# Package 'CloneStrat'

March 11, 2020

**Title** Multi-regional clonal deconvolution of tumor sequencing data

**Version** 0.1.1

**Description** Functions to deconvolute clones and sub-clones in multi-regional/temporal massive parallel DNA sequencing of solid tumor in presence of microarray based copy number profiles. Additional functions include estimation of said copy number profiles from exome sequencing.

**Depends** R (¿= 3.5.0)

**URL** https://https://github.com/Subhayan18/CloneStrat/

**BugReports** https://github.com/Subhayan18/CloneStrat/issues/

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Imports** readxl, mclust, fpc, KODAMA, dplyr, ggplot2, rlang, sequenza, vcfR, bootcluster, factoextra, FactoMineR

**NeedsCompilation** no

**RoxygenNote** 7.0.2

## R topics documented:

---

`cluster.doc` *Clonal deconvolution*

---

## Description

Clone / Sub-clone decomposition of DNA sequencing data. This is recommended to be used for more than one sample preferably collected from the same individual at different times. If the sample qualities vary, it is recommended to perform scaling first with `seqn.scale`.

## Usage

```
cluster.doc(data, sample, vaf, optimization.method, clustering.method)
```

## Arguments

| | |
|---|---|
| `data` | A `dataframe` containing summary from DNA sequencing. It must include a column of sample IDs and a corresponding column with the variant allele frequencies. |
| `sample` | `Integer or character` of the collumn name or collumn number of the sample IDs. |
| `vaf` | `Integer or character` of the collumn name or collumn number of the variant allele frequency. |
| `optimization.method` | Method to find optimal number of clusters; *GMM* or *bootstrap*. Default is *GMM*. |
| `clustering.method` | Clustering methods; *hkm*, *bootkm* or *hybrid*. Default is *hkm*. |

## Details

`cluster.doc` is meant to do two things, first determine the optimum number of clusters that *should* be fitted and second, to infer what groups the clusters thus obtained should be assigned to.

The data inputs interactively requested from the user help obtain the following information

`chromosomal segmentation` helps in determining the number of clone/sub-clone cloud to be expected in the data. As variant alleles from different aberrant chromosomes may have similar relative frequencies but discordant clonal interpretation. On the contrary convergent clonal alleles may demonstrate divergent frequencies if arisen from dissimilar aneuploidy.

`clouds` give the program a visual feedback from the user that assume to carry some biological interpretation of the frequency distributions present in the data. This is a subjective estimate that the program later uses for cluster assignment.

Out of the two methods used for cluster optimization, *GMM* stands for *Gaussian Mixed Models* whereas *bootstrap*, as the name suggests perform *bootstrap* resampling of the VAFs in 50 repetitions with 20 runs each to find the most stable parameter for clustering. *GMM*

outputs the optimization curve with `BIC` or *Bayesian Information Criterion* against number of clusters chosen in the `X-axis` where *bootstrap* shows the `Smin` statistics instead in the `Y-axis`. In both cases the statistics are to be interpreted as proxies for the *entropy* of the system. The maximum entropy is likely to indicate the most stable solution.

`clustering.method` gives the user three choices:

`hkm` is *Heierarchical K-means clustering* which uses heierarchical clustering first to determine the cluster centers that are subsequently used as the starting point for the K-means clustering.

`bootkm` performs a *bootstrap* resampling of 20 fitted K-means clusters with 50 resamplings to out put the clusters.

`hybrid` performs *hkm* on the principal component of the data.

## Value

A list of 12 objects is returned that includes all the summary statistics, diagnositics and the predictions as well as the mapping internally used for clonal deconvolution.

`predicted.data` is necessarily an extension to the input `data` with the addition of the predicted clone and sub-clone status of each variant for corresponding samples.

`density.map` is a distance matrix convoluted from cluster distances and desity departures.

`collapse` are clusters that are initially prredicted but later collapsed on each other dues to similarity between them.

`fitted.hkm,fitted bootkm or fitted.hybrid` is a vector of initial cluster assignment by the algorithm chosen. Only one of these will have an output and the rest will show `NA`.

`Number of unscaled clusters` gives umber of predicted clusters before collapsing with density estimates.

`Number of scaled clusters` gives number of predicted clusters after collapsing (if any).

`cluster.diagnostics` if the optimization method was chosen to be *GMM*, this is an object of `S3` class that includes clustering diagnostics from the model-based clustering. If the chosen method was *bootstrap* then this is a list.

`cluster centers` are the centroids of the predicted scaled clusters.

`cluster mapping` provides the map between scaled clusters and the clonal deconvolution assignments

`Dunn index` is the Dunn index for the fitted cluster.

## See Also

`seqn.scale` `cluster.doubt`

## Examples

```
cluster.doc(test.dat, 1, 2, optimization.method = 'GMM', clustering.method = 'hkm')
```

`cluster.doubt` *User overriden clonal deconvolution*

## Description

Sample specific user curated Clone / Sub-clone decomposition of DNA sequencing data

## Usage

```
cluster.doubt(CD.obj, sample, vaf, sample.name, cluster.num)
```

## Arguments

| | |
|---|---|
| `CD.obj` | A `cluster.doc` object |
| `sample` | Collumn number of the *predicted.data* from the `cluster.doc` output that contain sample IDs |
| `vaf` | Collumn number of the *predicted.data* from the `cluster.doc` output that contain variant allele frequencies used for the analysis. |
| `sample.name` | a vector of sample IDs |
| `cluster.num` | a numeric vetor of clone/sub-clonal split of respective sample |

## Value

A list of 3 objects

`fitted.cluster` includes the clustering results from the final fit with user input

`predicted.data` is the original fit rendered from `cluster.doc`

`userfed.data` shows the changed clustering results due to the user defined clone / sub-clone smear for the selected samples

## See Also

`cluster.doc`

## Examples

```
cd.res<-cluster.doc(test.dat)
cd.new<-cluster.doubt(cd.res,sample,vaf,c("Sample_1","Sample_3"),c(2,2,3,2))
```

CopySeg *Copynumber estimation*

## Description

Allelic segmentations are estimated for one sample at a time with unfiltered sequencing calls using the package `sequenza`.

## Usage

```
CopySeg(x, tumor.sample, normal.sample, DP, AD, AF, file.name)
```

## Arguments

| | |
|---|---|
| `x` | A `vcfR` object of the WES calls. |
| `tumor.sample` | a `character` denoting the *name* of the sample (only one sample). The sample names can be queried from `x`. |
| `normal.sample` | a `character` denoting the *name* of the normal tissue sample against which the tumor sample is to be analyzed. |
| `DP` | a `character` denoting *ID* for total read depth (empirical or approx) to be extracted from the *FORMAT* field of `x` |
| `AD` | a `character` deoning *ID* for depth of the reference allele. |
| `AF` | an optional `character` deoning *ID* for allele fraction of the reference allele. This is often separately present in the VCF file. Default is `NULL`. |
| `file.name` | an optional `character` to define output file name. Default is *tumor.sample*. |

## Details

The function writes a *.txt* data in working directory with the name defined in `file.name` used by *sequenza*. The output file written can be used in conjunction with post variants call sequence file. These can be merged and used for surther analysis with `cluster.doc` or `seqn.scale`

## Value

A transformed `dataframe` usable in *CloneStrat* that represents data on all variants in the .vcf file. It returns summaries on the variants with the collumn *CN.profile* depicting the estimated allelic segmentations.

## Examples

```
CopySeg(x = sample.vcf, tumor.sample = c("tumor"),
normal.sample = "normal", DP = 'DP', AD = 'AD', AF = 'AF', file.name = 'temp')
```

## mutect2.qc *Quality Control on Mutect2 output*

### Description

A quality control (QC) and transformation on the WES output from the Mutect2 variant caller. This re-organizes the data in a way that is friendlier for using in *CloneStrat*

### Usage

```
mutect2.qc(WES, sample.name)
```

### Arguments

| | |
|---|---|
| WES | A dataframe of the Mutect2 output |
| sample.name | a vector of sample names or IDs |

### Value

A transformed dataframe usable in *CloneStrat* that represents data on each variant of each sample in rows

### Examples

```
res<-mutect2.qc(WES,sample.name)
```

## seqn.scale *Probabilistic quotient normalization of DNA sequencing data*

### Description

A normalization technique based on cancer / tumor cell fractions of the samples sequenced to infer homogeneity

### Usage

```
seqn.scale(x, vaf, CCF)
```

### Arguments

| | |
|---|---|
| x | A dataframe containing summary from DNA sequencing with first column as sample IDs of corresponding variants. |
| vaf | The column number of x that includes VAFs. |
| CCF | The column number of x that includes CCFs. |

### Details

Probabilistic quotient normalization normalization technique described in *Dieterle, et al. (2006)* applied on the cancer cell fraction (CCF) of respective samples to rescale variant allele frequencies (VAF) accordingly. The general idea is to put most confidence in the sample with highest CCF and adjust the VAFs of other samples based on the departure in CCF of the other samples from that with the highest.

This method is particularly suggested if the CCFs accross samples vary more than 10

### Value

A `dataframe` with all the elements of `x` with the new estimated VAFs in the column *scaled.vaf* and an additional column *unscaled.vaf* that includes the original VAFs.

### See Also

`cluster.doc`

### Examples

```
pqn.dat<-seqn.scale(test.dat,vaf=2,CCF=3)
hist(pqn.dat$scaled.vaf)
```

---

## `T.goodness.test` *Test of fit of clonal deconvolution*

---

### Description

A chi square test to assess the *goodness of fit* of the clonal : sub-clonal clouds. This test can be used to obtain outliers that do not fit into the proposed clonal deconvolution space.

### Usage

```
T.goodness.test(x)
```

### Arguments

| | |
|---|---|
| `x` | A `dataframe` with the first three columns in the specific order: sample name or ID of a variant, variant allele frquencies (VAF) and cancer cell fraction (CCF) |

### Value

A list of two objects. *x* is same as the input `dataframe` with addede columns named *expected VAF_*, *chi_sq_* and *P value_* corresponding to each cloud of clone : Sub-clone combination. *rej* is a subset of `x` containing variants that fail the test for at least one cloud.

*expected VAF_* represents estimated variant allele frequencies for a given cloud.

*chi_sq_* is the Chi square test statistic for the cloud.

*P value_* is the P value corresponding to the *chi_sq_* statistic.

### Examples

```
T.goodness.test(test.dat)
```

## test.dat — *Random number generated WES data for eight hypothetical samples*

### Description

Data generated with varying random normal probabilities. ideal chromosomal segmentation profile is assumed resulting in three separate distinct clouds of clones and sub-clones.

### Usage

```
data(test.dat)
```

### Format

An object of class `"dataframe"`

### Value

`sample` is column of IDs corresponding to 8 distinct samples.

`vaf` denotes the variant allele frequencies of each variant (see `annotation`).

`CCF` are the cancer cell fractions of each sample.

`annotation` indicates corresponding variants for which observations are notes in each row. Variants can be shared among several samples as well as be private mutation.

### Examples

```
data(test.dat)
table(test.dat$CCF)
table(test.dat$annotation)
hist(test.dat$vaf)
```

## variant.auto.plot — *Automated Multi-sample plot*

### Description

Automated plotting of all variants present in the WES data

### Usage

```
variant.auto.plot(CD.obj, annotation.col)
```

### Arguments

| | |
|---|---|
| `CD.obj` | A `cluster.doc object` |
| `annotation.col` | name of the column containing annotations of the variants in original WES `dataframe` used in the clonal deconvolution using `cluster.doc` |

### Value

Plot objects with the relevant annotation highlighted.

This function plots all variants present in the sample. Depending on the number of variants this can generate a *lot* of plots. All of these plots will be saved under a new directory named `img` inside the working directory. Hence, it is important to check that there are no directory named `img` inside the working directory

### Examples

```
cd.res<-cluster.doc(test.dat,1,2)
variant.auto.plot(cd.res,'annotation')
```

---

## variant.plot *Multi-sample variant plot*

---

### Description

Plotting a specific variant present in more than one WES sample

### Usage

```
variant.plot(CD.obj, annotation.col, variant)
```

### Arguments

| | |
|---|---|
| `CD.obj` | A `cluster.doc object` |
| `annotation.col` | name of the column containing annotations of the variants in original WES `dataframe` used in the clonal deconvolution using `cluster.doc` |
| `variant` | a `character` string specifying *only one* annotation which is to be displayed |

### Value

A plot object with the relevant annotation highlighted

### Examples

```
cd.res<-cluster.doc(test.dat,1,2)
variant.plot(cd.res,'annotation','variant_74')
```

# Index