[[1]]$Chr)
```

FLASH_main *Flash core*

Description

Returns number of clusters based on model selection

Usage

```
FLASH_main(Schrod_cells, model.selection, conta, Nclus, tree = NULL,
  dissimMatrix = NULL)
```

Arguments

Schrod_cells	Output from Schrodinger cellularities
model.selection	One of "tree", "AIC", "BIC" or numeric. "tree" will use "ccc","ch" and "gap" methods from NbClust to determine the number of clusters. "BIC","AIC" or numeric values will use methods from QuantumClone.
conta	vector with contamination fraction in each sample
Nclus	vector with the number of clusters to test (alternatively only min and max values)
tree	Hierarchical tree from hclust
dissimMatrix	Dissimilarity matrix, required if model selection is "tree"

From_freq_to_cell *Wrap-up function*

Description

Function that computes the most likely position for each mutation based on the genotype

Usage

```
From_freq_to_cell(SNV_list, FREEC_list = NULL, Sample_names,
  Genotype_provided = FALSE, save_plot = TRUE, contamination, ncores = 4,
  restrict.to.AB = FALSE, output_directory = NULL,
  force.single.copy = FALSE)
```

Arguments

SNV_list	A list of dataframes (one for each sample), with as columns : (for the first column of the first sample the name of the sample), the chromosome "Chr",the position of the mutation "Start", the number of reads supporting the mutation "Alt", the depth of coverage at this locus "Depth", and if the output from FREEC for the samples are not associated, the genotype "Genotype".
FREEC_list	list of dataframes from FREEC for each samples (usually named Sample_ratio.txt), in the same order as SNV_list
Sample_names	Name of the samples
Genotype_provided	If the FREEC_list is provided, then should be FALSE (default), otherwise TRUE
save_plot	Should the plots be saved? Default is TRUE
contamination	Numeric vector describind the contamination in all samples (ranging from 0 to 1). Default is 0. No longer used for clustering.
ncores	Number of cores to be used during EM algorithm
restrict.to.AB	Should the analysis keep only sites located in A and AB sites in all samples?
output_directory	Directory in which to save results
force.single.copy	Should all mutations in overdiploid regions set to single copy? Default is FALSE

FullEM *Expectation Maximization algorithm*

Description

Optimization of clone positions and proportion of mutations in each clone followed by filtering on most likely possibility for each mutation and a re-optimization.

Usage

```
FullEM(Schrod, nclust, prior_center, prior_weight = NULL, contamination,
  epsilon = 5 * 10^(-3), optim = "default")
```

Arguments

Schrod	A list of dataframes (one for each sample), generated by the Patient_schrodinger_cellularities() function.
nclust	Number of clones to look for (mandatory if prior_center or prior_weight are null)
prior_center	Clone coordinates (from another analysis) to be used
prior_weight	Prior on the fraction of mutation in each clone
contamination	Numeric vector with the fraction of normal cells contaminating the sample

epsilon	Stopping condition for the algorithm: what is the minimal tolerated difference of position or weighted between two steps
optim	use L-BFS-G optimization from R ("default"), or from optimx ("optimx"), or Differential Evolution ("DEoptim")

---

grbase *Computes gradient of function*

Description

Computes gradient of function

Usage

```
grbase(fik, adj.factor, centers, Alt, Depth)
```

Arguments

fik	Evaluation of fik for previous iteration
adj.factor	Factor to compute the probability: makes transition between the cellularity of the clone and the frequency observed
centers	vector with cellularity of each clone (numeric vector, ordered by samples)
Alt	Matrix with number of draws in rows for a mutation/possibility, and samples in columns
Depth	Matrix with number of not draws (Depth - Alt) in rows for a mutation/possibility, and samples in columns

Examples

```
fik<-matrix(c(1,0,0,1),nrow = 2)
adj.factor<-matrix(1/2,nrow =2 ,ncol =1)
centers<-c(0.25,0.75)
Alt<-c(125,375)
Depth<-c(1000,1000)
grbase(fik,adj.factor,centers,Alt,Depth)
```

grzero *Gradient 0*

Description

Return center values for max if adj.factor has a single value for all variants/possibilities in each samples

Usage

```
grzero(fik, adj.factor, Alt, Depth)
```

Arguments

fik	matrix with probability of each possibility to belong to clone k
adj.factor	matrix with coefficient making transition between cellularity and frequency
Alt	matrix with samples in columns and number of alternative reads in rows
Depth	matrix with samples in coluns and depth of coverage in rows

hard.clustering *Hard clustering based on EM output*

Description

Attributes a mutation to its most likely clone based on the output of the EM algorithm

Usage

```
hard.clustering(EM_out)
```

Arguments

EM_out	Output from EM.algo or FullEM

Input_Example *Example generated by QuantumCat*

Description

A dataset generated by: Input_Example<-QuantumCat(number_of_clones = 4, number_of_mutations = 100, ploidy = "AB",depth = 150, number_of_samples = 2, contamination = c(0,0))

Usage

```
Input_Example
```

Format

A list of dataframes

SampleName First column containing the name of the sample

Chr The chromosome either 1 2 ... X Y or chr1 chr2 ... chrY

Start The genomic position of the variant

Alt Number of reads supporting variant allele

Depth Total number of reads (reference + alternative allele) at position

is_included *Group theory*

Description

Clone2 is included in Clone1 if all values of Clone2 are lower or equal to the ones in Clone1 at the same position. Returns TRUE is Clone2 is included in Clone1.

Usage

```
is_included(Clone1, Clone2)
```

Arguments

Clone1	Numeric vector, representing the cellularity of Clone1 in different samples
Clone2	Numeric vector, representing the cellularity of Clone2 in different samples

list_prod *List product*

Description

Returns the product of all elements in a list, e.g. a vector if the elements of the list are vectors, etc.

Usage

```
list_prod(L, col = NULL)
```

Arguments

L	list used
col	If it is a list of matrices, and only one column should be used, name of the column.

Examples

```
list_prod(list(matrix(1:4,nrow = 2),matrix(1:4,nrow = 2)))
```

longueur *Length*

Description

Computes the length from the clone on the n-th row of the matrix, to the most ancestral clone

Usage

```
longueur(matrix, n)
```

Arguments

matrix	The interaction matrix of the tree (1 on the i-th row j-th column means "clone j is the progeny of clone i")
n	Index of the clone in the matrix

m.step *Maximization step*

Description

Optimization of clone positions and proportion of mutations in each clone, based on the previously calculated expectation

Usage

```
m.step(fik, Schrod, previous.weights, previous.centers, contamination,
  adj.factor, optim = "default")
```

Arguments

fik	Matrix giving the probability of each mutation to belong to a specific clone
Schrod	A list of dataframes (one for each sample), generated by the Patient_schrodinger_cellularities() function.
previous.weights	Weights from the previous optimization step (used as priors for this step)
previous.centers	Clone coordinates from previous optimization step (used as priors for this step)
contamination	Numeric vector with the fraction of normal cells contaminating the sample
adj.factor	Factor to compute the probability: makes transition between the cellularity of the clone and the frequency observed
optim	use L-BFS-G optimization from R ("default"), or from optimx ("optimx"), or Differential Evolution ("DEoptim")

MajorityVote *Majority vote*

Description

Extract majority vote from multiple indices

Usage

```
MajorityVote(index)
```

Arguments

index	vector with number of clusters selected by indices

Value

Numeric value of the number of clusters to chose

manual_plot_trees *Plot tree*

Description

Creates a visual output for the phylogeny created by Tree_generation()

Usage

```
manual_plot_trees(connexion_list, d, cex = 0.8, p)
```

Arguments

connexion_list	Data frame of the concatenation of the interaction matrix and the cellularity of each clone at different time points.
d	Number of clusters found by QuantumClone
cex	Coefficient of expansion for the texts in phylogenetic tree plots. Default is 0.8
p	Probability of a tree

Examples

```
# Extract one tree out of the 3 available trees:
Example_tree<-QuantumClone::Tree[[1]]
manual_plot_trees(Example_tree[[1]], d= 4,p = Example_tree[[2]])
```

multiplot_trees *Plots multiple trees*

Description

Plots all trees created by the function Tree_generation. The red line means that mutations occured.

Usage

```
multiplot_trees(result_list, d, cex = 0.8)
```

Arguments

result_list	List of lists (tree generated and the probability associated with each tree)
d	Number of clusters found by QuantumClone
cex	Coefficient of expansion for the texts in phylogenetic tree plots. Default is 0.8

Examples

```
multiplot_trees(QuantumClone::Tree, d= 4)
```

NMI_cutree *NMI*

Description

Computes the NMI based on the clustering

Usage

```
NMI_cutree(cut_tree, chr)
```

Arguments

cut_tree	a numeric vector of cluster selection
chr	the ground truth for clusters

Value

numeric value of NMI (between 0 and 1)

Examples

```
set.seed(123)
#1: Cluster data
FQC<-FlashQC(QuantumClone::Input_Example,conta = c(0,0),Nclus = 2:10)

#2: Compute NMI
NMI_cutree(FQC$cluster,chr = QuantumClone::Input_Example[[1]]$Chr)
```

One_D_plot *Plots*

Description

Creates density plot when only one sample is given

Usage

```
One_D_plot(EM_out, contamination)
```

Arguments

EM_out	output from the EM algorithm
contamination	Numeric vector giving the proportion of normal cells in each samples

`One_step_clustering` *Cellularity clustering*

Description

Wrap up function that clusters cellularities. This is based on the most likely possibility for each mutation, give ints frequency and genotype.

Usage

```
One_step_clustering(SNV_list, FREEC_list = NULL, contamination,
  nclone_range = 2:5, clone_priors = NULL, prior_weight = NULL,
  Initializations = 1, preclustering = "FLASH", simulated = FALSE,
  epsilon = NULL, save_plot = TRUE, ncores = 1, restrict.to.AB = FALSE,
  output_directory = NULL, model.selection = "BIC", optim = "default",
  keep.all.models = FALSE, force.single.copy = FALSE)
```

Arguments

`SNV_list`	A list of dataframes (one for each sample), with as columns : (for the first column of the first sample the name of the sample), the chromosome "Chr",the position of the mutation "Start", the number of reads supporting variant "Alt", as well as the total number of reads overlapping position "Depth", and if the output from FREEC for the samples are not associated, the genotype "Genotype".
`FREEC_list`	list of dataframes from FREEC for each samples (usually named Sample_ratio.txt), in the same order as SNV_list
`contamination`	Numeric vector describind the contamination in all samples (ranging from 0 to 1). Default is 0. No longer used for clustering.
`nclone_range`	A number or range of clusters that should be used for clustering
`clone_priors`	List of vectors with the putated position of clones
`prior_weight`	Numeric with the proportion mutations in each clone
`Initializations`	Number of initial conditions to be tested for the EM algorithm
`preclustering`	The type of preclustering used for priors: "Flash","kmedoid" or NULL. NULL will generate centers using uniform distribution. WARNING: overrides priors given
`simulated`	Should be TRUE if the data has been been generated by the QuantumCat algorithm
`epsilon`	Stop value: maximal admitted value of the difference in cluster position and weights between two optimization steps. If NULL, will take 1/(average depth)
`save_plot`	Should the plots be saved? Default is TRUE
`ncores`	Number of cores to be used during EM algorithm
`restrict.to.AB`	Boolean: Should the analysis keep only sites located in A and AB sites in all samples?

output_directory
: Directory in which to save results

model.selection
: The function to minimize for the model selection: can be "AIC", "BIC", or numeric. In numeric, the function uses a variant of the BIC by multiplication of the k*ln(n) factor. If >1, it will select models with lower complexity.

optim
: use L-BFS-G optimization from R ("default"), or from optimx ("optimx"), or Differential Evolution ("DEoptim")

keep.all.models
: Should the function output the best model (default; FALSE), or all models tested (if set to true)

force.single.copy
: Should all mutations in overdiploid regions set to single copy? Default is FALSE

Examples

```
Mutations<-QuantumClone::Input_Example
 for(i in 1:2){
 Mutations[[i]]<-cbind(rep(paste("Example_",i,sep=""),times=10),Mutations[[i]])
 colnames(Mutations[[i]])[1]<-"Sample"
}
print("The data should look like this:")
print(head(Mutations[[1]]))

cat("Cluster data: will try to cluster between 3 and 4 clones, with 1 maximum search each time,
      and will use priors from preclustering (e.g. k-medoids on A and AB sites)")
print("The genotype is provided in the list frame, and
          there is no associated data from FREEC to get genotype from.")
print("The computation will run on a single CPU.")
Clustering_output<-One_step_clustering(SNV_list = Mutations,
FREEC_list = NULL,contamination = c(0,0),nclone_range = 3:4,
clone_priors = NULL,prior_weight = NULL ,
Initializations = 1,preclustering = "FLASH", simulated = TRUE,
save_plot = TRUE,ncores=1,output_directory="Example")
print("The data can be accessed by Clustering_output$filtered_data")
print("All plots are now saved in the working directory")
```

---

parallelEM *Expectation Maximization algorithm*

Description

Optimization of clone positions and proportion of mutations in each clone followed by filtering on most likely possibility for each mutation and a re-optimization. Then gives out the possibility with maximal likelihood Relies on foreach

Usage

```
parallelEM(Schrod, nclust, epsilon, contamination, prior_center = NULL,
  prior_weight = NULL, Initializations = 1, optim = "default",
  keep.all.models = FALSE)
```

Arguments

Schrod	A list of dataframes (one for each sample), generated by the Patient_schrodinger_cellularities() function.
nclust	Number of clones to look for (mandatory if prior_center or prior_weight are null)
epsilon	Stopping condition for the algorithm: what is the minimal tolerated difference of position or weighted between two steps
contamination	Numeric vector with the fraction of normal cells contaminating the sample
prior_center	Clone coordinates (from another analysis) to be used
prior_weight	Prior on the fraction of mutation in each clone
Initializations	Maximal number of independant initial condition tests to be tried
optim	use L-BFS-G optimization from R ("default"), or from optimx ("optimx"), or Differential Evolution ("DEoptim")
keep.all.models	Should the function output the best model (default; FALSE), or all models tested (if set to true)

Patient_schrodinger_cellularities

Patient Schrodinger Cellularities

Description

Computes all possible cellularities for all mutations across all samples. Calls CellularitiesFromFreq on all mutations to evaluate all possibilities

Usage

```
Patient_schrodinger_cellularities(SNV_list, FREEC_list = NULL,
  Genotype_provided = FALSE, contamination, restrict.to.AB = FALSE,
  force.single.copy = FALSE)
```

Arguments

SNV_list	A list of dataframes (one for each sample), with as columns : (for the first column of the first sample the name of the sample), the chromosome "Chr",the position of the mutation "Start", the number of reads supporting the mutation "Alt", the depth of coverage at this locus "Depth", and if the output from FREEC for the samples are not associated, the genotype "Genotype".
FREEC_list	list of dataframes from FREEC for each samples (usually named Sample_ratio.txt), in the same order as SNV_list
Genotype_provided	If the FREEC_list is provided, then should be FALSE (default), otherwise TRUE
contamination	Numeric vector describind the contamination in all samples (ranging from 0 to 1). Default is 0. No longer used for clustering.
restrict.to.AB	Should the analysis keep only sites located in A and AB sites in all samples?
force.single.copy	Should all mutations in overdiploid regions set to single copy? Default is FALSE

---

phylo_tree_generation *Data generation*

Description

Creates a phylogenetic tree on simple assumptions: 1. There is a common ancestor to all clones 2. Each clone creates a partition of the space. One node will carry mutations, the other will not. A node that is not mutated can be partitioned. 3. Nodes that have less than 2 less than 1 4. Nodes that have less than 10 WARNING: Tree_generation recreates a tree from data while phylo_tree_generation randomly creates a phylogeny

Usage

```
phylo_tree_generation(number_of_clones, number_of_samples)
```

Arguments

number_of_clones	The wanted number of observable clones (meaning bearing at least 1 mutation)
number_of_samples	The number of samples on which the data should be simulated

plot_cell_from_Return_out *Plot cellularity*

Description

2D plot of cellularity based on the output of the EM

Usage

```
plot_cell_from_Return_out(lis, Sample_names, output_dir = NULL)
```

Arguments

lis	Output from Return_one_cell_by_mut, list of cellularities (one list-element per sample)
Sample_names	Name of the samples.
output_dir	Directory in which to save plots

plot_QC_out *Plot QC_output*

Description

This function was implemented to re-plot easily the diagrams of clonality for changes/enhancement. Returns a ggplot object Uses ggplot2 package

Usage

```
plot_QC_out(QClone_Output, Sample_names = NULL, simulated = FALSE,
  sample_selected = 1:2)
```

Arguments

QClone_Output	Output from QuantumClone algorithm
Sample_names	: character vector of the names of each sample (in the same order as the data)
simulated	Was the data generated by QuantumCat?
sample_selected	: number of the sample to be considered for plot (can be 1 or 2 samples)

Examples

```
require(ggplot2)
message("Using preclustered data:")
QC_out<-QuantumClone::QC_output
plot_QC_out(QC_out,Sample_names = c("Diagnosis","Relapse"))
```

plot_with_margins_densities *Plot with margin densities*

Description

Adapted from http://stackoverflow.com/questions/11883844/inserting-a-table-under-the-legend-in-a-ggplot2-histogram Uses gridExtra package

Usage

```
plot_with_margins_densities(QClone_Output)
```

Arguments

QClone_Output	Output from QuantumClone algorithm

Examples

```
require(ggplot2)
require(gridExtra)
message("Using preclustered data:")
QC_out<-QuantumClone::QC_output
plot_with_margins_densities(QC_out)
```

Precision_Recall *Precision*

Description

Computes the precision based on the clustering

Usage

```
Precision_Recall(hx, Truth)
```

Arguments

hx	a numeric vector of cluster selection
Truth	the ground truth for clusters

Value

TP The number of true positive links

TN The number of true negative links

FP The number of false positive links

FN The number of false negative links

Pr The precision, defined by $Pr = \frac{TP}{TP+FP}$

R The recall, defined by $R = \frac{TP}{TP+FN}$

F1 The F1 index, defined by $F1 = \frac{2 \times P \times R}{P+R}$

RI Rand Index, defined by $RI = \frac{TP+TN}{TP+TN+FP+FN}$

validat Is positives + negatives equal to total number of links - returns absolute difference if false

Examples

```
set.seed(123)
#1: Cluster data
FQC<-FlashQC(QuantumClone::Input_Example,conta = c(0,0),Nclus = 2:10)

#2: Compute NMI
Precision_Recall(hx = FQC$cluster,Truth = QuantumClone::Input_Example[[1]]$Chr)

### From Stanford NLP example:
cluster<-c(rep(1,6),rep(2,6),rep(3,5))
truth<-c(rep(1,5),2,
         1,rep(2,4),3,
         rep(1,2),rep(3,3))
Precision_Recall(cluster,truth)
```

Probability.to.belong.to.clone

Probability

Description

Returns dataframe with all informations about mutation (Number of copies, Cellularity, etc.) and probability to belong to a clone

Usage

```
Probability.to.belong.to.clone(SNV_list, clone_prevalence, contamination,
  clone_weights = NULL)
```

Arguments

SNV_list	A list of dataframes (one for each sample), with as columns : (for the first column of the first sample the name of the sample), the chromosome "Chr",the position of the mutation "Start", the number of reads supporting the mutation "Alt", the depth of coverage at this locus "Depth", and if the output from FREEC for the samples are not associated, the genotype "Genotype".
clone_prevalence	List of numeric vectors giving the cellular prevalence of each clone in each sample, not normalized for contamination. This should be stored in 'QC_output$EM.output$centers'
contamination	Numeric vector giving the contamination by normal cells
clone_weights	Numeric vector giving the proportion of mutations in each clone

Examples

```
set.seed(123)
SNVs<-QuantumCat(number_of_clones = 2,number_of_mutations = 50,number_of_samples = 1,ploidy = "AB")
Probability.to.belong.to.clone(SNV_list=SNVs,
clone_prevalence=list(c(0.5,1),c(0.5,1)),contamination=c(0,0))
```

ProbDistMatrix *Distance*

Description

Creates a matrix of distance between two points based on the p-value to be from the same distribution

Usage

```
ProbDistMatrix(Schrod_cells)
```

Arguments

Schrod_cells	The classic output from Schrodinger function, with the normalized Alt

Value

returns a square numeric matrix

QC_output *Example of output generated by clustering of Input_Example*

Description

Clustering output generated by: QC_output<-One_step_clustering(SNV_list = Input_Example,contamination = c(0,0), nclone_range = 2:5,maxit = 1, save_plot = FALSE,ncores = 1, epsilon = 0.01) On October, 30th 2015

Usage

```
QC_output
```

Format

list of lists

EMOutput Results of Expectation maximization: *fik: probability of a mutation (line) to belong to clone k (column) *weights: proportion of mutations belonging to clone k *centers: cellularity of clone k in sample j *val: value of log-likelihood

filtered.data input data without mutations removed (missing genotype or genotype not AB if filtered on AB)

cluster Result of hard clustering: each mutation is attributed to a single clone

QuantumCat *Data generation*

Description

Creates plausible data as would be oserved by genome sequencing

Usage

```
QuantumCat(number_of_clones, number_of_mutations, ploidy = 2, depth = 100,
  number_of_samples = 2, Random_clones = F, contamination = NULL,
  Subclonal.CNA.fraction = NULL)
```

Arguments

`number_of_clones`
: The wanted number of observable clones (meaning bearing at least 1 mutation)

`number_of_mutations`
: The total observed number of mutations (across all clones)

`ploidy`
: The general ploidy of the tumor. Default is 2. If "disomic" : only AB regions will be generated.

depth The depth of sequencing (does not account for contamination). Default is 100x

number_of_samples
The number of samples on which the data should be simulated. Default is 2.

Random_clones Should the number of clones be generated randomly (sample(1:10))

contamination A numeric vector indicating the fraction of normal cells in each sample.

Subclonal.CNA.fraction
Cell fraction of the subclone that has subclonal CNA

Examples

```
print("Generate small set of mutations from 2 differents clones...")
print("...in 1 sample, contaminated at 10% by normal cells")

QuantumCat(number_of_clones=2,number_of_mutations=50,number_of_samples=1,contamination=0.1)
```

QuantumClone *One step analysis function*

Description

Sequentially calls a function to test all accessible cellularities for all mutations in the samples,then cluster them, and finally draws phylogenetic trees based on the uncovered cellularities

Usage

```
QuantumClone(SNV_list, FREEC_list = NULL, contamination, nclone_range = 2:5,
  clone_priors = NULL, prior_weight = NULL, simulated = FALSE,
  save_plot = TRUE, epsilon = NULL, Initializations = 2,
  preclustering = "FLASH", timepoints = NULL, ncores = 1,
  output_directory = NULL, model.selection = "BIC", optim = "default",
  keep.all.models = FALSE, force.single.copy = FALSE)
```

Arguments

SNV_list A list of dataframes (one for each sample), with as columns : (for the first column of the first sample the name of the sample), the chromosome "Chr",the position of the mutation "Start", the number of reads supporting the mutation "Alt", the depth of coverage at this locus "Depth", and if the output from FREEC for the samples are not associated, the genotype "Genotype".

FREEC_list list of dataframes from FREEC for each samples (usually named Sample_ratio.txt), in the same order as SNV_list

contamination Numeric vector describind the contamination in all samples (ranging from 0 to 1). Default is 0. No longer used for clustering.

nclone_range A number or range of clusters that should be used for clustering

clone_priors List of vectors with the putated position of clones

prior_weight	Numeric with the proportion mutations in each clone
simulated	Should be TRUE if the data has been been generated by the QuantumCat algorithm
save_plot	Should the plots be saved? Default is TRUE
epsilon	Stop value: maximal admitted value of the difference in cluster position and weights between two optimization steps. If NULL, will take 1/(average depth).
Initializations	Number of initial conditions to be tested for the EM algorithm
preclustering	The type of preclustering used for priors: "Flash","kmedoid" or NULL. NULL will generate centers using uniform distribution. WARNING: overrides priors given
timepoints	a numeric vector indicating if the samples are from different timepoints or tumors (e.g. one tumor and metastates) If NULL, all samples are considered from the same tumor.
ncores	Number of cores to be used during EM algorithm
output_directory	Path to output directory
model.selection	The function to minimize for the model selection: can be "AIC", "BIC", or numeric. In numeric, the function uses a variant of the BIC by multiplication of the k*ln(n) factor. If >1, it will select models with lower complexity.
optim	use L-BFS-G optimization from R, with exact gradient, ("default"), or L-BFS-G with numerical gradient computation from optimx ("optimx"), or Differential Evolution ("DEoptim"), or a mixture of EM with exact center computation (if fully diploid), or "optim" if this fails ("compound"). Note that DEoptim is the only one that does not use EM algorithm
keep.all.models	Should the function output the best model (default; FALSE), or all models tested (if set to true)
force.single.copy	Should all mutations in overdiploid regions set to single copy? Default is FALSE

Examples

```
Mutations<-QuantumClone::Input_Example
 for(i in 1:2){
 Mutations[[i]]<-cbind(rep(paste("Example_",i,sep=""),times=10),Mutations[[i]])
 colnames(Mutations[[i]])[1]<-"Sample"
}
print("The data should look like this:")
print(head(Mutations[[1]]))

cat("Cluster data: will try to cluster between 3 and 4 clones, with 1 maximum search each time,
      and will use priors from preclustering.")
print("The genotype is provided in the list frame, and
          there is no associated data from FREEC to get genotype from.")
print("The computation will run on a single CPU.")
```

```
Clustering_output<-QuantumClone(SNV_list = Mutations,
FREEC_list = NULL,contamination = c(0,0),nclone_range = 3:4,
clone_priors = NULL,prior_weight = NULL ,
Initializations = 1,preclustering = "FLASH", simulated = TRUE,
save_plot = TRUE,ncores=1,output_directory="Example")
print("The data can be accessed by Clustering_output$filtered_data")
print("All plots are now saved in the working directory")
```

`strcount` *String count*

Description

Counting the number of characters for each element of a vector

Usage

```
strcount(x, pattern = "", split = "")
```

Arguments

`x`	The vector from which elements should be counted
`pattern`	Pattern to be recognized. Default is "
`split`	Pattern used to split elements of the vector. Default is "

`Tidy_output` *Tidying output from EM*

Description

Tidying input by Chr Start

Usage

```
Tidy_output(r, Genotype_provided, SNV_list)
```

Arguments

`r`	output from Cluster_plot_from_cell
`Genotype_provided`	If the FREEC_list is provided, then should be FALSE (default), otherwise TRUE
`SNV_list`	A list of dataframes (one for each sample), with as columns : (for the first column of the first sample the name of the sample),

Tree — *Example of output by Tree_generation*

Description

Reconstruction of the tumor phylogenetic tree Generated by : Tree<-Tree_generation(cbind(QuantumClone::QC_output$EM
QuantumClone::QC_output$EM.output$centers[[2]]))

Usage

```
Tree
```

Format

list of lists

Top level List of all possibilities

dataframe Table of hierarchical relations

numeric probability of this tree

Tree_generation — *Phylogenetic tree*

Description

Generates a list of possible trees based on the cellularity of each clone, and the spatial and temporal distribution of the samples. Assumption is made the different clones are on different lines of the matrix

Usage

```
Tree_generation(Clone_cellularities, timepoints = NULL)
```

Arguments

`Clone_cellularities`	A dataframe with cellularities (ranging from 0 to 1) of each clone (rows) in each sample (columns)
`timepoints`	A numeric vector giving the spatial and/or temporal distribution of the samples

update_probs *Update proportions*

Description

Creates vector with probability to be sampled

Usage

```
update_probs(Proportions, can_split)
```

Arguments

Proportions	numeric matrix
can_split	logical vector

zscore *Z-score*

Description

Computes the z-score of mutations being from the same distribution

Usage

```
zscore(Depth, Alt)
```

Arguments

Depth	a numeric vector of depth of sequencing for each variant
Alt	a numeric vector of the number of reads supporting each variant

Value

returns a square numeric matrix

Index