

# Package ‘poolfstat’

May 2, 2022

**Maintainer** Mathieu Gautier <mathieu.gautier@inrae.fr>

**Author** Mathieu Gautier

**Version** 2.1.1

**License** GPL (>= 2)

**Title** Computing f-Statistics and Building Admixture Graphs Based on Allele Count or Pool-Seq Read Count Data

**Description** Functions for the computation of f- and D-statistics (estimation of 'Fst', Patterson's 'F2', 'F3', 'F3\*', 'F4' and D parameters) in population genomics studies from allele count or Pool-Seq read count data and for the fitting, building and visualization of admixture graphs. The package also includes several utilities to manipulate Pool-Seq data stored in standard format (e.g., such as 'vcf' files or 'rsync' files generated by the 'PoPoolation' software) and perform conversion to alternative format (as used in the 'BayPass' and 'SelEstim' software). As of version 2.0, the package also includes utilities to manipulate standard allele count data (e.g., stored in 'TreeMix', 'BayPass' or 'SelEstim' format).

**LinkingTo** Rcpp, RcppProgress

**Imports** Rcpp (>= 1.0.5), methods, utils, foreach, doParallel, parallel, DiagrammeR, ape, stats, zoo, Ryacas, Matrix, RcppProgress, progress, nnls

**Depends** R (>= 3.0)

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2022-05-02 16:10:02 UTC

## R topics documented:

add.leaf . . . . .	3
compare.fitted.fstats . . . . .	4
compute.f4ratio . . . . .	5
compute.fstats . . . . .	6

compute.pairwiseFST . . . . .	7
computeFST . . . . .	9
countdata-class . . . . .	10
countdata.subset . . . . .	11
find.tree.popset . . . . .	12
fit.graph . . . . .	14
fitted.graph-class . . . . .	16
fstats-class . . . . .	17
generate.graph.params . . . . .	18
generate.jackknife.blocks . . . . .	20
genobypass2countdata . . . . .	20
genobypass2pooldata . . . . .	22
genoseestim2pooldata . . . . .	23
genotreemix2countdata . . . . .	25
graph.builder . . . . .	26
graph.params-class . . . . .	28
graph.params2qpGraphFiles . . . . .	29
graph.params2symbolic.fstats . . . . .	30
heatmap,pairwisefst-method . . . . .	31
is.countdata . . . . .	33
is.fitted.graph . . . . .	33
is.fstats . . . . .	34
is.graph.params . . . . .	34
is.pairwisefst . . . . .	34
is.pooldata . . . . .	35
make.example.files . . . . .	35
pairwisefst-class . . . . .	36
plot,fitted.graph-method . . . . .	36
plot,fstats-method . . . . .	37
plot,graph.params-method . . . . .	37
plot,pairwisefst-method . . . . .	38
plot_fstats . . . . .	38
pooldata-class . . . . .	40
pooldata.subset . . . . .	40
pooldata2genobypass . . . . .	42
pooldata2genoseestim . . . . .	43
poolfstat . . . . .	44
popsync2pooldata . . . . .	45
rooted.njtree.builder . . . . .	46
show,countdata-method . . . . .	48
show,fitted.graph-method . . . . .	48
show,fstats-method . . . . .	49
show,graph.params-method . . . . .	49
show,pairwisefst-method . . . . .	50
show,pooldata-method . . . . .	50
vcf2pooldata . . . . .	51

---

add.leaf	<i>Test all possible connection of a leaf to a graph with non-admixed and or admixed edges</i>
----------	--

---

## Description

Test all possible connection of a leaf to a graph with non-admixed and or admixed edges

## Usage

```
add.leaf(
  x,
  leaf.to.add,
  fstats,
  only.test.non.admixed.edges = FALSE,
  only.test.admixed.edges = FALSE,
  verbose = TRUE,
  ...
)
```

## Arguments

x	An object of class graph.params or fitted.graph (see details)
leaf.to.add	Name of the leaf to add
fstats	Object of class fstats that contains estimates of the fstats (see compute.fstats)
only.test.non.admixed.edges	If TRUE the function only test non.admixed edges (may be far faster)
only.test.admixed.edges	If TRUE the function only test admixed edges
verbose	If TRUE extra information is printed on the terminal
...	Some parameters to be passed the function fit.graph called internally

## Details

The input object x needs to be of class graph.params (as generated by the function generate.graph.params) or fitted.graph (as generated by the function fit.graph or by the function add.leaf itself in the graphs.fit.res elements of the output list). This is to ensure that the matrix describing the structure of the graph (graph slot of these objects) is valid (note that it can be plotted for checks). Hence graph.params objects may have been generated without fstats information (that should be supplied independently to the add.leaf function to obtain information on the fstats involving the candidate leaf defined with the leaf.to.add argument). By default the function tests all the possible positions of a newly added edge connecting the candidate leaf to the graph with both non-admixed (including a new rooting with the candidate leaf as an outgroup) and admixed edges. If n\_e is the the number of non-admixed edges of the original graph, the number of tested graphs for non-admixed edges equals n\_e+1. The newly added node is named "N-"name of the leaf to add (or with more N if the

name already exists). For admixed edges, the number of tested graphs equals  $n_e \cdot (n_e - 1) / 2$  and for a given tested graph, three nodes named "S-"name of the leaf to add, "S1-"name of the leaf to add and "S2-"name of the leaf to add (or with more S if the name already exists) are added and the admixture proportions are named with a letter (A to Z depending on the number of admixed nodes already present in the graph).

### Value

A list with the following elements:

1. "n.graphs": The number of tested graphs
2. "fitted.graphs.list": a list of fitted.graph objects (indexed from 1 to n.graphs and in the same order as the list "graphs") containing the results of fitting of each graph.
3. "best.fitted.graph": The graph (object of class fitted.graph) with the minimal BIC (see function fit.graph) among all the graphs within fitted.graphs.list
4. "bic": a vector of the n.graphs BIC (indexed from 1 to n.graphs and in the same order as the "fitted.graphs.list" list) (see fit.graph details for the computation of the scores).

### See Also

see [fit.graph](#) and [generate.graph.params](#).

---

compare.fitted.fstats *Compare fitted f2, f3 and f4 f-statistics of an admixture graph with estimated ones*

---

### Description

Compare fitted f2, f3 and f4 f-statistics of an admixture graph with estimated ones

### Usage

```
compare.fitted.fstats(fstats, fitted.graph, n.worst.stats = 5)
```

### Arguments

fstats	Object of class fstats containing estimates of fstats (as obtained with compute.fstats)
fitted.graph	Object of class fitted graph (as obtained with fit.graph function).
n.worst.stats	The number of worst statistics to be displayed in the terminal

### Details

Compare fitted and estimated f-statistics may allow identifying problematic edges on the graph.

**Value**

A matrix with 3 columns for each test (row names of the matrix corresponding to the test):

1. The estimated f-statistics (mean across block-Jackknife samples)
2. The fitted f-statistics (obtained from the fitted graph parameters)
3. A Z-score measuring the deviation of the fitted values from the estimated values in units of standard errors (i.e.,  $Z = (\text{fitted.value} - \text{target.value}) / \text{se}(\text{target.value})$ )

**See Also**

See [compute.fstats](#) and [fit.graph](#)

---

<code>compute.f4ratio</code>	<i>Compute F4ratio (estimation of admixture rate) from an fstats object</i>
------------------------------	---

---

**Description**

Compute F4ratio (estimation of admixture rate) from an fstats object

**Usage**

```
compute.f4ratio(x, num.quadruplet, den.quadruplet)
```

**Arguments**

<code>x</code>	A fstats object containing estimates of fstats
<code>num.quadruplet</code>	A character vector for the F4 quadruplet used in the F4ratio numerator (should be of the form "A,O;C,X" where A, O, C and X are the names of the population as defined in the countdata or pooldata object used to obtain fstats, see details)
<code>den.quadruplet</code>	A character vector for the F4 quadruplet used in the F4ratio denominator (should be of the form "A,O;C,B" where A, O, C and B are the names of the populations as defined in the countdata or pooldata object used to obtain fstats, see details)

**Details**

Assuming a 4 population phylogeny rooted with an outgroup O of the form (((A,B);C);O) and an admixed population X with two source populations related to B and C, the admixture rate alpha of the B-related ancestry is obtained using the ratio  $F4(A,O;C,X)/F4(A,O;C,B)$  (see Patterson et al., 2012 for more details).

**Value**

Either a scalar corresponding to the estimated admixture rate or, if F4 block-jackknife samples are available in the input fstats object (i.e., `compute.fstats` was run with `return.F4.blockjackknife.samples = TRUE`) a vector with three elements corresponding to the estimate of the admixture rate, the block-jackknife mean (may be slightly different than the previous since not exactly the same set of markers are used) and the standard error of the estimates.

**See Also**

To generate pooldata object, see [vcf2pooldata](#), [popsync2pooldata](#), [genobypass2pooldata](#) or [genoselestim2pooldata](#). To generate countdata object, see [genobypass2countdata](#) or [genotreemix2countdata](#).

**Examples**

```
make.example.files(writing.dir=tempdir())
pooldata=popsync2pooldata(sync.file=paste0(tempdir(),"/ex.sync.gz"),poolsizes=rep(50,15))
res.fstats=compute.fstats(pooldata)
```

---

compute.fstats	<i>Estimate the F-statistics (F2, F3, F3star, F4, Dstat)</i>
----------------	--

---

**Description**

Estimate the F-statistics (F2, F3, F3star, F4, Dstat)

**Usage**

```
compute.fstats(
  x,
  nsnp.per.bjack.block = 0,
  computeDstat = FALSE,
  return.F4.blockjackknife.samples = FALSE,
  verbose = TRUE
)
```

**Arguments**

x	A pooldata object containing Pool-Seq information or a countdata object containing allele count information
nsnp.per.bjack.block	Number of consecutive SNPs within a block for block-jackknife (default=0, i.e., no block-jackknife sampling)
computeDstat	If TRUE compute Dstatistics (i.e. scaled F4). This may add some non negligible computation time if the number of population is large (n>15)
return.F4.blockjackknife.samples	If TRUE (and nsnp.per.bjack.block>0) return F4 estimates for each block-jackknife sample (useful to compute F4 ratios standard errors)
verbose	If TRUE extra information is printed on the terminal

**Details**

The function estimates for the  $n$  populations (or pools) represented in the input object  $x$ :

1. The F2 statistics for all the  $n(n-1)/2$  pairs of populations (or pools) and their scaled version (equivalent to Fst as compute with `compute.pairwiseFST` with `method="Identity"`)
2. If  $n>2$ , The F3 statistics for all the  $n \text{pools}(n \text{pools} - 1)(n \text{pools} - 2)/2$  possible triplets of populations (or pools) and their scaled version (named F3star after Patterson et al., 2012)
3. If  $n>3$ , The F4 statistics and the D-statistics (a scaled version of the F4) for all the  $n \text{pools}(n \text{pools} - 1)(n \text{pools} - 2) * (n \text{pools} - 3)/8$  possible quadruplets of populations
4. The estimated within population heterozygosities (=1-Q1)

**Value**

An object of class `fstats` (see `help(fstats)` for details)

**See Also**

To generate `pooldata` object, see [vcf2pooldata](#), [popsync2pooldata](#), [genobypass2pooldata](#) or [genoselestim2pooldata](#). To generate `countdata` object, see [genobypass2countdata](#) or [genotremix2countdata](#).

**Examples**

```
make.example.files(writing.dir=tempdir())
pooldata=popsync2pooldata(sync.file=paste0(tempdir(),"/ex.sync.gz"),poolsizes=rep(50,15))
res.fstats=compute.fstats(pooldata)
```

---

`compute.pairwiseFST`     *Compute pairwise population population FST matrix (and possibly all pairwise SNP-specific FST)*

---

**Description**

Compute pairwise population population FST matrix (and possibly all pairwise SNP-specific FST)

**Usage**

```
compute.pairwiseFST(
  x,
  method = "Anova",
  min.cov.per.pool = -1,
  max.cov.per.pool = 1e+06,
  min.indgeno.per.pop = -1,
  min.maf = -1,
  output.snp.values = FALSE,
  nsnp.per.bjack.block = 0,
  verbose = TRUE
)
```

**Arguments**

x	A pooldata object containing Pool-Seq information or a countdata object containing allele count information
method	Either "Anova" (default method as described in the manuscript) or "Identity" (relies on an alternative modeling consisting in estimating unbiased Probability of Identity within and across pairs of pools)
min.cov.per.pool	For Pool-Seq data (i.e., pooldata objects) only: minimal allowed read count (per pool). If at least one pool is not covered by at least min.cov.perpool reads, the position is discarded in the corresponding pairwise comparisons
max.cov.per.pool	For Pool-Seq data (i.e., pooldata objects) only: maximal allowed read count (per pool). If at least one pool is covered by more than min.cov.perpool reads, the position is discarded in the corresponding pairwise comparisons.
min.indgeno.per.pop	For allele count data (i.e., countdata objects) only: minimal number of overall counts required in each population. If at least one pop is not genotyped for at least min.indgeno.per.pop (haploid) individual, the position is discarded
min.maf	Minimal allowed Minor Allele Frequency (computed from the ratio overall read counts for the reference allele over the read coverage) in the pairwise comparisons.
output.snp.values	If TRUE, provide SNP-specific pairwise FST for each comparisons (may lead to a huge result object if the number of pools and/or SNPs is large)
nsnp.per.bjack.block	Number of consecutive SNPs within a block for block-jackknife (default=0, i.e., no block-jackknife sampling)
verbose	If TRUE extra information is printed on the terminal

**Value**

An object of class pairwisefst (see help(pairwisefst) for details)

**See Also**

To generate pooldata object, see [vcf2pooldata](#), [popsync2pooldata](#), [genobypass2pooldata](#) or [genoselestim2pooldata](#). To generate countdata object, see [genobypass2countdata](#) or [genotreemix2countdata](#).

**Examples**

```
make.example.files(writing.dir=tempdir())
pooldata=popsync2pooldata(sync.file=paste0(tempdir(),"/ex.sync.gz"),poolsizes=rep(50,15))
PairwiseFST=compute.pairwiseFST(pooldata)
```



---

computeFST

*Compute FST from Pool-Seq data or Count data*


---

**Description**

Compute FST from Pool-Seq data or Count data

**Usage**

```
computeFST(
  x,
  method = "Anova",
  nsnp.per.bjack.block = 0,
  sliding.window.size = 0,
  verbose = TRUE
)
```

**Arguments**

x	A pooldata object containing Pool-Seq information or countdata object containing allele counts information
method	Either "Anova" (default method as described in Hivert et al (2018, eq. 9) for pool-seq data and Weir (1996, eq. 5.2) for count data) or "Identity" (relying on unbiased estimators of Probability of Identity within and across pairs of pools/populations)
nsnp.per.bjack.block	Number of consecutive SNPs within a block for block-jackknife (default=0, i.e., no block-jackknife sampling)
sliding.window.size	Number of consecutive SNPs within a window for multi-locus computation of Fst over sliding window with half-window size step (default=0, i.e., no sliding-window scan)
verbose	If TRUE extra information is printed on the terminal

**Value**

A list with the four following elements:

1. "FST": a scalar corresponding to the estimate of the genome-wide FST over all the populations
2. "snp.FST": a vector containing estimates of SNP-specific FST
3. "snp.Q1": a vector containing estimates of the overall within pop. SNP-specific probability of identity
4. "snp.Q2": a vector containing estimates of the overall between pop. SNP-specific probability of identity

5. "mean.fst" (if nsnp.per.bjack.block>0): genome-wide Fst estimate as the mean over block-jackknife samples (may slight differ from "FST" estimate since it is only computed on SNPs eligible for Block-Jackknife)
6. "se.fst" (if nsnp.per.bjack.block>0): standard-error of the genome-wide Fst estimate computed block-jackknife samples
7. "fst.bjack.samples" (if nsnp.per.bjack.block>0): a vector containing estimates of the overall between pop. SNP-specific probability of identity
8. "sliding.windows.fst" (if sliding.window.size>0): a 4-columns data frame containing information on multi-locus Fst computed for sliding windows of SNPs over the whole genome with i) column with the chromosome/contig of origin of each window; ii) the mid-position of each window; iii) the cumulated mid-position of each window (to facilitate further plotting); and iv) the estimated multi-locus Fst

### See Also

To generate pooldata object, see [vcf2pooldata](#), [popsync2pooldata](#), [genobypass2pooldata](#) or [genoselestim2pooldata](#). To generate countdata object, see [genobypass2countdata](#) or [genotreemix2countdata](#).

### Examples

```
make.example.files(writing.dir=tempdir())
pooldata=popsync2pooldata(sync.file=paste0(tempdir(),"/ex.sync.gz"),poolsizes=rep(50,15))
res.fst=computeFST(pooldata)
```

---

countdata-class

*S4 class to represent a Count data set.*

---

### Description

S4 class to represent a Count data set.

### Slots

npops The number of populations

nsnp The number of SNPs

refallele.count A matrix (nsnp rows and npops columns) with the allele counts for the reference allele

total.count A matrix (nsnp rows and npops columns) with the total number of counts (i.e., twice the number of genotyped individual for diploid species and autosomal markers)

snp.info A data frame (nsnp rows and 4 columns) detailing for each SNP, the chromosome (or scaffold), the position, Reference allele name and Alternate allele name (if available)

popnames A vector of length npops with the corresponding population names

### See Also

To generate countdata object, see [genobypass2countdata](#) and [genotreemix2countdata](#)

---

countdata.subset	<i>Create a subset of a countdata object that contains count data as a function of pop or SNP indexes</i>
------------------	---

---

### Description

Create a subset of a countdata object that contains count data as a function of pop or SNP indexes

### Usage

```
countdata.subset(
  countdata,
  pop.index = 1:countdata@pops,
  snp.index = 1:countdata@snp,
  min.indgeno.per.pop = -1,
  min.maf = -1,
  return.snp.idx = FALSE,
  verbose = TRUE
)
```

### Arguments

countdata	A countdata object containing Allele count information
pop.index	Indexes of the pools (at least two), that should be selected to create the new pooldata object (default=all the pools)
snp.index	Indexes of the SNPs (at least two), that should be selected to create the new pooldata object (default=all the SNPs)
min.indgeno.per.pop	Minimal number of overall counts required in each population. If at least one pop is not genotyped for at least min.indgeno.per.pop (haploid) individual, the position is discarded
min.maf	Minimal allowed Minor Allele Frequency (computed from the ratio overall counts for the reference allele over the overall number of (haploid) individual genotyped)
return.snp.idx	If TRUE, the row.names of the snp.info slot of the returned pooldata object are named as "rsx" where x is the index of SNP in the initial pooldata object (default=FALSE)
verbose	If TRUE return some information

### Details

This function allows subsetting a pooldata object by selecting only some pools and/or some SNPs (e.g., based on their position on the genome). Additional filtering steps on SNPs can be carried out on the resulting subset to discard SNP with low polymorphism or poorly or too highly covered. In addition, coverage criteria can be applied on a per-pool basis with the cov.qthres.per.pool argument.

'more specific SNP selection based on their positions on the genome or their characteristics. For instance if  $q_{max}=0.95$ , a position is discarded if in a given pool it has a number of reads higher than the 95-th percentile of the empirical coverage distribution in this same pool (defined over the SNPs selected by `snp.index`). Similarly, if  $q_{max}=0.05$ , a position is discarded if in a given pool it has a number of reads lower than the 5-th percentile of the empirical coverage distribution in this same pool. This mode of selection may be more relevant when considering pools with heterogeneous read coverages.

### Value

A `countdata` object with 6 elements:

1. "refallele.count": a matrix (`nsnp` rows and `npops` columns) with the allele counts for the reference allele
2. "total.count": a matrix (`nsnp` rows and `npops` columns) with the total number of counts (i.e., twice the number of genotyped individual for diploid species and autosomal markers)
3. "snp.info": a matrix with `nsnp` rows and four columns containing respectively the contig (or chromosome) name (1st column) and position (2nd column) of the SNP; the allele taken as reference in the `refallele.count` matrix (3rd column); and the alternative allele (4th column)
4. "popnames": a vector of length `npops` containing the names of the pops
5. "nsnp": a scalar corresponding to the number of SNPs
6. "npops": a scalar corresponding to the number of populations

### See Also

To generate `countdata` object, see [genobypass2countdata](#), [genotremix2countdata](#)

### Examples

```
make.example.files(writing.dir=tempdir())
pooldata=popsync2pooldata(sync.file=paste0(tempdir(),"/ex.sync.gz"),poolsizes=rep(50,15))
pooldata2genobypass(pooldata=pooldata, writing.dir=tempdir())
##NOTE: This example is just for the sake of illustration as it amounts to
##interpret read count as allele count which must not be done in practice!
countdata=genobypass2countdata(genobypass.file=paste0(tempdir(),"/genobypass"))
subset.by.snps=countdata.subset(countdata,snp.index=10:100)
subset.by.pops.and.snps=countdata.subset(countdata,pop.index=c(1,2),snp.index=10:100)
```

---

find.tree.popset

*Find sets of populations that may used as scaffold tree*

---

### Description

Find sets of populations that may used as scaffold tree

**Usage**

```
find.tree.popset(
  fstats,
  f3.zcore.threshold = -1.65,
  f4.zscore.absolute.threshold = 1.96,
  excluded.pops = NULL,
  nthreads = 1,
  verbose = TRUE
)
```

**Arguments**

<code>fstats</code>	Object of class <code>fstats</code> containing estimates of <code>fstats</code> (see the function <code>compute.fstats</code> )
<code>f3.zcore.threshold</code>	The significance threshold for Z-score of formal test of admixture based on the F3-statistics (default=-2)
<code>f4.zscore.absolute.threshold</code>	The significance threshold for  Z-score  of formal test of treeness based on the F4-statistics (default=2)
<code>excluded.pops</code>	Vector of pop names to be exclude from the exploration
<code>nthreads</code>	Number of available threads for parallelization of some part of the parsing (default=1, i.e., no parallelization)
<code>verbose</code>	If TRUE extra information is printed on the terminal

**Details**

The procedure first discards all the populations P that shows a significant signal of admixture with a Z-score for F3 statistics of the form  $F3(P;Q,R) < f3.zscore.thresholds$ . It then identifies all the sets of populations that pass the F4-based treeness with themselves. More precisely, for a given set E containing n populations, the procedure ensure that all the  $n(n-1)(n-2)(n-3)/8$  possible F4 quadruplets have a  $|Z\text{-score}| < f4.zscore.absolute.threshold$ . The function aims at maximizing the size of the sets.

**Value**

A list with the following elements:

1. "n.sets": The number of sets of (scaffold) unadmixed populations identified
2. "set.size": The number of populations included in each set
3. "pop.sets": A character matrix of n.sets rows and set.size columns giving for each set identified the names of the included populations.
4. "Z\_f4.range": A matrix of n.sets rows and 2 columns reported for each set the range of variation (min and max value) of the absolute F4 Z-scores for the quadruplets passing the treeness test. More precisely, for a given set consisting of n=set.size populations, a total of  $n(n-1)(n-2)(n-3)/8$  quadruplets can be formed. Yet, any set of four populations A, B, C and D is represented by three quadruplets A,B;C,D (or one of its seven other equivalent combinations formed by permuting each pairs); A,C;B,D (or one of its seven other equivalent

combinations) and A,D;B,C (or one of its seven other combinations). Among these three, only a single quadruplet is expected to pass the treeness test (i.e., if the correct unrooted tree topology is (A,C;B,D), then the absolute value of the Z-scores associated to F4(A,B;C,D) and F4(A,D;B,C) or their equivalent will be high.

5. "passing.quadruplets": A matrix of n.sets rows and set.size columns reporting for each sets the  $n(n-1)(n-2)(n-3)/24$  quadruplets that pass the treeness test (see Z\_f4.range detail).

### See Also

see [compute.fstats](#).

### Examples

```
make.example.files(writing.dir=tempdir())
pooldata=popsync2pooldata(sync.file=paste0(tempdir(),"/ex.sync.gz"),poolsizes=rep(50,15))
res.fstats=compute.fstats(pooldata,nsnp.per.bjack.block = 50)
#NOTE: toy example (in practice nsnp.per.bjack.block should be higher)
popsets=find.tree.popset(res.fstats,f3.zcore.threshold=-3)
```

---

fit.graph

*Estimate parameters of an admixture graph*

---

### Description

Estimate parameters of an admixture graph

### Usage

```
fit.graph(
  graph.params,
  Q.lambda = 0,
  eps.admix.prop = 1e-06,
  edge.fact = 1000,
  admix.fact = 100,
  compute.ci = F,
  drift.scaling = F,
  outfileprefix = NULL,
  verbose = TRUE
)
```

### Arguments

graph.params	An object of class graph.params containing graph information and relevant Fstats estimates (see the function generate.graph.params)
Q.lambda	A scalar (usually small) to add to the diagonal elements of the error covariance matrix of fstats estimates (may improve numerical stability of its decomposition for large number of populations)

eps.admix.prop	A scalar defining admixture proportion domain (eps.admix.prop vary between eps.admix.prop and 1-eps.admix.prop)
edge.fact	The multiplying factor of edges length in graph representation
admix.fact	The multiplying factor of admixture proportion in graph representation
compute.ci	Derive 95% Confidence Intervals for the parameters of the admixture graph (edge lengths and admixture rates)
drift.scaling	If TRUE scale edge lengths in drift units (require estimates of leave heterozygosities)
outfileprefix	The prefix of the dot file that will represent the graph (with extension ".dot"). If NULL, no graph file generated
verbose	If TRUE extra information is printed on the terminal

## Details

Let  $f$  represent the  $n$ -length vector of basis target (i.e., observed) F2 and F3 statistics and  $g(e; a) = X(a) * e$  the vector of their expected values given the vector of graph edges lengths  $e$  and the incidence matrix  $X(a)$  that depends on the structure of the graph and the admixture rates  $a$  (if there is no admixture in the graph,  $X(a)$  only contains 0 or 1). The function attempts to find the  $e$  and  $a$  graph parameter values that minimize a cost (score of the model) defined as  $S(e; a) = (f - g(e; a))'Q^{-1}(f - g(e; a))$ . Assuming  $f \sim N(g(e; a), Q)$  (i.e., the observed f-statistics vector is multivariate normal distributed around an expected  $g$  vector specified by the admixture graph and a covariance structure empirically estimated),  $S = -2\log(L) - K$  where  $L$  is the likelihood of the fitted graph and  $K = n * \log(2 * \pi) + \log(|Q|)$ . Also, for model comparison purpose, a standard  $BIC$  is then derived from  $S$  as  $BIC = S + p * \log(n) - K$  ( $p$  being the number of graph parameters, i.e., edge lengths and admixture rates). As mentioned by Patterson et al. (2012), the score  $S(e; a)$  is quadratic in edge lengths  $e$  given  $a$ . The function uses the Lawson-Hanson non-negative linear least squares algorithm implemented in the `nls` function (package `nls`) to estimate  $e$  (subject to the constraint of positive edge lengths) by finding the vector  $e$  that minimize  $S(e; a) = (f - X(a) * e)'Q^{-1}(f - X(a) * e) = \|G * f - G * X(a) * e\|^2$  (where  $G$  results from the Cholesky decomposition of  $Q^{-1}$ , i.e.,  $Q^{-1} = G'G$ ). Note that the `*Q.lambda*` argument may be used to add a small constant (e.g.,  $1e - 4$ ) to the diagonal elements of  $Q$  to avoid numerical problems (see Patterson et al., 2012). Yet `*Q.lambda*` is always disregarded when computing the final score  $S$  and  $BIC$ . Minimization of  $S(e; a)$  is thus reduced to the identification of the admixture rates ( $a$  vector) which is performed using the L-BFGS-B method (i.e., Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm with box constraints) implemented in the `optim` function (stats package). The `*eps.admix.prop*` argument allows specifying the lower and upper bound of the admixture rates to `*eps.admix.prop*` and `*1-eps.admix.prop*` respectively. Scaling of the edges lengths in drift units (i.e., in units of  $t/2N$  where  $t$  is time in generations and  $N$  is the effective population size) is performed as described in Lipson et al. (MBE, 2013) by dividing the estimated edges lengths by half the estimated heterozygosity of their parental nodes (using the property  $h_p = h_c + 2e(C, P)$  where  $h_p$  and  $h_c$  are the heterozygosities of a child  $C$  and its parent  $P$  node and  $e(C, P)$  is the estimated length of the branch relating  $C$  and  $P$ ). Finally, if `compute.ci=TRUE`, a (rough) 95% confidence intervals is computed using a bisection method (with a  $1e - 4$  precision) for each parameters in turn (all others being set to their estimated value). Note that 95% CI are here defined as the set of values associated to a score  $S$  such that  $S_{opt} < S < S_{opt} + 3.84$  (where  $S_{opt}$  is the optimized score), i.e., with a likelihood-ratio test statistic with respect to the fitted values  $< 3.84$  (the 95% threshold of a one ddl Chi-square distribution).

**Value**

An object of class fitted.graph (see help(fitted.graph) for details)

**See Also**

To generate a graph.params object, see [generate.graph.params](#). The fitted graph may be plotted directly using plot that calls grViz() function and the resulting fitted fstats may be compared to the estimated ones with [compare.fitted.fstats](#).

---

fitted.graph-class	<i>S4 class to represent a population tree or admixture graph and its underlying fitted parameter.</i>
--------------------	--

---

**Description**

S4 class to represent a population tree or admixture graph and its underlying fitted parameter.

**Details**

The dot.graph element allows to plot the graph using grViz() from the DiagrammeR package or with the dot program after writing the files (e.g., dot -Tpng inputgraph.dot in terminal). Note that the dot file may be customized (e.g., to change leave color, parameter names...).

**Slots**

graph The graph in 3 column format originated from the fitted.graph.params object

dot.graph The fitted graph in dot format

score the score of the model (squared Mahalanobis distance between the observed and fitted basis F-statistics vectors)

bic The Bayesian Information Criterion associated to the model

fitted.outstats a matrix containing the target values of the fstats, the fitted values and the Z-score measuring the deviation of the fitted values from the target values in units of standard errors (i.e.,  $Z=(\text{fitted.value}-\text{target.value})/\text{se}(\text{target.value})$ )

edges.length a vector containing the estimated edges.length. Note finally, that the (two) edges coming from the roots are assumed of equal length (i.e., unrooted branch) as these are non-identifiable by the method.

edges.length.scaled If drift.scaling=TRUE, the estimated edges.length in units of  $t/2N$

edges.length.ci A matrix with two columns (or four columns if drift scaled lengths are computed) containing for each edge length (in a row) the 95% CI lower and higher bounds (columns 3 and 4 containing 95% CI lower and higher bounds of drift scaled lengths, if any)

admix.prop a vector containing the estimated admixture proportions (if any)

admix.prop.ci a matrix with two columns containing for each admixture proportion (in a row) the 95% CI lower and higher bounds



- `nodes.het` The estimated heterozygosities for all nodes (if available; see `drift.scaling` argument in `fit.graph`)
- `fitted.f2.mat` the matrix of all the fitted F2 statistics (obtained from fitted admixture graph parameter values) from which all the fitted fstats can be derived.
- `optim.results` list containing results of the optim call

### See Also

To generate `fitted.graph` object, see [fit.graph](#).

---

`fstats-class`                      *S4 class to represent fstats results obtained with computeFstats.*

---

### Description

S4 class to represent fstats results obtained with `computeFstats`.

### Slots

- `f2.values` A data frame with  $npop(npop-1)/2$  rows and 1 (or 3 if `blockjackknife` is TRUE) columns containing estimates of the f2-statistics over all the SNPs and if `blockjackknife=TRUE`, the estimated block-jackknife and standard error (s.e.)
- `fst.values` A data frame with  $npop(npop-1)/2$  rows and 1 (or 3 if `blockjackknife` is TRUE) columns containing estimates of the scaled f2.values (same as obtained with `compute.pairwiseFST` with `method="Identity"`) over all the SNPs and if `blockjackknife=TRUE`, the estimated block-jackknife and standard error (s.e.). The F2 scaling factor is equal to  $1-Q2$  (where  $Q2$  is the AIS probability between the two populations)
- `f3.values` A data frame with  $npops(npops-1)(npops-2)/2$  rows and 1 (or 4 if `blockjackknife` is TRUE) columns containing estimates of the f3-statistics over all the SNPs and if `blockjackknife=TRUE`, the estimated block-jackknife and standard error (s.e.) and Z-score measuring the deviation of the f3-statistics from 0 in units of s.e.
- `f3star.values` A data frame with  $npops(npops-1)(npops-2)/2$  rows and 1 (or 4 if `blockjackknife` is TRUE) columns containing estimates of the scaled f3-statistics over all the SNPs and if `blockjackknife=TRUE`, the estimated block-jackknife and standard error (s.e.) and Z-score measuring the deviation of the f3-statistics from 0 in units of s.e. The F3 scaling factor is equal to  $1-Q1$  (where  $Q1$  is the AIS probability within the target population, i.e., population C for  $F3(C;A,B)$ )
- `f4.values` A data frame with  $npops(npops-1)(npops-2)(npops-3)/8$  rows and 1 (or 4 if `blockjackknife` is TRUE) columns containing estimates of the f4-statistics over all the SNPs and if `blockjackknife=TRUE`, the estimated block-jackknife and standard error (s.e.) and Z-score measuring the deviation of the f4-statistics from 0 in units of s.e.
- `Dstat.values` A data frame with  $npops(npops-1)(npops-2)(npops-3)/8$  rows and 1 (or 4 if `blockjackknife` is TRUE) columns containing estimates of the D-statistics (scaled f4-statistics) over all the SNPs and if `blockjackknife=TRUE`, the estimated block-jackknife and standard error (s.e.) and Z-score measuring the deviation of the f3-statistics from 0 in units of s.e. For

a given quadruplet (A,B;C,D), the parameter D corresponds to  $F4(A,B;C,D)$  scaled by  $(1-Q2(A,B))*(1-Q2(C,D))$  where  $Q2(X,Y)$  is the is the AIS probability between the X and Y populations.

`F4.bjack.samples` If `blockjackknife=TRUE` and options `return.F4.blockjackknife.samples` is activated in `compute.fstats`, a matrix with  $n\text{pops}(n\text{pops}-1)(n\text{pops}-2)(n\text{pops}-3)/8$  rows and `nblock.jackknife.samples` columns

`comparisons` A list containing matrices with population names associated to the different test comparisons (e.g., the "F2" elements of the list is a  $n\text{pop}(n\text{pop}-1)/2$  rows x 2 columns with each row containing the name of the two populations compared)

`Q.matrix` The estimated error covariance matrix for all the F2 and F3 estimates (required by graph fitting functions to compute graph scores)

`heterozygosities` A data frame with `npop` rows and 1 (or 3 if `blockjackknife` is TRUE) columns containing estimates of the within population heterozygosities  $(1-Q1)$  over all the SNPs and if `blockjackknife=TRUE`, the estimated block-jackknife and standard error (s.e.)

`blockjackknife` A logical indicating whether block-jackknife estimates of standard errors are available (TRUE) or not (FALSE)

### See Also

To generate pairwise object, see [compute.pairwiseFST](#)

---

`generate.graph.params` *Generate a graph parameter object to fit admixture graph to observed fstats*

---

### Description

Generate a graph parameter object to fit admixture graph to observed fstats

### Usage

```
generate.graph.params(
  graph,
  fstats = NULL,
  popref = NULL,
  outfileprefix = NULL,
  verbose = TRUE
)
```

### Arguments

<code>graph</code>	A three columns matrix containing graph information in a simple format (see details)
<code>fstats</code>	A fstats object containing estimates of fstats
<code>popref</code>	Reference population of the fstats basis used to fit the graph.

outfileprefix The prefix of the dot file that will represent the graph (with extension ".dot"). If NULL, no graph file generated

verbose If TRUE some information is printed on the terminal

## Details

The graph needs to be specified by a three column (character) matrix corresponding for each edge (wether admixed or not) to i) the child node; ii) the parent node; iii) the admixture proportion. For non-admixed edge, the third column must be blank. An admixed node should be referred two times as a child node with two different parent node and two different admixture proportions coded as alpha and (1-alpha) (Note that the parentheses are mandatory) if alpha is the name of the admixture proportion. The root is automatically identified as a node only present in the parent node column. Several checks are made within the function but it is recommended to check the graph by plotting the resulting dot file named outfileprefix.dot using for instance the grViz() from the DiagrammeR package that may be called directly with plot or with the dot program (e.g., dot -Tpng inputgraph.dot in terminal). Note that the dot file may be easily customized (e.g., to change leave color, parameter names...). The fstats object should be of class fstats (see help(fstats) for details) containing estimates of F2 and F3 statistics and block jackknife as generated with the `compute.fstats` function with computeF3 set to TRUE. If no fstats object is provided, only graph parameters will be generated.

## Value

An object of class graph.params (see help(graph.params) for details)

## See Also

The object may be used to estimate graph parameters with the function `fit.graph` or to generate files for the qpGraph software with `graph.params2qpGraphFiles`. See also `graph.params2symbolic.fstats` to obtain symbolic representation of Fstats.

## Examples

```
graph=rbind(c("P1", "P7", ""), c("P2", "s1", ""), c("P3", "s2", ""), c("P6", "S", ""),
            c("S", "s1", "a"), c("S", "s2", "(1-a)"), c("s2", "P8", ""), c("s1", "P7", ""),
            c("P4", "P9", ""), c("P5", "P9", ""), c("P7", "P8", ""),
            c("P8", "R", ""), c("P9", "R", ""))
graph.params=generate.graph.params(graph)
plot(graph.params)
##NOTE: this calls grViz from DiagrammeR which cannot easily be plotted
#within pdf or other device. To that end the easiest is to output
#the graph in a dot file (using the outfileprefix argument) and
#then to use the dot program out of R in a terminal: dot -Tpng inputgraph.dot
```

---

```
generate.jackknife.blocks
```

*Generate block coordinates for block-jackknife*

---

### Description

Generate block coordinates for block-jackknife

### Usage

```
generate.jackknife.blocks(x, nsnp.per.bjack.block, verbose = TRUE)
```

### Arguments

x	A pooldata or countdata object containing SNP positions (snp.info slot)
nsnp.per.bjack.block	Number of consecutive SNPs of each block-jackknife block
verbose	If TRUE extra information is printed on the terminal

### Value

A list with the two following elements:

1. "blocks.det": A matrix with three columns containing for each identified block (in row) the index of the start SNP, the index of the end SNP and the block Size in bp
2. "snp.block.id": A vector containing the blocks assigned to each SNP eligible for block-Jackknife (non eligible SNPs are assigned NA)
3. "nblocks": A scalar corresponding to the number of blocks
4. "nsnps": Number of SNPs eligible for block-jackknife 'i.e., included in one block

---

```
genobypass2countdata Convert BayPass allele count input files into a countdata object
```

---

### Description

Convert BayPass allele count input files into a countdata object

### Usage

```
genobypass2countdata(
  genobypass.file = "",
  snp.pos = NA,
  popnames = NA,
  min.indgeno.per.pop = -1,
  min.maf = -1,
  verbose = TRUE
)
```

**Arguments**

genobypass.file	The name (or a path) of the BayPass allele count file (see the BayPass manual <a href="http://www1.montpellier.inra.fr/CBGP/software/bypass/">http://www1.montpellier.inra.fr/CBGP/software/bypass/</a> )
snp.pos	An optional two column matrix with nsnp rows containing the chromosome (or contig/scaffold) of origin and the position of each markers
popnames	A character vector with the names of pool
min.indgeno.per.pop	Minimal number of overall counts required in each population. If at least one pop is not genotyped for at least min.indgeno.per.pop (haploid) individual, the position is discarded
min.maf	Minimal allowed Minor Allele Frequency (computed from the ratio overall counts for the reference allele over the overall number of (haploid) individual genotyped)
verbose	If TRUE extra information is printed on the terminal

**Details**

Information on SNP position is only required for some graphical display or to carried out block-jackknife sampling estimation of confidence intervals. If no mapping information is given (default), SNPs will be assumed to be ordered on the same chromosome and separated by 1 bp. As blocks are defined with a number of consecutive SNPs (rather than a length), the latter assumption has actually no effect (except in the reported estimated block sizes in Mb).

**Value**

A countdata object containing 6 elements:

1. "refallele.count": a matrix (nsnp rows and npops columns) with the allele counts for the reference allele
2. "total.count": a matrix (nsnp rows and npops columns) with the total number of counts (i.e., twice the number of genotyped individual for diploid species and autosomal markers)
3. "snp.info": a matrix with nsnp rows and four columns containing respectively the contig (or chromosome) name (1st column) and position (2nd column) of the SNP; the allele taken as reference in the refallele.count matrix (3rd column); and the alternative allele (4th column)
4. "popnames": a vector of length npops containing the names of the pops
5. "nsnp": a scalar corresponding to the number of SNPs
6. "npops": a scalar corresponding to the number of populations

**Examples**

```
make.example.files(writing.dir=tempdir())
pooldata=popsync2pooldata(sync.file=paste0(tempdir(),"/ex.sync.gz"),poolsizes=rep(50,15))
pooldata2genobypass(pooldata=pooldata,writing.dir=tempdir())
##NOTE: This example is just for the sake of illustration as it amounts
##to interpret read count as allele count which must not be done in practice!
countdata=genobypass2countdata(genobypass.file=paste0(tempdir(),"/genobypass"))
```

---

genobaypass2pooldata *Convert BayPass read count and haploid pool size input files into a pooldata object*

---

## Description

Convert BayPass read count and haploid pool size input files into a pooldata object

## Usage

```
genobaypass2pooldata(
  genobaypass.file = "",
  poolsize.file = "",
  snp.pos = NA,
  poolnames = NA,
  min.cov.per.pool = -1,
  max.cov.per.pool = 1e+06,
  min.maf = -1,
  verbose = TRUE
)
```

## Arguments

genobaypass.file	The name (or a path) of the BayPass read count file (see the BayPass manual <a href="http://www1.montpellier.inra.fr/CBGP/software/baypass/">http://www1.montpellier.inra.fr/CBGP/software/baypass/</a> )
poolsize.file	The name (or a path) of the BayPass (haploid) pool size file (see the BayPass manual <a href="http://www1.montpellier.inra.fr/CBGP/software/baypass/">http://www1.montpellier.inra.fr/CBGP/software/baypass/</a> )
snp.pos	An optional two column matrix with nsnp rows containing the chromosome (or contig/scaffold) of origin and the position of each markers
poolnames	A character vector with the names of pool
min.cov.per.pool	Minimal allowed read count (per pool). If at least one pool is not covered by at least min.cov.perpool reads, the position is discarded
max.cov.per.pool	Maximal allowed read count (per pool). If at least one pool is covered by more than min.cov.perpool reads, the position is discarded
min.maf	Minimal allowed Minor Allele Frequency (computed from the ratio overall read counts for the reference allele over the read coverage)
verbose	If TRUE extra information is printed on the terminal

## Details

Information on SNP position is only required for some graphical display or to carried out block-jackknife sampling estimation of confidence intervals. If no mapping information is given (default),

SNPs will be assumed to be ordered on the same chromosome and separated by 1 bp. As blocks are defined with a number of consecutive SNPs (rather than a length), the latter assumption has actually no effect (except in the reported estimated block sizes in Mb).

### Value

A pooldata object containing 7 elements:

1. "refallele.readcount": a matrix with nsnp rows and npools columns containing read counts for the reference allele (chosen arbitrarily) in each pool
2. "readcoverage": a matrix with nsnp rows and npools columns containing read coverage in each pool
3. "snp.info": a matrix with nsnp rows and four columns containing respectively the contig (or chromosome) name (1st column) and position (2nd column) of the SNP; the allele taken as reference in the refallele.readcount matrix (3rd column); and the alternative allele (4th column)
4. "poolsizes": a vector of length npools containing the haploid pool sizes
5. "poolnames": a vector of length npools containing the names of the pools
6. "nsnp": a scalar corresponding to the number of SNPs
7. "npools": a scalar corresponding to the number of pools

### Examples

```
make.example.files(writing.dir=tempdir())
pooldata=popsync2pooldata(sync.file=paste0(tempdir(),"/ex.sync.gz"),poolsizes=rep(50,15))
pooldata2genobypass(pooldata=pooldata,writing.dir=tempdir())
pooldata=genobypass2pooldata(genobypass.file=paste0(tempdir(),"/genobypass"),
                             poolsize.file=paste0(tempdir(),"/poolsize"))
```

---

genoseestim2pooldata *Convert SelEstim read count input files into a pooldata object*

---

### Description

Convert SelEstim read count input files into a pooldata object

### Usage

```
genoseestim2pooldata(
  genoseestim.file = "",
  poolnames = NA,
  min.cov.per.pool = -1,
  max.cov.per.pool = 1e+06,
  min.maf = -1,
  nlines.per.readblock = 1e+06,
  verbose = TRUE
)
```

**Arguments**

<code>genoselestim.file</code>	The name (or a path) of the SelEstim read count file (see the SelEstim manual <a href="http://www1.montpellier.inra.fr/CBGP/software/selestim/">http://www1.montpellier.inra.fr/CBGP/software/selestim/</a> )
<code>poolnames</code>	A character vector with the names of pool
<code>min.cov.per.pool</code>	Minimal allowed read count (per pool). If at least one pool is not covered by at least <code>min.cov.per.pool</code> reads, the position is discarded
<code>max.cov.per.pool</code>	Maximal allowed read count (per pool). If at least one pool is covered by more than <code>min.cov.per.pool</code> reads, the position is discarded
<code>min.maf</code>	Minimal allowed Minor Allele Frequency (computed from the ratio overall read counts for the reference allele over the read coverage)
<code>nlines.per.readblock</code>	Number of Lines read simultaneously. Should be adapted to the available RAM.
<code>verbose</code>	If TRUE extra information is printed on the terminal

**Value**

A pooldata object containing 7 elements:

1. "refallele.readcount": a matrix with `nsnp` rows and `npools` columns containing read counts for the reference allele (chosen arbitrarily) in each pool
2. "readcoverage": a matrix with `nsnp` rows and `npools` columns containing read coverage in each pool
3. "snp.info": a matrix with `nsnp` rows and four columns containing respectively the contig (or chromosome) name (1st column) and position (2nd column) of the SNP; the allele taken as reference in the `refallele.readcount` matrix (3rd column); and the alternative allele (4th column)
4. "poolsizes": a vector of length `npools` containing the haploid pool sizes
5. "poolnames": a vector of length `npools` containing the names of the pools
6. "nsnp": a scalar corresponding to the number of SNPs
7. "npools": a scalar corresponding to the number of pools

**Examples**

```
make.example.files(writing.dir=tempdir())
pooldata=popsync2pooldata(sync.file=paste0(tempdir(),"/ex.sync.gz"),poolsizes=rep(50,15))
pooldata2genoselestim(pooldata=pooldata,writing.dir=tempdir())
pooldata=genoselestim2pooldata(genoselestim.file=paste0(tempdir(),"/genoselestim"))
```



---

genotreemix2countdata *Convert allele count input files from the Treemix program into a countdata object*

---

### Description

Convert allele count input files from the Treemix program into a countdata object

### Usage

```
genotreemix2countdata(  
  genotreemix.file = "",  
  snp.pos = NA,  
  min.indgeno.per.pop = -1,  
  min.maf = -1,  
  verbose = TRUE  
)
```

### Arguments

genotreemix.file	The name (or a path) of the Treemix allele count file (see the Treemix manual <a href="https://bitbucket.org/nygcresearch/treemix/wiki/Home">https://bitbucket.org/nygcresearch/treemix/wiki/Home</a> )
snp.pos	An optional two column matrix with nsnp rows containing the chromosome (or contig/scaffold) of origin and the position of each markers
min.indgeno.per.pop	Minimal number of overall counts required in each population. If at least one pop is not genotyped for at least min.indgeno.per.pop (haploid) individual, the position is discarded
min.maf	Minimal allowed Minor Allele Frequency (computed from the ratio overall counts for the reference allele over the overall number of (haploid) individual genotyped)
verbose	If TRUE extra information is printed on the terminal

### Details

Information on SNP position is only required for some graphical display or to carried out block-jackknife sampling estimation of confidence intervals. If no mapping information is given (default), SNPs will be assumed to be ordered on the same chromosome and separated by 1 bp. As blocks are defined with a number of consecutive SNPs (rather than a length), the latter assumption has actually no effect (except in the reported estimated block sizes in Mb).

### Value

A countdata object containing 6 elements:

1. "refallele.count": a matrix (nsnp rows and npops columns) with the allele counts for the reference allele
2. "total.count": a matrix (nsnp rows and npops columns) with the total number of counts (i.e., twice the number of genotyped individual for diploid species and autosomal markers)
3. "snp.info": a matrix with nsnp rows and four columns containing respectively the contig (or chromosome) name (1st column) and position (2nd column) of the SNP; the allele taken as reference in the refallele.count matrix (3rd column); and the alternative allele (4th column)
4. "popnames": a vector of length npops containing the names of the pops
5. "nsnp": a scalar corresponding to the number of SNPs
6. "npops": a scalar corresponding to the number of populations

### Examples

```
make.example.files(writing.dir=tempdir())
pooldata=popsync2pooldata(sync.file=paste0(tempdir(),"/ex.sync.gz"),poolsizes=rep(50,15))
##NOTE: This example is just for the sake of illustration as it amounts
##to interpret read count as allele count which must not be done in practice!
dum=matrix(paste(pooldata@refallele.readcount,
  pooldata@readcoverage-pooldata@refallele.readcount,sep=","),
  ncol=pooldata@npops)
colnames(dum)=pooldata@poolnames
write.table(dum,file=paste0(tempdir(),"/genotremix"),quote=FALSE,row.names=FALSE)
countdata=genotremix2countdata(genotremix.file=paste0(tempdir(),"/genotremix"))
```

---

graph.builder

*Implement a graph builder heuristic by successively adding leaves to an initial graph*

---

### Description

Implement a graph builder heuristic by successively adding leaves to an initial graph

### Usage

```
graph.builder(
  x,
  leaves.to.add,
  fstats,
  heap.dbic = 6,
  max.heap.size = 25,
  verbose = TRUE,
  ...
)
```

**Arguments**

x	An object (or list of objects) of class graph.params or fitted.graph (see details)
leaves.to.add	Names of the leaves to successively add (in the given order)
fstats	Object of class fstats that contains estimates of the fstats (see compute.fstats)
heap.dbic	Maximal BIC distance from the best graph to be kept in the heap (heap.dbic=6 by default)
max.heap.size	Maximal number of graphs stored in the heap (max.heap.size=25 by default)
verbose	If TRUE extra information is printed on the terminal
...	Some parameters to be passed the function add.leaf called internally

**Details**

The input object x needs to be of class graph.params as generated by the function generate.graph.params; or fitted.graph as generated by the functions fit.graph, add.leaf (in the output list element named "fitted.graphs.list") or rooted.nj.builder (in the output element named "best.rooted.tree"). This is to ensure that the matrix describing the structure of the graph (graph slot of these objects) is valid (note that it can be plotted for checks). Hence graph.params objects may have been generated without fstats information (that should be supplied independently to the add.leaf function to obtain information on the fstats involving the candidate leaf defined with the leaf.to.add argument). The functions successively add each leaf given in the leaves.to.add vector to the list of fitted graph stored in a heap using the function add.leaf. For the first iteration (i.e., first tested leaf) the heap consists of the input graph or list of graph x. At each iteration, the function add.leaf is used to test the candidate leaf to each graph from the current heap in turn. A new heap of graphs is then built by each time including the fitted graphs with a BIC less than heap.dbic larger than the best resulting graphs (treating each graph independently). If the final number of graphs in the heap is larger than max.heap.size, the max.heap.size graphs with the lowest BIC are kept in the heap. After testing the latest leaf, graphs with a BIC larger than heap.dbic units of the best graph are discarded from the final list of graphs. In practice, it is recommended to test different orders of inclusion of the leaves (as specified in the vector leaves.to.add)

**Value**

A list with the following elements:

1. "n.graphs": The final number of fitted graphs
2. "fitted.graphs.list": a list of fitted.graph objects (indexed from 1 to n.graphs and in the same order as the list "graphs") containing the results of fitting of each graph.
3. "best.fitted.graph": The graph (object of class fitted.graph) with the minimal BIC (see function fit.graph) among all the graphs within fitted.graphs.list
4. "bic": a vector of the n.graphs BIC (indexed from 1 to n.graphs and in the same order as the "fitted.graphs.list" list) (see fit.graph details for the computation of the scores).

**See Also**

see [fit.graph](#), [generate.graph.params](#) and [add.leaf](#).

---

graph.params-class     *S4 class to represent a population tree or admixture graph and its underlying parameter.*

---

## Description

S4 class to represent a population tree or admixture graph and its underlying parameter.

## Details

The graph is specified by a three column (character) matrix giving for each edge (whether admixed or not) to i) the child node; ii) the parent node; iii) the admixture proportion. For non-admixed edge, the third column must be blank. An admixed node should be referred two times as a child node with two different parent node and two different admixture proportions coded as alpha and (1-alpha) (parentheses are mandatory) if alpha is the name of the parameter for admixture proportion. The dot.graph element allows to plot the graph using grViz() from the DiagrammeR package or with the dot program after writing the files (e.g., dot -Tpng inputgraph.dot in terminal). Note that the dot file may be customized (e.g., to change leave color, parameter names...).

## Slots

graph The graph in 3 column format (see details)

dot.graph The graph in dot format

is.admgraph If FALSE the graph is binary tree (i.e., no admixture events), if TRUE the graph is an admixture graph

n.leaves Number of leaves of the graph

leaves Name of the leaves

root.name Name of the root

n.nodes Number of nodes (including root)

nodes.names Name of the nodes

n.edges Number of edges (including admixture edges)

edges.names Names of the edges (coded as "Parent node Name"<->"Child node Name")

n.adm.nodes Number of admixed nodes (=0 if is.admgraph=FALSE). This is also the number of admixed parameters since only two-ways admixture are assumed for a given node

adm.params.names Names of the admixed parameters

graph.matrix The graph incidence matrix consisting of n.leaves rows and n.edges columns. The elements of the matrix are the weights of each edge (in symbolic representation) for the different possible paths from the leaves to the graph root.

root.edges.idx Indexes of the graph.matrix columns associated to the (two) edges connected to the root

f2.target The (n.leaves-1) stats F2 involving popref (i.e., of the form F2(popref;pop))

- f2.target.pops A matrix of  $(n.leaves-1)$  rows and 2 columns containing the names of populations of the F2 stats. The first column is by construction always popref. The order is the same as in f2.target
- f3.target The  $(n.leaves-1)(n.leaves-2)/2$  stats F3 involving popref as a target (i.e., of the form F3(popref;popA,popB))
- f3.target.pops A matrix of  $(n.leaves-1)(n.leaves-2)/2$  rows and 3 columns containing the name of popref in the first column and the names of the two populations involved in the F3 stats. The order is the same as in f3.target
- popref The name of the reference population defining the fstats basis
- f.Qmat A square matrix of rank  $n.leaves(n.leaves-1)/2$  corresponding to the error covariance matrix of the F2 and F3 estimates
- Het Estimated leave heterozygosities (if present in the fstats object)

### See Also

To generate graph.params object, see [generate.graph.params](#). The object may be used to estimate graph parameters with the function [fit.graph](#) or to generate files for the qpGraph software with [graph.params2qpGraphFiles](#). See also [graph.params2symbolic.fstats](#) to obtain symbolic representation of Fstats from the matrix "Omega".

---

graph.params2qpGraphFiles

*Generate files for the qpGraph software from a graph.params object*

---

### Description

Generate files for the qpGraph software from a graph.params object

### Usage

```
graph.params2qpGraphFiles(  
  graph.params,  
  outfileprefix = "out",  
  n.printed.dec = 4,  
  verbose = TRUE  
)
```

### Arguments

- graph.params An object of class graph.params containing graph information with Fstats information (see the function generate.graph.params)
- outfileprefix The prefix of the qpGraph files
- n.printed.dec Number of decimal to be printed (if not enough may lead to fatalx error in qp-Graph)
- verbose If TRUE extra information is printed on the terminal

**Details**

This function generates the three files required by qpGraph: i) a file named outfileprefix.graph containing the graph in appropriate format; ii) a file named outfileprefix.fstats file containing the fstats estimates of fstats (and their covariance); iii) a file named outfileprefix.parqpGraph containing essential parameter information to run qpGraph (this may be edited by hand if other options are needed). The qpGraph software may then be run using the following options -p outfileprefix.parqpGraph -g outfileprefix.graph -o out.ggg -d out.dot.

**Value**

The three files described in the details section

**See Also**

To generate graph.params object, see [generate.graph.params](#)

---

graph.params2symbolic.fstats

*Provide a symbolic representation of all the F-statistics and the model system of equations*

---

**Description**

Provide a symbolic representation of all the F-statistics and the model system of equations

**Usage**

```
graph.params2symbolic.fstats(x, outfile = NULL)
```

**Arguments**

x	An object of class graph.params containing graph information and relevant Fstats estimates (see the function generate.graph.params)
outfile	The file where to print the equations (default=NULL, equations are not printed in a file)

**Value**

A list with the following elements:

1. "model.matrix": A symbolic representation of the matrix M relating the basis F-statistics and graph edge length as  $F=M*b$  where F is the vector of the basis Fstats (row names of model.matrix M) and b is the vector of graph edges (column names of model.matrix M).
2. "omega": A symbolic representation of the scaled covariance matrix of allele frequency with edge names and admixture parameter names as specified in the edges.names and adm.params.names slot of the input graph.params object x

3. "F2.equations": A symbolic representation of the  $nleaves(nleaves-1)/2$  different F2 as a function of graph parameters
4. "F3.equations": A symbolic representation of the  $nleaves(nleaves-1)(nleaves-2)/2$  different F3 as a function of graph parameters
5. "F4.equations": A symbolic representation of the  $npops(npops-1)(npops-2)(npops-3)/8$  different F4 as a function of graph parameters

### See Also

To generate a graph.params object, see [generate.graph.params](#).

### Examples

```
graph=rbind(c("P1", "P7", ""), c("P2", "s1", ""), c("P3", "s2", ""), c("P6", "S", ""),
            c("S", "s1", "a"), c("S", "s2", "(1-a)"), c("s2", "P8", ""), c("s1", "P7", ""),
            c("P4", "P9", ""), c("P5", "P9", ""), c("P7", "P8", ""),
            c("P8", "R", ""), c("P9", "R", ""))
graph.params=generate.graph.params(graph)
graph.equations=graph.params2symbolic.fstats(graph.params)
```

---

heatmap, pairwiseFST-method

*Show pairwiseFST object*

---

### Description

Show pairwiseFST object

### Usage

```
## S4 method for signature 'pairwiseFST'
heatmap(
  x,
  Rowv = NULL,
  Colv = if (symm) "Rowv" else NULL,
  distfun = as.dist,
  hclustfun = hclust,
  reorderfun = function(d, w) reorder(d, w),
  add.expr,
  symm = FALSE,
  revC = identical(Colv, "Rowv"),
  scale = c("row", "column", "none"),
  na.rm = TRUE,
  margins = c(5, 5),
  ColSideColors,
  RowSideColors,
  cexRow = 0.2 + 1/log10(nrow(x@PairwiseFSTmatrix)),
```

```

    cexCol = 0.2 + 1/log10(ncol(x@PairwiseFSTmatrix)),
    labRow = NULL,
    labCol = NULL,
    main = NULL,
    xlab = NULL,
    ylab = NULL,
    keep.dendro = FALSE,
    verbose = getOption("verbose"),
    ...
)

```

### Arguments

<code>x</code>	Object of class <code>pairwise fst</code>
<code>Rowv</code>	determines if and how the row dendrogram should be computed and reordered. Either a dendrogram or a vector of values used to reorder the row dendrogram or NA to suppress any row dendrogram (and reordering) or by default, NULL, see ‘Details’ below.
<code>Colv</code>	determines if and how the column dendrogram should be reordered. Has the same options as the <code>Rowv</code> argument above and additionally when <code>x</code> is a square matrix, <code>Colv = "Rowv"</code> means that columns should be treated identically to the rows (and so if there is to be no row dendrogram there will not be a column one either).
<code>distfun</code>	function used to compute the distance (dissimilarity) between both rows and columns. Defaults to <code>as.dist</code> .
<code>hclustfun</code>	function used to compute the hierarchical clustering when <code>Rowv</code> or <code>Colv</code> are not dendrograms. Defaults to <code>hclust</code> . Should take as argument a result of <code>distfun</code> and return an object to which <code>as.dendrogram</code> can be applied.
<code>reorderfun</code>	function( <code>d</code> , <code>w</code> ) of dendrogram and weights for reordering the row and column dendrograms. The default uses <code>reorder.dendrogram</code> .
<code>add.expr</code>	expression that will be evaluated after the call to <code>image</code> . Can be used to add components to the plot.
<code>symm</code>	logical indicating if <code>x</code> should be treated symmetrically; can only be true when <code>x</code> is a square matrix.
<code>revC</code>	logical indicating if the column order should be reversed for plotting, such that e.g., for the symmetric case, the symmetry axis is as usual.
<code>scale</code>	character indicating if the values should be centered and scaled in either the row direction or the column direction, or none. The default is "row" if <code>symm</code> false, and "none" otherwise.
<code>na.rm</code>	logical indicating whether NA's should be removed.
<code>margins</code>	numeric vector of length 2 containing the margins (see <code>par(mar = *)</code> ) for column and row names, respectively.
<code>ColSideColors</code>	(optional) character vector of length <code>ncol(x)</code> containing the color names for a horizontal side bar that may be used to annotate the columns of <code>x</code> .



RowSideColors	(optional) character vector of length nrow(x) containing the color names for a vertical side bar that may be used to annotate the rows of x.
cexRow, cexCol	positive numbers, used as cex.axis in for the row or column axis labeling. The defaults currently only use number of rows or columns, respectively.
labRow, labCol	character vectors with row and column labels to use; these default to rownames(x) or colnames(x), respectively.
main, xlab, ylab	main, x- and y-axis titles; defaults to none.
keep.dendro	logical indicating if the dendrogram(s) should be kept as part of the result (when Rowv and/or Colv are not NA).
verbose	logical indicating if information should be printed.
...	additional arguments passed on to image, e.g., col specifying the colors.

---

is.countdata	<i>Check countdata objects</i>
--------------	--------------------------------

---

**Description**

Check countdata objects

**Usage**

```
is.countdata(x)
```

**Arguments**

x	The name of the object to be tested
---	-------------------------------------

---

is.fitted.graph	<i>Check fitted.graph objects</i>
-----------------	-----------------------------------

---

**Description**

Check fitted.graph objects

**Usage**

```
is.fitted.graph(x)
```

**Arguments**

x	Object to be tested
---	---------------------

---

is.fstats	<i>Check fstats objects</i>
-----------	-----------------------------

---

**Description**

Check fstats objects

**Usage**

```
is.fstats(x)
```

**Arguments**

x	The name of the object to be tested
---	-------------------------------------

---

is.graph.params	<i>Check graph.params objects</i>
-----------------	-----------------------------------

---

**Description**

Check graph.params objects

**Usage**

```
is.graph.params(x)
```

**Arguments**

x	The name (or a path) of the graph.params objet
---	--

---

is.pairwisefst	<i>Check pairwisefst objects</i>
----------------	----------------------------------

---

**Description**

Check pairwisefst objects

**Usage**

```
is.pairwisefst(x)
```

**Arguments**

x	The name (or a path) of the pairwisefst object
---	--

---

is.pooldata	<i>Check pooldata objects</i>
-------------	-------------------------------

---

**Description**

Check pooldata objects

**Usage**

```
is.pooldata(x)
```

**Arguments**

x	The name of the object to be tested
---	-------------------------------------

---

make.example.files	<i>Create example files</i>
--------------------	-----------------------------

---

**Description**

Write in the current directory example files corresponding to a sync (as obtained when parsing mpileup files with PoPoolation) and vcf (as obtained when parsing mpileup files with VarScan) gzipped files

**Usage**

```
make.example.files(writing.dir = "")
```

**Arguments**

writing.dir	Directory where to copy example files (e.g., set writing.dir=getwd() to copy in the current working directory)
-------------	--

**Examples**

```
make.example.files(writing.dir=tempdir())
```

---

pairwisefst-class      *S4 class to represent a pairwise Fst results obtained with the compute.pairwiseFST*

---

### Description

S4 class to represent a pairwise Fst results obtained with the compute.pairwiseFST

### Slots

values A data frame with  $n_{pop}*(n_{pop}-1)/2$  rows and 3 (or 7 if blockjackknife is TRUE) columns containing for both the Fst and Q2, estimates over all the SNPs and if blockjackknife=TRUE, the estimated block-jackknife and standard error (s.e.). The seventh (or third if blockjackknife=FALSE) column gives the number of SNPs.

PairwiseFSTmatrix A  $n_{pop} \times n_{pop}$  matrix containing the pairwise FST estimates

PairwiseSnpFST A matrix ( $n_{snp}$  rows and  $n_{pops}$  columns) with read count data for the reference allele

PairwiseSnpQ1 A matrix ( $n_{snp}$  rows and  $n_{pops}$  columns) with overall read coverage

PairwiseSnpQ2 A matrix ( $n_{snp}$  rows and 4 columns) detailing for each SNP, the chromosome (or scaffold), the position, allele 1 and allele 2

blockjackknife A logical indicating whether block-jackknife estimates of standard errors are available (TRUE) or not (FALSE)

### See Also

To generate pairwise object, see [compute.pairwiseFST](#)

---

plot,fitted.graph-method  
*plot pairwisefst object*

---

### Description

plot pairwisefst object

### Usage

```
## S4 method for signature 'fitted.graph'
plot(x, y)
```

### Arguments

x                      Object of class fitted.graph  
y                      dummy argument

---

plot,fstats-method     *plot fstats object*

---

**Description**

plot fstats object

**Usage**

```
## S4 method for signature 'fstats'  
plot(x, y, ...)
```

**Arguments**

x	Object of class fstats
y	dummy argument
...	Other arguments to be passed to plot_fstats

**See Also**

see [plot\\_fstats](#) for details on plot\_fstats arguments

---

plot,graph.params-method  
*plot graph in graph.params object*

---

**Description**

plot graph in graph.params object

**Usage**

```
## S4 method for signature 'graph.params'  
plot(x, y)
```

**Arguments**

x	Object of class fitted.graph
y	dummy argument

---

```
plot, pairwisefst-method
      plot pairwisefst object
```

---

**Description**

plot pairwisefst object

**Usage**

```
## S4 method for signature 'pairwisefst'
plot(x, y, ...)
```

**Arguments**

```
x          Object of class pairwisefst
y          dummy argument
...        Some arguments to be passed to plot_fstats
```

**See Also**

see [plot\\_fstats](#) for details on plot\_fstats arguments

---

```
plot_fstats      Plot F2, F3, F3star, F4, D or pairwise Fst values with their Confidence Intervals
```

---

**Description**

Plot F2, F3, F3star, F4, D or pairwise Fst values with their Confidence Intervals

**Usage**

```
plot_fstats(
  x,
  stat.name = "F2",
  ci.perc = 95,
  value.range = c(NA, NA),
  pop.sel = NA,
  pop.f3.target = NA,
  highlight.signif = TRUE,
  main = stat.name,
  ...
)
```

**Arguments**

<code>x</code>	An object of class <code>fstats</code> (to plot F2, F3 or F4 statistics) or <code>pairwisefst</code> (to plot pairwise <code>fst</code> )
<code>stat.name</code>	For <code>fstats</code> object, the name of the stat (either F2, F3, F3star, F4 or Dstat)
<code>ci.perc</code>	Percentage of the Confidence Interval in number of standard errors (default=95%)
<code>value.range</code>	Range of test values (x-axis) to be plotted (default=NA,NA: i.e., all test values are plotted)
<code>pop.sel</code>	Only plot test values involving these populations (default=NA: i.e., all test values are plotted)
<code>pop.f3.target</code>	For F3-statistics, only plot F3 involving <code>pop.f3.target</code> as a target
<code>highlight.signif</code>	If TRUE highlight significant tests in red (see details)
<code>main</code>	Main title of the plot (default= <code>stat.name</code> )
<code>...</code>	Some other graphical arguments to be passed

**Details**

Data will only be plotted if jackknife estimates of the estimator s.e. have been performed i.e. if the functions `compute.fstats` or `compute.pairwiseFST` were run with `nsnp.per.block>0`

**Value**

A plot of the Fstats of interest. Significant F3 statistics (i.e., showing formal evidence for admixture of the target population) are highlighted in red. Significant F4 statistics (i.e., showing formal evidence against treeness of the pop. quadruplet) are highlighted in red.

**See Also**

To generate `x` object, see [compute.pairwiseFST](#) (for `pairwisefst` object) or [compute.fstats](#) (for `fstats` object)

**Examples**

```
make.example.files(writing.dir=tempdir())
pooldata=popsync2pooldata(sync.file=paste0(tempdir(),"/ex.sync.gz"),
  poolsizes=rep(50,15),poolnames=paste0("P",1:15))
res.fstats=compute.fstats(pooldata,nsnp.per.bjack.block=25)
plot_fstats(res.fstats,stat.name="F3",cex=0.5)
plot_fstats(res.fstats,stat.name="F3",value.range=c(NA,0.001),
  pop.f3.target=c("P7","P5"),cex.axis=0.7)
plot_fstats(res.fstats,stat.name="F4",cex=0.5)
#allow to reduce the size of the test name (y-axis)
plot_fstats(res.fstats,stat.name="F4",cex=0.5,
  pop.sel=c("P1","P2","P3","P4","P5"))
plot_fstats(res.fstats,stat.name="F4",cex=0.5,
  pop.sel=c("P1","P2","P3","P4","P5"),highlight.signif=FALSE)
```

---

pooldata-class      *S4 class to represent a Pool-Seq data set.*

---

### Description

S4 class to represent a Pool-Seq data set.

### Slots

npools The number of pools  
 nsnp The number of SNPs  
 refallele.readcount A matrix (nsnp rows and npools columns) with read count data for the reference allele  
 readcoverage A matrix (nsnp rows and npools columns) with overall read coverage  
 snp.info A data frame (nsnp rows and 4 columns) detailing for each SNP, the chromosome (or scaffold), the position, Reference allele name and Alternate allele name (if available)  
 poolsizes A vector of length npools with the corresponding haploid pool sizes  
 poolnames A vector of length npools with the corresponding haploid pool names

### See Also

To generate pooldata object, see [vcf2pooldata](#), [popsync2pooldata](#), [genobypass2pooldata](#) and [genoselestim2pooldata](#)

---

pooldata.subset      *Create a subset of the pooldata object that contains Pool-Seq data as a function of pool and/or SNP indexes*

---

### Description

Create a subset of the pooldata object that contains Pool-Seq data as a function of pool and/or SNP indexes

### Usage

```
pooldata.subset(
  pooldata,
  pool.index = 1:pooldata@npools,
  snp.index = 1:pooldata@nsnp,
  min.cov.per.pool = -1,
  max.cov.per.pool = 1e+06,
  min.maf = -1,
  cov.qthres.per.pool = c(0, 1),
  return.snp.idx = FALSE,
  verbose = TRUE
)
```



### Arguments

pooldata	A pooldata object containing Pool-Seq information
pool.index	Indexes of the pools (at least two), that should be selected to create the new pooldata object (default=all the pools)
snp.index	Indexes of the SNPs (at least two), that should be selected to create the new pooldata object (default=all the SNPs)
min.cov.per.pool	Minimal allowed read count (per pool). If at least one pool is not covered by at least min.cov.perpool reads, the position is discarded
max.cov.per.pool	Maximal allowed read count (per pool). If at least one pool is covered by more than min.cov.perpool reads, the position is discarded
min.maf	Minimal allowed Minor Allele Frequency (computed from the ratio over all read counts for the reference allele over the read coverage)
cov.qthres.per.pool	A two-elements vector containing the minimal (qmin) and maximal (qmax) quantile coverage thresholds applied to each pools ( $0 \leq qmin < qmax \leq 1$ ). See details below
return.snp.idx	If TRUE, the row.names of the snp.info slot of the returned pooldata object are named as "rsx" where x is the index of SNP in the initial pooldata object (default=FALSE)
verbose	If TRUE return some information

### Details

This function allows subsetting a pooldata object by selecting only some pools and/or some SNPs (e.g., based on their position on the genome). Additional filtering steps on SNPs can be carried out on the resulting subset to discard SNP with low polymorphism or poorly or too highly covered. In addition, coverage criteria can be applied on a per-pool basis with the cov.qthres.per.pool argument. 'more specific SNP selection based on their positions on the genome or their characteristics. For instance if qmax=0.95, a position is discarded if in a given pool it has a number of reads higher than the 95-th percentile of the empirical coverage distribution in this same pool (defined over the SNPs selected by snp.index). Similarly, if qmax=0.05, a position is discarded if in a given pool it has a number of reads lower than the 5-th percentile of the empirical coverage distribution in this same pool. This mode of selection may be more relevant when considering pools with heterogeneous read coverages.

### Value

A pooldata object with 7 elements:

1. "refallele.readcount": a matrix with nsnp rows and npools columns containing read counts for the reference allele (chosen arbitrarily) in each pool
2. "readcoverage": a matrix with nsnp rows and npools columns containing read coverage in each pool

3. "snp.info": a matrix with nsnp rows and four columns containing respectively the contig (or chromosome) name (1st column) and position (2nd column) of the SNP; the allele in the reference assembly (3rd column); the allele taken as reference in the refallele matrix.readcount matrix (4th column); and the alternative allele (5th column)
4. "poolsizes": a vector of length npools containing the haploid pool sizes
5. "poolnames": a vector of length npools containing the names of the pools
6. "nsnp": a scalar corresponding to the number of SNPs
7. "npools": a scalar corresponding to the number of pools

### See Also

To generate pooldata object, see [vcf2pooldata](#), [popsync2pooldata](#)

### Examples

```
make.example.files(writing.dir=tempdir())
pooldata=popsync2pooldata(sync.file=paste0(tempdir(),"/ex.sync.gz"),poolsizes=rep(50,15))
subset.by.pools=pooldata.subset(pooldata,pool.index=c(1,2))
subset.by.snps=pooldata.subset(pooldata,snp.index=10:100)
subset.by.pools.and.snps=pooldata.subset(pooldata,pool.index=c(1,2),snp.index=10:100)
subset.by.pools.qcov.thr=pooldata.subset(pooldata,pool.index=1:8,cov.qthres.per.pool=c(0.05,0.95))
```

---

pooldata2genobaypass    *Convert a pooldata object into BayPass input files.*

---

### Description

Convert a pooldata object into BayPass allele read count and haploid pool size files. A file containing SNP details is also printed out. Options to generate sub-samples (e.g., for large number of SNPs) are also available.

### Usage

```
pooldata2genobaypass(
  pooldata,
  writing.dir = getwd(),
  prefix = "",
  subsamplesize = -1,
  subsamplingmethod = "thinning"
)
```

### Arguments

pooldata	A pooldata object containing Pool-Seq information (see <a href="#">vcf2pooldata</a> and <a href="#">popsync2pooldata</a> )
writing.dir	Directory where to create the files (e.g., set writing.dir=getwd() to copy in the current working directory)

**prefix** Prefix used for output file names  
**subsamplesize** Size of the sub-samples. If  $\leq 1$  (default), all the SNPs are considered in the output  
**subsamplingmethod** If sub-sampling is activated (argument `subsamplesize`), define the method used for subsampling that might be either i) "random" (A single data set consisting of randomly chosen SNPs is generated) or ii) "thinning", sub-samples are generated by taking SNPs one every  $n_{sub} = \text{floor}(n_{snp}/\text{subsamplesize})$  in the order of the map (a suffix ".subn" is added to each sub-sample files where n varies from 1 to  $n_{sub}$ ).

### Value

Files containing allele count (in BayPass format), haploid pool size (in BayPass format), and SNP details (as in the `snp.info` matrix from the `pooldata` object)

### See Also

To generate `pooldata` object, see [vcf2pooldata](#), [popsync2pooldata](#)

### Examples

```

make.example.files(writing.dir=tempdir())
pooldata=popsync2pooldata(sync.file=paste0(tempdir(),"/ex.sync.gz"),poolsizes=rep(50,15))
pooldata2genobaypass(pooldata=pooldata,writing.dir=tempdir())
    
```

---

`pooldata2genoseestim` *Convert a pooldata object into SelEstim input files.*

---

### Description

Convert a `pooldata` object into SelEstim allele read count. A file containing SNP details is also printed out. Options to generate sub-samples (e.g., for large number of SNPs) are also available.

### Usage

```

pooldata2genoseestim(
  pooldata,
  writing.dir = getwd(),
  prefix = "",
  subsamplesize = -1,
  subsamplingmethod = "thinning"
)
    
```

**Arguments**

pooldata	A pooldata object containing Pool-Seq information (see <a href="#">vcf2pooldata</a> and <a href="#">popsync2pooldata</a> )
writing.dir	Directory where to create the files (e.g., set <code>writing.dir=getwd()</code> to copy in the current working directory)
prefix	Prefix used for output file names
subsamplesize	Size of the sub-samples. If $\leq 1$ (default), all the SNPs are considered in the output
subsamplingmethod	If sub-sampling is activated (argument <code>subsamplesize</code> ), define the method used for subsampling that might be either i) "random" (A single data set consisting of randomly chosen SNPs is generated) or ii) "thinning", sub-samples are generated by taking SNPs one every <code>nsub=floor(nsnp/subsamplesize)</code> in the order of the map (a suffix ".subn" is added to each sub-sample files where n varies from 1 to nsub).

**Value**

Files containing allele count (in SelEstim Pool-Seq format) and SNP details (as in the `snp.info` matrix from the pooldata object)

**See Also**

To generate pooldata object, see [vcf2pooldata](#), [popsync2pooldata](#)

**Examples**

```
make.example.files(writing.dir=tempdir())
pooldata=popsync2pooldata(sync.file=paste0(tempdir(),"/ex.sync.gz"),poolsizes=rep(50,15))
pooldata2genoseestim(pooldata=pooldata,writing.dir=tempdir())
```

---

poolfstat

*PoolFstat*

---

**Description**

Functions for the computation of f- and D-statistics (estimation of  $F_{st}$ , Patterson's  $F_2$ ,  $F_3$ ,  $F_3^*$ ,  $F_4$  and D parameters) in population genomics studies from allele count or Pool-Seq read count data and for the fitting, building and visualization of admixture graphs. The package also includes several utilities to manipulate Pool-Seq data stored in standard format (e.g., such as 'vcf' files or 'rsync' files generated by the the 'PoPoolation' software) and perform conversion to alternative format (as used in the 'BayPass' and 'SelEstim' software). As of version 2.0, the package also includes utilities to manipulate standard allele count data (e.g., stored in TreeMix, BayPass and SelEstim format).

**Details**

Computing f-Statistics and building admixture graphs based on allele count or Pool-Seq read count data

---

popsync2pooldata      *Convert Popoolation Sync files into a pooldata object*

---

## Description

Convert Popoolation Sync files into a pooldata object

## Usage

```
popsync2pooldata(
  sync.file = "",
  poolsizes = NA,
  poolnames = NA,
  min.rc = 1,
  min.cov.per.pool = -1,
  max.cov.per.pool = 1e+06,
  min.maf = 0.01,
  noindel = TRUE,
  nlines.per.readblock = 1e+06,
  nthreads = 1
)
```

## Arguments

sync.file	The name (or a path) of the Popoolation sync file (might be in compressed format)
poolsizes	A numeric vector with haploid pool sizes
poolnames	A character vector with the names of pool
min.rc	Minimal allowed read count per base. Bases covered by less than min.rc reads are discarded and considered as sequencing error. For instance, if nucleotides A, C, G and T are covered by respectively 100, 15, 0 and 1 over all the pools, setting min.rc to 0 will lead to discard the position (the polymorphism being considered as tri-allelic), while setting min.rc to 1 (or 2, 3..14) will make the position be considered as a SNP with two alleles A and C (the only read for allele T being disregarded).
min.cov.per.pool	Minimal allowed read count (per pool). If at least one pool is not covered by at least min.cov.perpool reads, the position is discarded
max.cov.per.pool	Maximal allowed read count (per pool). If at least one pool is covered by more than min.cov.perpool reads, the position is discarded
min.maf	Minimal allowed Minor Allele Frequency (computed from the ratio overall read counts for the reference allele over the read coverage)
noindel	If TRUE, positions with at least one indel count are discarded

nlines.per.readblock  
 Number of Lines read simultaneously. Should be adapted to the available RAM.

nthreads  
 Number of available threads for parallelization of some part of the parsing (default=1, i.e., no parallelization)

### Value

A pooldata object containing 7 elements:

1. "refallele.readcount": a matrix with nsnp rows and npools columns containing read counts for the reference allele (chosen arbitrarily) in each pool
2. "readcoverage": a matrix with nsnp rows and npools columns containing read coverage in each pool
3. "snp.info": a matrix with nsnp rows and four columns containing respectively the contig (or chromosome) name (1st column) and position (2nd column) of the SNP; the allele taken as reference in the refallele.readcount matrix (3rd column); and the alternative allele (4th column)
4. "poolsizes": a vector of length npools containing the haploid pool sizes
5. "poolnames": a vector of length npools containing the names of the pools
6. "nsnp": a scalar corresponding to the number of SNPs
7. "npools": a scalar corresponding to the number of pools

### Examples

```
make.example.files(writing.dir=tempdir())
pooldata=popsync2pooldata(sync.file=paste0(tempdir(),"/ex.sync.gz"),poolsizes=rep(50,15))
```

---

rooted.njtree.builder *Construct and root an Neighbor-Joining tree of presumably nonadmixed leaves*

---

### Description

Construct and root an Neighbor-Joining tree of presumably nonadmixed leaves

### Usage

```
rooted.njtree.builder(  

  fstats,  

  pop.sel,  

  edge.fact = 1000,  

  plot.nj = FALSE,  

  verbose = TRUE  

)
```

**Arguments**

<code>fstats</code>	Object of class <code>fstats</code> that contains estimates of the <code>fstats</code> (see <code>compute.fstats</code> )
<code>pop.sel</code>	Names of the leaves (pops) used to build the nj tree (at least 3 required)
<code>edge.fact</code>	The multiplying factor of edges length in graph representation
<code>plot.nj</code>	If TRUE plot the Neighbor-Joining tree
<code>verbose</code>	If TRUE extra information is printed on the terminal

**Details**

A Neighbor-Joining tree is first built (using `nj` function from the package `ape`) based on the F2-distance matrix of the leaves in `pop.sel` which are presumably non-admixed (see the function `find.tree.popset` to find such groups of scaffold populations using estimated F3 and F4 test statistics). For non-admixed leaves, F2 are indeed expected to be additive along the resulting binary tree (see Lipson et al., 2013). The resulting tree is then rooted using the method described in Lipson et al. (2013) which is based on the property that the estimated heterozygosity of the root  $h_R$  equals  $h_R = 1 - Q_2(A, B)$  if A and B are two populations sharing R as the only common ancestor in the tree. This estimator should then be consistent across all the possible pairs of populations A and B that are only connected through R in the tree (i.e., that each belong to one of the two partitions of the tree defined by a root position R). Note that  $1 - Q_2(A, B) = (1 - Q_1(A))/2 + (1 - Q_1(B))/2 + F_2(A, B) = (h_A + h_B)/2 + F_2(A, B)$  where  $h_A$ ,  $h_B$  and  $F_2(A, B)$  are estimated with the function `compute.fstats`.

**Value**

A list with the following elements:

1. "n.rooted.trees": The number of possible rooted binary trees that were evaluated
2. "fitted.rooted.trees.list": a list of objects of class `fitted.graph` containing information on all the possible graphs (indexed from 1 to `n.rooted.trees`). Each tree may be visualized or further used using functions applied to objects of class `fitted.graph` (e.g., `plot`, `add.leave`)
3. `best.rooted.tree` The tree (object of class `fitted.graph`) among all the graphs within `fitted.rooted.trees.list` displaying the minimal the minimal sd over estimates of  $h_P$  (see details)
4. "root.het.est.var": For a matrix of `n.tree` rows (same order as in the list `rooted.tree`) and 4 columns with i) the average estimated root heterozygosity  $h_R$  across all the pairs of population leave that are relevant for estimation (see details); ii) the size of the range of variation and iii) the s.d. of the estimates of  $h_R$ , and iv) the number of population pairs relevant for estimation
5. "nj.tree.eval": If `n.edges > 3`, gives the five worst configuration fit (by calling the `compare.fitted.fstats` function) which are the same irrespective of rooting

**See Also**

see [fit.graph](#), [generate.graph.params](#) and [add.leaf](#).

---

show,countdata-method *Show countdata object*

---

**Description**

Show countdata object

**Usage**

```
## S4 method for signature 'countdata'  
show(object)
```

**Arguments**

object            Object of class countdata

---

show,fitted.graph-method  
*Show fitted.graph object*

---

**Description**

Show fitted.graph object

**Usage**

```
## S4 method for signature 'fitted.graph'  
show(object)
```

**Arguments**

object            Object of class fitted.graph



---

show, fstats-method      *Show fstats object*

---

### **Description**

Show fstats object

### **Usage**

```
## S4 method for signature 'fstats'  
show(object)
```

### **Arguments**

object                  Object of class fstats

---

show, graph.params-method  
*Show graph.params object*

---

### **Description**

Show graph.params object

### **Usage**

```
## S4 method for signature 'graph.params'  
show(object)
```

### **Arguments**

object                  Object of class graph.params

show,pairwisefst-method

*Show pairwisefst object*

---

### **Description**

Show pairwisefst object

### **Usage**

```
## S4 method for signature 'pairwisefst'  
show(object)
```

### **Arguments**

object            Object of class pairwisefst

---

show,pooldata-method    *Show pooldata object*

---

### **Description**

Show pooldata object

### **Usage**

```
## S4 method for signature 'pooldata'  
show(object)
```

### **Arguments**

object            Object of class pooldata

---

vcf2pooldata	<i>Convert a VCF file into a pooldata object.</i>
--------------	---

---

### Description

Convert VCF files into a pooldata object.

### Usage

```
vcf2pooldata(
  vcf.file = "",
  poolsizes = NA,
  poolnames = NA,
  min.cov.per.pool = -1,
  min.rc = 1,
  max.cov.per.pool = 1e+06,
  min.maf = -1,
  remove.indels = FALSE,
  min.dist.from.indels = 0,
  nlines.per.readblock = 1e+06,
  verbose = TRUE
)
```

### Arguments

<code>vcf.file</code>	The name (or a path) of the Popoolation sync file (might be in compressed format)
<code>poolsizes</code>	A numeric vector with haploid pool sizes
<code>poolnames</code>	A character vector with the names of pool
<code>min.cov.per.pool</code>	Minimal allowed read count (per pool). If at least one pool is not covered by at least <code>min.cov.per.pool</code> reads, the position is discarded
<code>min.rc</code>	Minimal allowed read count per base (options silenced for VarScan vcf). Bases covered by less than <code>min.rc</code> reads are discarded and considered as sequencing error. For instance, if nucleotides A, C, G and T are covered by respectively 100, 15, 0 and 1 over all the pools, setting <code>min.rc</code> to 0 will lead to discard the position (the polymorphism being considered as tri-allelic), while setting <code>min.rc</code> to 1 (or 2, 3..14) will make the position be considered as a SNP with two alleles A and C (the only read for allele T being disregarded). For VarScan vcf, markers with more than one alternative allele are discarded because the VarScan AD field only contains one alternate read count.
<code>max.cov.per.pool</code>	Maximal allowed read count (per pool). If at least one pool is covered by more than <code>min.cov.per.pool</code> reads, the position is discarded
<code>min.maf</code>	Minimal allowed Minor Allele Frequency (computed from the ratio overall read counts for the reference allele over the read coverage)

<code>remove.indels</code>	Remove indels identified using the number of characters of the alleles in the REF or ALT fields (i.e., if at least one allele is more than 1 character, the position is discarded)
<code>min.dist.from.indels</code>	Remove SNPs within <code>min.dist.from.indels</code> from an indel i.e. SNP with position <code>p</code> verifying $(\text{indel.pos} - \text{min.dist}) \leq p \leq (\text{indel.pos} + \text{min.dist} + \text{l.indels} - 1)$ where <code>l.indel=length</code> of the ref. indel allele. If <code>min.dist.from.indels &gt; 0</code> , INDELS are also removed (i.e., <code>remove.indels</code> is set to TRUE).
<code>nlines.per.readblock</code>	Number of Lines read simultaneously. Should be adapted to the available RAM.
<code>verbose</code>	If TRUE extra information is printed on the terminal

### Details

Genotype format in the vcf file for each pool is assumed to contain either i) an AD field containing allele counts separated by a comma (as produced by popular software such as GATK or samtools/bcftools) or ii) both a RD (reference allele count) and a AD (alternate allele count) as obtained with the VarScan mpileup2snp program (when run with the `-output-vcf` option). The underlying format is automatically detected by the function. For VarScan generated vcf, it should be noticed that SNPs with more than one alternate allele are discarded (because only a single count is then reported in the AD fields) making the `min.rc` unavailable. The VarScan `-min-reads2` option might replace to some extent this functionality although SNP where the two major alleles in the Pool-Seq data are different from the reference allele (e.g., expected to be more frequent when using a distantly related reference genome for mapping) will be disregarded.

### Value

A pooldata object containing 7 elements:

1. "refallele.readcount": a matrix with `nsnp` rows and `npools` columns containing read counts for the reference allele (chosen arbitrarily) in each pool
2. "readcoverage": a matrix with `nsnp` rows and `npools` columns containing read coverage in each pool
3. "snp.info": a matrix with `nsnp` rows and four columns containing respectively the contig (or chromosome) name (1st column) and position (2nd column) of the SNP; the allele taken as reference in the `refallele.readcount` matrix (3rd column); and the alternative allele (4th column)
4. "poolsizes": a vector of length `npools` containing the haploid pool sizes
5. "poolnames": a vector of length `npools` containing the names of the pools
6. "nsnp": a scalar corresponding to the number of SNPs
7. "npools": a scalar corresponding to the number of pools

### Examples

```
make.example.files(writing.dir=tempdir())
pooldata=vcf2pooldata(vcf.file=paste0(tempdir(),"/ex.vcf.gz"),poolsizes=rep(50,15))
```

# Index

add.leaf, [3](#), [27](#), [47](#)

compare.fitted.fstats, [4](#), [16](#)  
compute.f4ratio, [5](#)  
compute.fstats, [5](#), [6](#), [14](#), [19](#), [39](#)  
compute.pairwiseFST, [7](#), [7](#), [18](#), [36](#), [39](#)  
computeFST, [9](#)  
countdata (countdata-class), [10](#)  
countdata-class, [10](#)  
countdata.subset, [11](#)

find.tree.popset, [12](#)  
fit.graph, [4](#), [5](#), [14](#), [17](#), [19](#), [27](#), [29](#), [47](#)  
fitted.graph (fitted.graph-class), [16](#)  
fitted.graph-class, [16](#)  
fstats (fstats-class), [17](#)  
fstats-class, [17](#)

generate.graph.params, [4](#), [16](#), [18](#), [27](#),  
[29–31](#), [47](#)  
generate.jackknife.blocks, [20](#)  
genobypass2countdata, [6–8](#), [10](#), [12](#), [20](#)  
genobypass2pooldata, [6–8](#), [10](#), [22](#), [40](#)  
genoselestim2pooldata, [6–8](#), [10](#), [23](#), [40](#)  
genotremix2countdata, [6–8](#), [10](#), [12](#), [25](#)  
graph.builder, [26](#)  
graph.params (graph.params-class), [28](#)  
graph.params-class, [28](#)  
graph.params2qpGraphFiles, [19](#), [29](#), [29](#)  
graph.params2symbolic.fstats, [19](#), [29](#), [30](#)

heatmap, pairwiseFST-method, [31](#)

is.countdata, [33](#)  
is.fitted.graph, [33](#)  
is.fstats, [34](#)  
is.graph.params, [34](#)  
is.pairwiseFST, [34](#)  
is.pooldata, [35](#)

make.example.files, [35](#)

pairwiseFST (pairwiseFST-class), [36](#)  
pairwiseFST-class, [36](#)  
plot, fitted.graph-method, [36](#)  
plot, fstats-method, [37](#)  
plot, graph.params-method, [37](#)  
plot, pairwiseFST-method, [38](#)  
plot\_fstats, [37](#), [38](#), [38](#)  
pooldata (pooldata-class), [40](#)  
pooldata-class, [40](#)  
pooldata.subset, [40](#)  
pooldata2genobypass, [42](#)  
pooldata2genoselestim, [43](#)  
poolfstat, [44](#)  
popsync2pooldata, [6–8](#), [10](#), [40](#), [42–44](#), [45](#)

rooted.njtree.builder, [46](#)

show, countdata-method, [48](#)  
show, fitted.graph-method, [48](#)  
show, fstats-method, [49](#)  
show, graph.params-method, [49](#)  
show, pairwiseFST-method, [50](#)  
show, pooldata-method, [50](#)

vcf2pooldata, [6–8](#), [10](#), [40](#), [42–44](#), [51](#)